MySQL Cluster NDB 7.3, MySQL Cluster NDB 7.4

Abstract

This is the MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 extract from the MySQL 5.6 Reference Manual.

For legal information, see the Legal Notices.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Document generated on: 2015-12-17 (revision: 45922)

Table of Contents

Preface and Legal Notices

This is the MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 extract from the MySQL 5.6 Reference Manual.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Chapter 1 Preface and Notes

This is an extract from the Reference Manual for the MySQL Database System, version 5.6. It contains information about MySQL Cluster NDB 7.3 releases through MySQL Cluster NDB 7.3.12 and MySQL Cluster NDB 7.4 releases through MySQL Cluster NDB 7.4.9. Differences between minor versions of MySQL Cluster covered in this extract are noted in the present text with reference to the `NDBCLUSTER` storage version number (7.3.*x*, 7.4.*x*; differences between versions of the MySQL Server 5.6 software, on which MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 are based, are noted with reference to MySQL 5.6 releases (5.6.*x*). For license information, see the legal notice. This product may contain third-party code. For license information on third-party code, see Appendix A, *Licenses for Third-Party Components*.

This extract is not intended for use with older versions of the MySQL Cluster software due to the many functional and other differences between current versions (MySQL Cluster NDB 7.3, MySQL Cluster NDB 7.4) and previous versions. For information about previous MySQL Cluster versions (NDB 7.2 and earlier), see *MySQL Cluster NDB 7.2*. This extract provides information that is specific to MySQL Cluster and is not intended to replace the *MySQL 5.6 Reference Manual*, which provides additional information about MySQL 5.6 which may also be necessary for use of MySQL Cluster NDB 7.3, MySQL Cluster NDB 7.4, or both. If you are using MySQL Server 5.5 or an earlier release of the MySQL software, please refer to the appropriate manual. For example, *MySQL 5.5 Reference Manual*, covers the 5.5 series of MySQL Server releases.

If you are using MySQL 5.7, please refer to the *MySQL 5.7 Reference Manual*.

Chapter 2 MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4

This chapter contains information about *MySQL Cluster*, which is a high-availability, high-redundancy version of MySQL adapted for the distributed computing environment. Recent releases of MySQL Cluster use version 7 of the `NDB` storage engine (also known as `NDBCLUSTER`) to enable running several computers with MySQL servers and other software in a cluster; the latest releases available for production, MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4, incorporate `NDB` engine versions 7.3 and 7.4, respectively.

Support for the `NDB` storage engine is not included in standard MySQL Server 5.6 binaries built by Oracle. Instead, users of MySQL Cluster binaries from Oracle should upgrade to the most recent binary release of MySQL Cluster for supported platforms—these include RPMs that should work with most Linux distributions. MySQL Cluster users who build from source should use the sources provided for MySQL Cluster. (Locations where the sources can be obtained are listed later in this section.)

This chapter contains information about MySQL Cluster NDB 7.3 releases through 5.6.27-ndb-7.3.12 as well as MySQL Cluster NDB 7.4 releases through 5.6.27-ndb-7.4.9. Currently, both the MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 release series are Generally Available (GA). MySQL Cluster NDB 7.2 and MySQL Cluster NDB 7.1 are previous GA release series which are still available. We recommend that new deployments use MySQL Cluster NDB 7.4. For information about MySQL Cluster NDB 7.2, see MySQL Cluster NDB 7.2; for information about MySQL Cluster NDB 7.1 and earlier versions of MySQL Cluster, see http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.

Supported Platforms. MySQL Cluster is currently available and supported on a number of platforms. For exact levels of support available for on specific combinations of operating system versions, operating system distributions, and hardware platforms, please refer to http://www.mysql.com/support/supportedplatforms/cluster.html.

Availability. MySQL Cluster binary and source packages are available for supported platforms from http://dev.mysql.com/downloads/cluster/.

MySQL Cluster release numbers. MySQL Cluster follows a somewhat different release pattern from the mainline MySQL Server 5.6 series of releases. In this *Manual* and other MySQL documentation, we identify these and later MySQL Cluster releases employing a version number that begins with “NDB”. This version number is that of the `NDBCLUSTER` storage engine used in the release, and not of the MySQL server version on which the MySQL Cluster release is based.

Version strings used in MySQL Cluster software. The version string displayed by MySQL Cluster programs uses this format:

```
mysql-mysql_server_version-ndb-ndb_engine_version
```

`mysql_server_version` represents the version of the MySQL Server on which the MySQL Cluster release is based. For all MySQL Cluster NDB 7.3 and current MySQL Cluster NDB 7.4 releases, this is “5.6”. *`ndb_engine_version`* is the version of the `NDB` storage engine used by this release of the MySQL Cluster software. You can see this format used in the `mysql` client, as shown here:

```
shell> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.27-ndb-7.4.9 Source distribution
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SELECT VERSION()\G
*************************** 1. row ***************************
VERSION(): 5.6.27-ndb-7.4.9
1 row in set (0.00 sec)
```

This version string is also displayed in the output of the `SHOW` command in the `ndb_mgm` client:

```
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @10.0.10.6  (5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=2    @10.0.10.8  (5.6.27-ndb-7.4.9, Nodegroup: 0)
[ndb_mgmd(MGM)] 1 node(s)
id=3    @10.0.10.2  (5.6.27-ndb-7.4.9)
[mysqld(API)]   2 node(s)
id=4    @10.0.10.10  (5.6.27-ndb-7.4.9)
id=5 (not connected, accepting connect from any host)
```

The version string identifies the mainline MySQL version from which the MySQL Cluster release was branched and the version of the `NDB` storage engine used. For example, the full version string for MySQL Cluster NDB 7.4.4 (the first MySQL Cluster NDB 7.4 GA release) is `mysql-5.6.23-ndb-7.4.4`. From this we can determine the following:

- Since the portion of the version string preceding "`-ndb-`" is the base MySQL Server version, this means that MySQL Cluster NDB 7.4.4 derives from MySQL 5.6.23, and contains all feature enhancements and bug fixes from MySQL 5.6 up to and including MySQL 5.6.23.

- Since the portion of the version string following "`-ndb-`" represents the version number of the `NDB` (or `NDBCLUSTER`) storage engine, MySQL Cluster NDB 7.4.4 uses version 7.4.4 of the `NDBCLUSTER` storage engine.

New MySQL Cluster releases are numbered according to updates in the `NDB` storage engine, and do not necessarily correspond in a one-to-one fashion with mainline MySQL Server releases. For example, MySQL Cluster NDB 7.4.4 (as previously noted) is based on MySQL 5.6.23, while MySQL Cluster NDB 7.4.3 was based on MySQL 5.6.22 (version string: `mysql-5.6.22-ndb-7.4.3`).

Compatibility with standard MySQL 5.6 releases. While many standard MySQL schemas and applications can work using MySQL Cluster, it is also true that unmodified applications and database schemas may be slightly incompatible or have suboptimal performance when run using MySQL Cluster (see Section 3.6, "Known Limitations of MySQL Cluster"). Most of these issues can be overcome, but this also means that you are very unlikely to be able to switch an existing application datastore—that currently uses, for example, `MyISAM` or `InnoDB`—to use the `NDB` storage engine without allowing for the possibility of changes in schemas, queries, and applications. In addition, the MySQL Server and MySQL Cluster codebases diverge considerably, so that the standard `mysqld` cannot function as a drop-in replacement for the version of `mysqld` supplied with MySQL Cluster.

MySQL Cluster development source trees. MySQL Cluster development trees can also be accessed from https://github.com/mysql/mysql-server.

The MySQL Cluster development sources maintained at https://github.com/mysql/mysql-server are licensed under the GPL. For information about obtaining MySQL sources using Bazaar and building them yourself, see Installing MySQL Using a Development Source Tree.

> **Note**
>
> As with MySQL Server 5.6, MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 releases are built using `CMake`.

Currently, MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 releases are Generally Available (GA). We recommend that new deployments use MySQL Cluster NDB 7.4. MySQL Cluster NDB 7.1 and earlier versions are no longer in active development. For an overview of major features added in MySQL Cluster NDB 7.4, see Section 3.4.2, "MySQL Cluster Development in MySQL Cluster NDB 7.4". For similar information about MySQL Cluster NDB 7.3, see Section 3.4.1, "MySQL Cluster Development in MySQL Cluster NDB 7.3". For an overview of major features added in past MySQL Cluster releases, see http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-development.html, and MySQL Cluster Development History.

This chapter represents a work in progress, and its contents are subject to revision as MySQL Cluster continues to evolve. Additional information regarding MySQL Cluster can be found on the MySQL Web site at http://www.mysql.com/products/cluster/.

Additional Resources. More information about MySQL Cluster can be found in the following places:

- For answers to some commonly asked questions about MySQL Cluster, see Appendix B, *MySQL 5.6 FAQ: MySQL Cluster*.
- The MySQL Cluster mailing list: http://lists.mysql.com/cluster.
- The MySQL Cluster Forum: http://forums.mysql.com/list.php?25.
- Many MySQL Cluster users and developers blog about their experiences with MySQL Cluster, and make feeds of these available through PlanetMySQL.

Chapter 3 MySQL Cluster Overview

Table of Contents

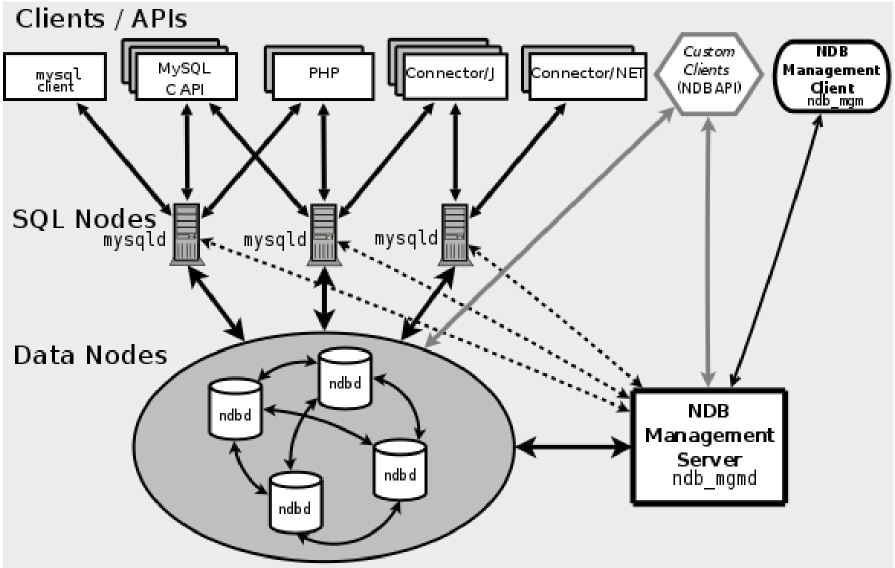
MySQL Cluster is a technology that enables clustering of in-memory databases in a shared-nothing system. The shared-nothing architecture enables the system to work with very inexpensive hardware, and with a minimum of specific requirements for hardware or software.

MySQL Cluster is designed not to have any single point of failure. In a shared-nothing system, each component is expected to have its own memory and disk, and the use of shared storage mechanisms such as network shares, network file systems, and SANs is not recommended or supported.

MySQL Cluster integrates the standard MySQL server with an in-memory clustered storage engine called `NDB` (which stands for "*N*etwork *D*ata*B*ase"). In our documentation, the term `NDB` refers to the part of the setup that is specific to the storage engine, whereas "MySQL Cluster" refers to the combination of one or more MySQL servers with the `NDB` storage engine.

A MySQL Cluster consists of a set of computers, known as *hosts*, each running one or more processes. These processes, known as *nodes*, may include MySQL servers (for access to NDB data), data nodes (for storage of the data), one or more management servers, and possibly other specialized data access programs. The relationship of these components in a MySQL Cluster is shown here:

Figure 3.1 MySQL Cluster Components



All these programs work together to form a MySQL Cluster (see Chapter 6, *MySQL Cluster Programs*. When data is stored by the NDB storage engine, the tables (and table data) are stored in the data nodes. Such tables are directly accessible from all other MySQL servers (SQL nodes) in the cluster. Thus, in a payroll application storing data in a cluster, if one application updates the salary of an employee, all other MySQL servers that query this data can see this change immediately.

Although a MySQL Cluster SQL node uses the mysqld server daemon, it differs in a number of critical respects from the mysqld binary supplied with the MySQL 5.6 distributions, and the two versions of mysqld are not interchangeable.

In addition, a MySQL server that is not connected to a MySQL Cluster cannot use the NDB storage engine and cannot access any MySQL Cluster data.

The data stored in the data nodes for MySQL Cluster can be mirrored; the cluster can handle failures of individual data nodes with no other impact than that a small number of transactions are aborted due to losing the transaction state. Because transactional applications are expected to handle transaction failure, this should not be a source of problems.

Individual nodes can be stopped and restarted, and can then rejoin the system (cluster). Rolling restarts (in which all nodes are restarted in turn) are used in making configuration changes and software upgrades (see Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”). Rolling restarts are also used as part of the process of adding new data nodes online (see Section 7.13, “Adding MySQL Cluster Data Nodes Online”). For more information about data nodes, how they are organized in a MySQL Cluster, and how they handle and store MySQL Cluster data, see Section 3.2, “MySQL Cluster Nodes, Node Groups, Replicas, and Partitions”.

Backing up and restoring MySQL Cluster databases can be done using the NDB-native functionality found in the MySQL Cluster management client and the ndb_restore program included in the MySQL Cluster distribution. For more information, see Section 7.3, “Online Backup of MySQL Cluster”, and Section 6.20, “ndb_restore — Restore a MySQL Cluster Backup”. You can also use the standard MySQL functionality provided for this purpose in mysqldump and the MySQL server. See mysqldump — A Database Backup Program, for more information.

MySQL Cluster nodes can use a number of different transport mechanisms for inter-node communications, including TCP/IP using standard 100 Mbps or faster Ethernet hardware. It is also possible to use the high-speed *Scalable Coherent Interface* (SCI) protocol with MySQL Cluster, although this is not required to use MySQL Cluster. SCI requires special hardware and software; see Section 5.4, “Using High-Speed Interconnects with MySQL Cluster”, for more about SCI and using it with MySQL Cluster.

3.1 MySQL Cluster Core Concepts

NDBCLUSTER (also known as NDB) is an in-memory storage engine offering high-availability and data-persistence features.

The NDBCLUSTER storage engine can be configured with a range of failover and load-balancing options, but it is easiest to start with the storage engine at the cluster level. MySQL Cluster's NDB storage engine contains a complete set of data, dependent only on other data within the cluster itself.

The “Cluster” portion of MySQL Cluster is configured independently of the MySQL servers. In a MySQL Cluster, each part of the cluster is considered to be a *node*.

> **Note**
>
> In many contexts, the term “node” is used to indicate a computer, but when discussing MySQL Cluster it means a *process*. It is possible to run multiple nodes on a single computer; for a computer on which one or more cluster nodes are being run we use the term *cluster host*.

There are three types of cluster nodes, and in a minimal MySQL Cluster configuration, there will be at least three nodes, one of each of these types:

- *Management node*: The role of this type of node is to manage the other nodes within the MySQL Cluster, performing such functions as providing configuration data, starting and stopping nodes, running backup, and so forth. Because this node type manages the configuration of the other nodes, a node of this type should be started first, before any other node. An MGM node is started with the command `ndb_mgmd`.

- *Data node*: This type of node stores cluster data. There are as many data nodes as there are replicas, times the number of fragments (see Section 3.2, “MySQL Cluster Nodes, Node Groups, Replicas, and Partitions”). For example, with two replicas, each having two fragments, you need four data nodes. One replica is sufficient for data storage, but provides no redundancy; therefore, it is recommended to have 2 (or more) replicas to provide redundancy, and thus high availability. A data node is started with the command `ndbd` (see Section 6.1, “`ndbd` — The MySQL Cluster Data Node Daemon”) or `ndbmtd` (see Section 6.3, “`ndbmtd` — The MySQL Cluster Data Node Daemon (Multi-Threaded)”).

  MySQL Cluster tables are normally stored completely in memory rather than on disk (this is why we refer to MySQL Cluster as an *in-memory* database). However, some MySQL Cluster data can be stored on disk; see Section 7.12, “MySQL Cluster Disk Data Tables”, for more information.

- *SQL node*: This is a node that accesses the cluster data. In the case of MySQL Cluster, an SQL node is a traditional MySQL server that uses the `NDBCLUSTER` storage engine. An SQL node is a `mysqld` process started with the `--ndbcluster` and `--ndb-connectstring` options, which are explained elsewhere in this chapter, possibly with additional MySQL server options as well.

  An SQL node is actually just a specialized type of *API node*, which designates any application which accesses MySQL Cluster data. Another example of an API node is the `ndb_restore` utility that is used to restore a cluster backup. It is possible to write such applications using the NDB API. For basic information about the NDB API, see Getting Started with the NDB API.

> **Important**
>
> It is not realistic to expect to employ a three-node setup in a production environment. Such a configuration provides no redundancy; to benefit from MySQL Cluster's high-availability features, you must use multiple data and SQL nodes. The use of multiple management nodes is also highly recommended.

For a brief introduction to the relationships between nodes, node groups, replicas, and partitions in MySQL Cluster, see Section 3.2, “MySQL Cluster Nodes, Node Groups, Replicas, and Partitions”.

Configuration of a cluster involves configuring each individual node in the cluster and setting up individual communication links between nodes. MySQL Cluster is currently designed with the intention that data nodes are homogeneous in terms of processor power, memory space, and bandwidth. In addition, to provide a single point of configuration, all configuration data for the cluster as a whole is located in one configuration file.

The management server manages the cluster configuration file and the cluster log. Each node in the cluster retrieves the configuration data from the management server, and so requires a way to determine where the management server resides. When interesting events occur in the data nodes, the nodes transfer information about these events to the management server, which then writes the information to the cluster log.

In addition, there can be any number of cluster client processes or applications. These include standard MySQL clients, `NDB`-specific API programs, and management clients. These are described in the next few paragraphs.

Standard MySQL clients. MySQL Cluster can be used with existing MySQL applications written in PHP, Perl, C, C++, Java, Python, Ruby, and so on. Such client applications send SQL statements to and receive responses from MySQL servers acting as MySQL Cluster SQL nodes in much the same way that they interact with standalone MySQL servers.

MySQL clients using a MySQL Cluster as a data source can be modified to take advantage of the ability to connect with multiple MySQL servers to achieve load balancing and failover. For example, Java clients using Connector/J 5.0.6 and later can use `jdbc:mysql:loadbalance://` URLs (improved in Connector/J 5.1.7) to achieve load balancing transparently; for more information about using Connector/J with MySQL Cluster, see Using Connector/J with MySQL Cluster.

NDB client programs. Client programs can be written that access MySQL Cluster data directly from the `NDBCLUSTER` storage engine, bypassing any MySQL Servers that may be connected to the cluster, using the *NDB API*, a high-level C++ API. Such applications may be useful for specialized purposes where an SQL interface to the data is not needed. For more information, see The NDB API.

`NDB`-specific Java applications can also be written for MySQL Cluster using the *MySQL Cluster Connector for Java*. This MySQL Cluster Connector includes *ClusterJ*, a high-level database API similar to object-relational mapping persistence frameworks such as Hibernate and JPA that connect directly to `NDBCLUSTER`, and so does not require access to a MySQL Server. Support is also provided in MySQL Cluster NDB 7.1 and later for *ClusterJPA*, an OpenJPA implementation for MySQL Cluster that leverages the strengths of ClusterJ and JDBC; ID lookups and other fast operations are performed using ClusterJ (bypassing the MySQL Server), while more complex queries that can benefit from MySQL's query optimizer are sent through the MySQL Server, using JDBC. See Java and MySQL Cluster, and The ClusterJ API and Data Object Model, for more information.

MySQL Cluster NDB 7.3 and later also supports applications written in JavaScript using Node.js. The MySQL Connector for JavaScript includes adapters for direct access to the `NDB` storage engine and as well as for the MySQL Server. Applications using this Connector are typically event-driven and use a domain object model similar in many ways to that employed by ClusterJ. For more information, see MySQL NoSQL Connector for JavaScript.

The Memcache API for MySQL Cluster, implemented as the loadable *ndbmemcache* storage engine for memcached version 1.6 and later, can be used to provide a persistent MySQL Cluster data store, accessed using the memcache protocol.

The standard `memcached` caching engine is included in MySQL Cluster NDB 7.3 and later distributions. Each `memcached` server has direct access to data stored in MySQL Cluster, but is also able to cache data locally and to serve (some) requests from this local cache.

For more information, see ndbmemcache—Memcache API for MySQL Cluster.

Management clients. These clients connect to the management server and provide commands for starting and stopping nodes gracefully, starting and stopping message tracing (debug versions only), showing node versions and status, starting and stopping backups, and so on. An example of this type of program is the ndb_mgm management client supplied with MySQL Cluster (see Section 6.5, "ndb_mgm — The MySQL Cluster Management Client"). Such applications can be written using the *MGM API*, a C-language API that communicates directly with one or more MySQL Cluster management servers. For more information, see The MGM API.

Oracle also makes available MySQL Cluster Manager, which provides an advanced command-line interface simplifying many complex MySQL Cluster management tasks, such restarting a MySQL Cluster with a large number of nodes. The MySQL Cluster Manager client also supports commands for getting and setting the values of most node configuration parameters as well as mysqld server options and variables relating to MySQL Cluster. See MySQL™ Cluster Manager 1.3.6 User Manual, for more information.

Event logs. MySQL Cluster logs events by category (startup, shutdown, errors, checkpoints, and so on), priority, and severity. A complete listing of all reportable events may be found in Section 7.6, "Event Reports Generated in MySQL Cluster". Event logs are of the two types listed here:

- *Cluster log*: Keeps a record of all desired reportable events for the cluster as a whole.
- *Node log*: A separate log which is also kept for each individual node.

> **Note**
>
> Under normal circumstances, it is necessary and sufficient to keep and examine only the cluster log. The node logs need be consulted only for application development and debugging purposes.

Checkpoint. Generally speaking, when data is saved to disk, it is said that a *checkpoint* has been reached. More specific to MySQL Cluster, a checkpoint is a point in time where all committed transactions are stored on disk. With regard to the NDB storage engine, there are two types of checkpoints which work together to ensure that a consistent view of the cluster's data is maintained. These are shown in the following list:

- *Local Checkpoint (LCP)*: This is a checkpoint that is specific to a single node; however, LCPs take place for all nodes in the cluster more or less concurrently. An LCP involves saving all of a node's data to disk, and so usually occurs every few minutes. The precise interval varies, and depends upon the amount of data stored by the node, the level of cluster activity, and other factors.
- *Global Checkpoint (GCP)*: A GCP occurs every few seconds, when transactions for all nodes are synchronized and the redo-log is flushed to disk.

For more information about the files and directories created by local checkpoints and global checkpoints, see MySQL Cluster Data Node File System Directory Files.

3.2 MySQL Cluster Nodes, Node Groups, Replicas, and Partitions

This section discusses the manner in which MySQL Cluster divides and duplicates data for storage.

A number of concepts central to an understanding of this topic are discussed in the next few paragraphs.

(Data) Node. An ndbd process, which stores a *replica* —that is, a copy of the *partition* (see below) assigned to the node group of which the node is a member.

Each data node should be located on a separate computer. While it is also possible to host multiple ndbd processes on a single computer, such a configuration is not supported.

It is common for the terms "node" and "data node" to be used interchangeably when referring to an ndbd process; where mentioned, management nodes (ndb_mgmd processes) and SQL nodes (mysqld processes) are specified as such in this discussion.

Node Group. A node group consists of one or more nodes, and stores partitions, or sets of *replicas* (see next item).

The number of node groups in a MySQL Cluster is not directly configurable; it is a function of the number of data nodes and of the number of replicas (`NoOfReplicas` configuration parameter), as shown here:

```
[number_of_node_groups] = number_of_data_nodes / NoOfReplicas
```

Thus, a MySQL Cluster with 4 data nodes has 4 node groups if `NoOfReplicas` is set to 1 in the `config.ini` file, 2 node groups if `NoOfReplicas` is set to 2, and 1 node group if `NoOfReplicas` is set to 4. Replicas are discussed later in this section; for more information about `NoOfReplicas`, see Section 5.3.6, "Defining MySQL Cluster Data Nodes".

> **Note**
>
> All node groups in a MySQL Cluster must have the same number of data nodes.

You can add new node groups (and thus new data nodes) online, to a running MySQL Cluster; see Section 7.13, "Adding MySQL Cluster Data Nodes Online", for more information.

Partition. This is a portion of the data stored by the cluster. There are as many cluster partitions as nodes participating in the cluster. Each node is responsible for keeping at least one copy of any partitions assigned to it (that is, at least one replica) available to the cluster.

A replica belongs entirely to a single node; a node can (and usually does) store several replicas.

NDB and user-defined partitioning. MySQL Cluster normally partitions `NDBCLUSTER` tables automatically. However, it is also possible to employ user-defined partitioning with `NDBCLUSTER` tables. This is subject to the following limitations:

1. Only the `KEY` and `LINEAR KEY` partitioning schemes are supported in production with `NDB` tables.

2. When using `ndbd`, the maximum number of partitions that may be defined explicitly for any `NDB` table is `8 * [number of node groups]`. (The number of node groups in a MySQL Cluster is determined as discussed previously in this section.)

   When using `ndbmtd`, this maximum is also affected by the number of local query handler threads, which is determined by the value of the `MaxNoOfExecutionThreads` configuration parameter. In such cases, the maximum number of partitions that may be defined explicitly for an `NDB` table is equal to `4 * MaxNoOfExecutionThreads * [number of node groups]`.

   See Section 6.3, "`ndbmtd` — The MySQL Cluster Data Node Daemon (Multi-Threaded)", for more information.

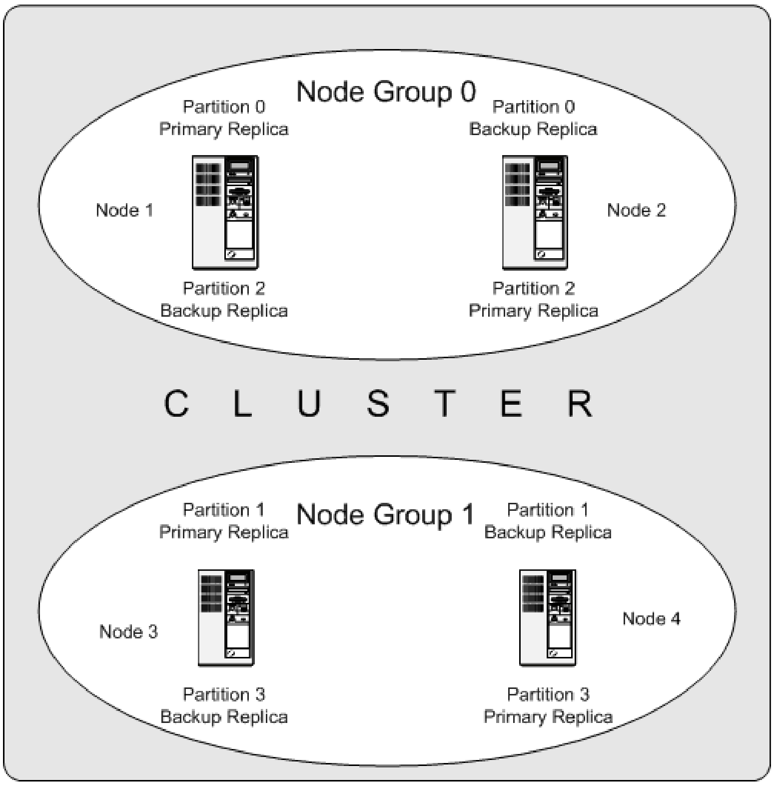
For more information relating to MySQL Cluster and user-defined partitioning, see Section 3.6, "Known Limitations of MySQL Cluster", and Partitioning Limitations Relating to Storage Engines.

Replica. This is a copy of a cluster partition. Each node in a node group stores a replica. Also sometimes known as a *partition replica*. The number of replicas is equal to the number of nodes per node group.

The following diagram illustrates a MySQL Cluster with four data nodes, arranged in two node groups of two nodes each; nodes 1 and 2 belong to node group 0, and nodes 3 and 4 belong to node group 1.

> **Note**
>
> Only data (`ndbd`) nodes are shown here; although a working cluster requires an `ndb_mgm` process for cluster management and at least one SQL node to access the data stored by the cluster, these have been omitted in the figure for clarity.

Figure 3.2 MySQL Cluster with Two Node Groups

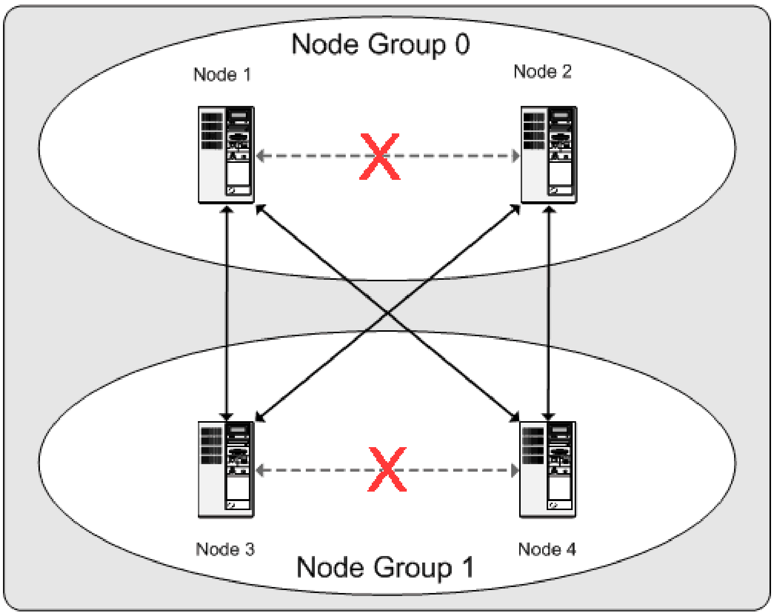


The data stored by the cluster is divided into four partitions, numbered 0, 1, 2, and 3. Each partition is stored—in multiple copies—on the same node group. Partitions are stored on alternate node groups as follows:

- Partition 0 is stored on node group 0; a *primary replica* (primary copy) is stored on node 1, and a *backup replica* (backup copy of the partition) is stored on node 2.
- Partition 1 is stored on the other node group (node group 1); this partition's primary replica is on node 3, and its backup replica is on node 4.
- Partition 2 is stored on node group 0. However, the placing of its two replicas is reversed from that of Partition 0; for Partition 2, the primary replica is stored on node 2, and the backup on node 1.
- Partition 3 is stored on node group 1, and the placement of its two replicas are reversed from those of partition 1. That is, its primary replica is located on node 4, with the backup on node 3.

What this means regarding the continued operation of a MySQL Cluster is this: so long as each node group participating in the cluster has at least one node operating, the cluster has a complete copy of all data and remains viable. This is illustrated in the next diagram.

Figure 3.3 Nodes Required for a 2x2 Cluster



In this example, where the cluster consists of two node groups of two nodes each, any combination of at least one node in node group 0 and at least one node in node group 1 is sufficient to keep the cluster "alive" (indicated by arrows in the diagram). However, if *both* nodes from *either* node group fail, the remaining two nodes are not sufficient (shown by the arrows marked out with an **X**); in either case, the cluster has lost an entire partition and so can no longer provide access to a complete set of all cluster data.

3.3 MySQL Cluster Hardware, Software, and Networking Requirements

One of the strengths of MySQL Cluster is that it can be run on commodity hardware and has no unusual requirements in this regard, other than for large amounts of RAM, due to the fact that all live data storage is done in memory. (It is possible to reduce this requirement using Disk Data tables—see Section 7.12, "MySQL Cluster Disk Data Tables", for more information about these.) Naturally, multiple and faster CPUs can enhance performance. Memory requirements for other MySQL Cluster processes are relatively small.

The software requirements for MySQL Cluster are also modest. Host operating systems do not require any unusual modules, services, applications, or configuration to support MySQL Cluster. For supported operating systems, a standard installation should be sufficient. The MySQL software requirements are simple: all that is needed is a production release of MySQL Cluster. It is not strictly necessary to compile MySQL yourself merely to be able to use MySQL Cluster. We assume that you are using the binaries appropriate to your platform, available from the MySQL Cluster software downloads page at http://dev.mysql.com/downloads/cluster/.

For communication between nodes, MySQL Cluster supports TCP/IP networking in any standard topology, and the minimum expected for each host is a standard 100 Mbps Ethernet card, plus a switch, hub, or router to provide network connectivity for the cluster as a whole. We strongly recommend that a MySQL Cluster be run on its own subnet which is not shared with machines not forming part of the cluster for the following reasons:

- **Security.** Communications between MySQL Cluster nodes are not encrypted or shielded in any way. The only means of protecting transmissions within a MySQL Cluster is to run your MySQL Cluster on a protected network. If you intend to use MySQL Cluster for Web applications, the cluster should definitely reside behind your firewall and not in your network's De-Militarized Zone (DMZ) or elsewhere.

See Section 7.11.1, “MySQL Cluster Security and Networking Issues”, for more information.

- **Efficiency.** Setting up a MySQL Cluster on a private or protected network enables the cluster to make exclusive use of bandwidth between cluster hosts. Using a separate switch for your MySQL Cluster not only helps protect against unauthorized access to MySQL Cluster data, it also ensures that MySQL Cluster nodes are shielded from interference caused by transmissions between other computers on the network. For enhanced reliability, you can use dual switches and dual cards to remove the network as a single point of failure; many device drivers support failover for such communication links.

Network communication and latency. MySQL Cluster requires communication between data nodes and API nodes (including SQL nodes), as well as between data nodes and other data nodes, to execute queries and updates. Communication latency between these processes can directly affect the observed performance and latency of user queries. In addition, to maintain consistency and service despite the silent failure of nodes, MySQL Cluster uses heartbeating and timeout mechanisms which treat an extended loss of communication from a node as node failure. This can lead to reduced redundancy. Recall that, to maintain data consistency, a MySQL Cluster shuts down when the last node in a node group fails. Thus, to avoid increasing the risk of a forced shutdown, breaks in communication between nodes should be avoided wherever possible.

The failure of a data or API node results in the abort of all uncommitted transactions involving the failed node. Data node recovery requires synchronization of the failed node's data from a surviving data node, and re-establishment of disk-based redo and checkpoint logs, before the data node returns to service. This recovery can take some time, during which the Cluster operates with reduced redundancy.

Heartbeating relies on timely generation of heartbeat signals by all nodes. This may not be possible if the node is overloaded, has insufficient machine CPU due to sharing with other programs, or is experiencing delays due to swapping. If heartbeat generation is sufficiently delayed, other nodes treat the node that is slow to respond as failed.

This treatment of a slow node as a failed one may or may not be desirable in some circumstances, depending on the impact of the node's slowed operation on the rest of the cluster. When setting timeout values such as `HeartbeatIntervalDbDb` and `HeartbeatIntervalDbApi` for MySQL Cluster, care must be taken care to achieve quick detection, failover, and return to service, while avoiding potentially expensive false positives.

Where communication latencies between data nodes are expected to be higher than would be expected in a LAN environment (on the order of 100 µs), timeout parameters must be increased to ensure that any allowed periods of latency periods are well within configured timeouts. Increasing timeouts in this way has a corresponding effect on the worst-case time to detect failure and therefore time to service recovery.

LAN environments can typically be configured with stable low latency, and such that they can provide redundancy with fast failover. Individual link failures can be recovered from with minimal and controlled latency visible at the TCP level (where MySQL Cluster normally operates). WAN environments may offer a range of latencies, as well as redundancy with slower failover times. Individual link failures may require route changes to propagate before end-to-end connectivity is restored. At the TCP level this can appear as large latencies on individual channels. The worst-case observed TCP latency in these scenarios is related to the worst-case time for the IP layer to reroute around the failures.

SCI support. It is also possible to use the high-speed Scalable Coherent Interface (SCI) with MySQL Cluster, but this is not a requirement. See Section 5.4, “Using High-Speed Interconnects with MySQL Cluster”, for more about this protocol and its use with MySQL Cluster.

3.4 MySQL Cluster Development History

This section lists changes in the implementation of MySQL Cluster in MySQL MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4, as compared to MySQL Cluster NDB 7.2 and earlier releases. Changes and features most likely to be of interest are shown in the following two tables:

MySQL Cluster NDB 7.3
MySQL Cluster NDB 7.3 is based on MySQL 5.6. For more information about new features in MySQL Server 5.6, see What Is New in MySQL 5.6.
MySQL Cluster NDB 7.3 supports foreign key constraints on tables. See FOREIGN KEY Constraints, and Using FOREIGN KEY Constraints, for more information.
MySQL Cluster NDB 7.3 provides support for Node.js using the MySQL NoSQL Connector for JavaScript. See MySQL NoSQL Connector for JavaScript, for more information.

MySQL Cluster NDB 7.4
MySQL Cluster NDB 7.4 is based on MySQL 5.6 (For more information about new features in MySQL Server 5.6, see What Is New in MySQL 5.6)
MySQL Cluster Replication conflict detection and resolution enhancements, including extensions to conflict exceptions tables (see Section 8.11, “MySQL Cluster Replication Conflict Resolution”)
Improvements in the management of circular (“active-active”) replication; primary/secondary assignment with `ndb_slave_conflict_role`
Per-fragment memory usage reporting in the `memory_per_fragment` table
A number of performance improvements, including the following enhancements: • Faster initial allocation of memory • Increased parallelization of local checkpoints (LCPs now support 32 fragments rather than 2) A group of configuration parameters (`MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNoderestart`, `MaxDiskWriteSpeedOwnRestart`) introduced in this version provides improved control over disk writes during LCPs Information about recent disk writes is available in the `disk_write_speed_base`, `disk_write_speed_aggregate`, and `disk_write_speed_aggregate_node` tables added to the `ndbinfo` database in the this version • Faster times for restoring a MySQL Cluster from backup • Optimization of the NDB receive thread
Improved error and other reporting during node restarts

This section contains information about MySQL Cluster NDB 7.4 releases through 5.6.27-ndb-7.4.9, which is now available as a General Availability release. MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.2 are previous GA release series which are still supported, although we recommend that new deployments use MySQL Cluster NDB 7.4. For information about MySQL Cluster NDB 7.2 and previous MySQL Cluster releases, see MySQL Cluster NDB 7.2, in the *MySQL 5.5 Reference Manual*.

3.4.1 MySQL Cluster Development in MySQL Cluster NDB 7.3

The following improvements to MySQL Cluster have been made in MySQL Cluster NDB 7.3:

- **Based on MySQL Server 5.6.** MySQL Cluster NDB 7.3 is based on MySQL Server 5.6, so that MySQL Cluster users can benefit from MySQL 5.6's improvements in scalability and performance monitoring. As with MySQL 5.6, MySQL Cluster NDB 7.3 uses `CMake` for configuring and building from source. For more information about changes and improvements in MySQL 5.6, see What Is New in MySQL 5.6.

- **Foreign keys.** Tables created using the `NDB` storage engine version 7.3.0 and later provide support for foreign key constraints. (This includes all MySQL Cluster NDB 7.3 releases.) For general information about how MySQL 5.6 and MySQL Cluster NDB 7.3 handle foreign keys, see FOREIGN KEY Constraints. For syntax and related information, see CREATE TABLE Syntax, and Using FOREIGN KEY Constraints.

- **Node.js support.** MySQL Cluster NDB 7.3 also supports applications written in JavaScript using Node.js. The MySQL Connector for JavaScript includes adapters for direct access to the NDB storage engine and as well as for the MySQL Server. Applications using this Connector are typically event-driven and use a domain object model similar in many ways to that employed by ClusterJ. For more information, see MySQL NoSQL Connector for JavaScript.

MySQL Cluster NDB 7.3 is also supported by MySQL Cluster Manager, which provides an advanced command-line interface that can simplify many complex MySQL Cluster management tasks. See MySQL™ Cluster Manager 1.3.6 User Manual, for more information.

3.4.2 MySQL Cluster Development in MySQL Cluster NDB 7.4

The following improvements to MySQL Cluster have been made in MySQL Cluster NDB 7.4:

- **Conflict detection and resolution enhancements.** A reserved column name namespace `NDB$` is now employed for exceptions table metacolumns, allowing an arbitrary subset of main table columns to be recorded, even if they are not part of the original table's primary key.

  Recording the complete original primary key is no longer required, due to the fact that matching against exceptions table columns is now done by name and type only. It is now also possible for you to record values of columns which not are part of the main table's primary key in the exceptions table.

  Read conflict detection is now possible. All rows read by the conflicting transaction are flagged, and logged in the exceptions table. Rows inserted in the same transaction are not included among the rows read or logged. This read tracking depends on the slave having an exclusive read lock which requires setting `ndb_log_exclusive_reads` in advance. See Read conflict detection and resolution, for more information and examples.

  Existing exceptions tables remain supported. For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- **Circular (“active-active”) replication improvements.** When using a circular or “active-active” MySQL Cluster Replication topology, you can assign one of the roles of primary of secondary to a given MySQL Cluster using the `ndb_slave_conflict_role` server system variable, which can be employed when failing over from a MySQL Cluster acting as primary, or when using conflict detection and resolution with `NDB$EPOCH2()` and `NDB$EPOCH2_TRANS()` (MySQL Cluster NDB 7.4.2 and later), which support delete-delete conflict handling.

  See the description of the `ndb_slave_conflict_role` variable, as well as NDB$EPOCH2(), for more information. See also Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- **Per-fragment memory usage reporting.** You can now obtain data about memory usage by individual MySQL Cluster fragments from the `memory_per_fragment` view, added in MySQL Cluster NDB 7.4.1 to the `ndbinfo` information database. For more information, see Section 7.10.17, “The ndbinfo memory_per_fragment Table”.

- **Node restart improvements.** MySQL Cluster NDB 7.4 includes a number of improvements which decrease the time needed for data nodes to be restarted. These are described in the following list:

  - Memory allocated that is allocated on node startup cannot be used until it has been touched, which causes the operating system to set aside the actual physical memory required. In previous versions of MySQL Cluster, the process of touching each page of memory that was allocated was singlethreaded, which made it relatively time-consuming. This process has now been reimplimented with multithreading. In tests with 16 threads, touch times on the order of 3 times shorter than with a single thread were observed.

  - Increased parallelization of local checkpoints; in MySQL Cluster NDB 7.4, LCPs now support 32 fragments rather than 2 as before. This greatly increases utilization of CPU power that would otherwise go unused, and can make LCPs faster by up to a factor of 10; this speedup in turn can greatly improve node restart times.

The degree of parallelization used for the node copy phase during node and system restarts can be controlled in MySQL Cluster NDB 7.4.3 and later by setting the `MaxParallelCopyInstances` data node configuration parameter to a nonzero value.

- Reporting on disk writes is provided by new `ndbinfo` tables `disk_write_speed_base`, `disk_write_speed_aggregate`, and `disk_write_speed_aggregate_node`, which provide information about the speed of disk writes for each LDM thread that is in use.

  This release also adds the data node configuration parameters `MinDiskWriteSpeed`, `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` to control write speeds for LCPs and backups when the present node, another node, or no node is currently restarting.

  These changes are intended to supersede configuration of disk writes using the `DiskCheckpointSpeed` and `DiskCheckpointSpeedInRestart` configuration parameters. These 2 parameters have now been deprecated, and are subject to removal in a future MySQL Cluster release.

- Faster times for restoring a MySQL Cluster from backup have been obtained by replacing delayed signals found at a point which was found to be critical to performance with normal (undelayed) signals. The elimination or replacement of these unnecessary delayed signals should noticeably reduce the amount of time required to back up a MySQL Cluster, or to restore a MySQL Cluster from backup.

- Several internal methods relating to the `NDB` receive thread have been optimized, to increase the efficiency of SQL processing by `NDB`. The receiver thread at time may have to process several million received records per second, so it is critical that it not perform unnecessary work or waste resources when retrieving records from MySQL Cluster data nodes.

- **Improved reporting of MySQL Cluster restarts and start phases.** The `restart_info` table (included in the `ndbinfo` information database beginning with MySQL Cluster NDB 7.4.2) provides current status and timing information about node and system restarts.

  Reporting and logging of MySQL Cluster start phases also provides more frequent and specific printouts during startup than previously. See Section 7.1, “Summary of MySQL Cluster Start Phases”, for more information.

- **NDB API: new Event API.** MySQL Cluster NDB 7.4.3 introduces an epoch-driven Event API that supercedes the earlier GCI-based model. The new version of the API also simplifies error detection and handling. These changes are realized in the NDB API by implementing a number of new methods for `Ndb` and `NdbEventOperation`, deprecating several other methods of both classes, and adding new type values to `Event::TableEvent`.

  The event handling methods added to `Ndb` in MySQL Cluster NDB 7.4.3 are `pollEvents2()`, `nextEvent2()`, `getHighestQueuedEpoch()`, and `getNextEventOpInEpoch2()`. The `Ndb` methods `pollEvents()`, `nextEvent()`, `getLatestGCI()`, `getGCIEventOperations()`, `isConsistent()`, and `isConsistentGCI()` are deprecated beginning with the same release.

  MySQL Cluster NDB 7.4.3 adds the `NdbEventOperation` event handling methods `getEventType2()`, `getEpoch()`, `isEmptyEpoch()`, and `isErrorEpoch`; it obsoletes `getEventType()`, `getGCI()`, `getLatestGCI()`, `isOverrun()`, `hasError()`, and `clearError()`.

  While some (but not all) of the new methods are direct replacements for deprecated methods, not all of the deprecated methods map to new ones. The Event Class, provides information as to which old methods correspond to new ones.

  Error handling using the new API is no longer handled using dedicated `hasError()` and `clearError()` methods, which are now deprecated (and thus subject to removal in a future release of MySQL Cluster). To support this change, the list of TableEvent types now includes the values TE_EMPTY (empty epoch), TE_INCONSISTENT (inconsistent epoch), and TE_OUT_OF_MEMORY (inconsistent data).

Improvements in event buffer management have also been made by implementing new get_eventbuffer_free_percent(), set_eventbuffer_free_percent(), and get_eventbuffer_memory_usage() methods. Memory buffer usage can now be represented in application code using EventBufferMemoryUsage. The ndb_eventbuffer_free_percent system variable, also implemented in MySQL Cluster NDB 7.4, makes it possible for event buffer memory usage to be checked from MySQL client applications.

For more information, see the detailed descriptions for the Ndb and NdbEventOperation methods listed. See also The Event::TableEvent Type, as well as The EventBufferMemoryUsage Structure.

- **Per-fragment operations information.** In MySQL Cluster NDB 7.4.3 and later, counts of various types of operations on a given fragment or fragment replica can obtained easily using the operations_per_fragment table in the ndbinfo information database. This includes read, write, update, and delete operations, as well as scan and index operations performed by these. Information about operations refused, and about rows scanned and returned from a given fragment replica, is also shown in operations_per_fragment. This table also provides information about interpreted programs used as attribute values, and values returned by them.

MySQL Cluster NDB 7.4 is also supported by MySQL Cluster Manager, which provides an advanced command-line interface that can simplify many complex MySQL Cluster management tasks. See MySQL™ Cluster Manager 1.3.6 User Manual, for more information.

3.5 MySQL Server Using InnoDB Compared with MySQL Cluster

MySQL Server offers a number of choices in storage engines. Since both NDBCLUSTER and InnoDB can serve as transactional MySQL storage engines, users of MySQL Server sometimes become interested in MySQL Cluster. They see NDB as a possible alternative or upgrade to the default InnoDB storage engine in MySQL 5.6. While NDB and InnoDB share common characteristics, there are differences in architecture and implementation, so that some existing MySQL Server applications and usage scenarios can be a good fit for MySQL Cluster, but not all of them.

In this section, we discuss and compare some characteristics of the NDB storage engine used by MySQL Cluster NDB 7.3 with InnoDB used in MySQL 5.6. The next few sections provide a technical comparison. In many instances, decisions about when and where to use MySQL Cluster must be made on a case-by-case basis, taking all factors into consideration. While it is beyond the scope of this documentation to provide specifics for every conceivable usage scenario, we also attempt to offer some very general guidance on the relative suitability of some common types of applications for NDB as opposed to InnoDB backends.

MySQL Cluster NDB 7.3 uses a mysqld based on MySQL 5.6, including support for InnoDB 1.1. While it is possible to use InnoDB tables with MySQL Cluster, such tables are not clustered. It is also not possible to use programs or libraries from a MySQL Cluster NDB 7.3 distribution with MySQL Server 5.6, or the reverse.

While it is also true that some types of common business applications can be run either on MySQL Cluster or on MySQL Server (most likely using the InnoDB storage engine), there are some important architectural and implementation differences. Section 3.5.1, “Differences Between the NDB and InnoDB Storage Engines”, provides a summary of the these differences. Due to the differences, some usage scenarios are clearly more suitable for one engine or the other; see Section 3.5.2, “NDB and InnoDB Workloads”. This in turn has an impact on the types of applications that better suited for use with NDB or InnoDB. See Section 3.5.3, “NDB and InnoDB Feature Usage Summary”, for a comparison of the relative suitability of each for use in common types of database applications.

For information about the relative characteristics of the NDB and MEMORY storage engines, see When to Use MEMORY or MySQL Cluster.

See Alternative Storage Engines, for additional information about MySQL storage engines.

3.5.1 Differences Between the NDB and InnoDB Storage Engines

The MySQL Cluster `NDB` storage engine is implemented using a distributed, shared-nothing architecture, which causes it to behave differently from `InnoDB` in a number of ways. For those unaccustomed to working with `NDB`, unexpected behaviors can arise due to its distributed nature with regard to transactions, foreign keys, table limits, and other characteristics. These are shown in the following table:

Feature	**`InnoDB` 1.1**	**MySQL Cluster `NDB` 7.3, MySQL Cluster NDB 7.4**
MySQL Server Version	5.6	5.6
`InnoDB` Version	`InnoDB` 5.6.29	`InnoDB` 5.6.29
MySQL Cluster Version	N/A	`NDB` 7.3.12
Storage Limits	64TB	3TB (Practical upper limit based on 48 data nodes with 64GB RAM each; can be increased with disk-based data and BLOBs)
Foreign Keys	Yes	Prior to MySQL Cluster NDB 7.3: No. (Ignored, as with `MyISAM`) Available in MySQL Cluster NDB 7.3.
Transactions	All standard types	`READ COMMITTED`
MVCC	Yes	No
Data Compression	Yes	No (MySQL Cluster checkpoint and backup files can be compressed)
Large Row Support (> 14K)	Supported for `VARBINARY`, `VARCHAR`, `BLOB`, and `TEXT` columns	Supported for `BLOB` and `TEXT` columns only (Using these types to store very large amounts of data can lower MySQL Cluster performance)
Replication Support	Asynchronous and semisynchronous replication using MySQL Replication	Automatic synchronous replication within a MySQL Cluster. Asynchronous replication between MySQL Clusters, using MySQL Replication
Scaleout for Read Operations	Yes (MySQL Replication)	Yes (Automatic partitioning in MySQL Cluster; MySQL Cluster Replication)
Scaleout for Write Operations	Requires application-level partitioning (sharding)	Yes (Automatic partitioning in MySQL Cluster is transparent to applications)
High Availability (HA)	Requires additional software	Yes (Designed for 99.999% uptime)

Feature	**InnoDB 1.1**	**MySQL Cluster NDB 7.3, MySQL Cluster NDB 7.4**
Node Failure Recovery and Failover	Requires additional software	Automatic (Key element in MySQL Cluster architecture)
Time for Node Failure Recovery	30 seconds or longer	Typically < 1 second
Real-Time Performance	No	Yes
In-Memory Tables	No	Yes (Some data can optionally be stored on disk; both in-memory and disk data storage are durable)
NoSQL Access to Storage Engine	Native memcached interface in development (see the MySQL Dev Zone article *MySQL Cluster 7.2 (DMR2): NoSQL, Key/Value, Memcached*)	Yes Multiple APIs, including Memcached, Node.js/JavaScript, Java, JPA, C++, and HTTP/REST
Concurrent and Parallel Writes	Not supported	Up to 48 writers, optimized for concurrent writes
Conflict Detection and Resolution (Multiple Replication Masters)	No	Yes
Hash Indexes	No	Yes
Online Addition of Nodes	Read-only replicas using MySQL Replication	Yes (all node types)
Online Upgrades	No	Yes
Online Schema Modifications	Yes, as part of MySQL 5.6.	Yes.

3.5.2 NDB and InnoDB Workloads

MySQL Cluster has a range of unique attributes that make it ideal to serve applications requiring high availability, fast failover, high throughput, and low latency. Due to its distributed architecture and multi-node implementation, MySQL Cluster also has specific constraints that may keep some workloads from performing well. A number of major differences in behavior between the `NDB` and `InnoDB` storage engines with regard to some common types of database-driven application workloads are shown in the following table::

Workload	**InnoDB**	**MySQL Cluster (NDB)**
High-Volume OLTP Applications	Yes	Yes
DSS Applications (data marts, analytics)	Yes	Limited (Join operations across OLTP datasets not exceeding 3TB in size)
Custom Applications	Yes	Yes
Packaged Applications	Yes	Limited (should be mostly primary key access). MySQL Cluster NDB 7.3 supports foreign keys.
In-Network Telecoms Applications (HLR, HSS, SDP)	No	Yes

Workload	InnoDB	MySQL Cluster (NDB)
Session Management and Caching	Yes	Yes
E-Commerce Applications	Yes	Yes
User Profile Management, AAA Protocol	Yes	Yes

3.5.3 NDB and InnoDB Feature Usage Summary

When comparing application feature requirements to the capabilities of InnoDB with NDB, some are clearly more compatible with one storage engine than the other.

The following table lists supported application features according to the storage engine to which each feature is typically better suited.

Preferred application requirements for InnoDB	Preferred application requirements for NDB
• Foreign keys **Note** MySQL Cluster NDB 7.3 supports foreign keys. • Full table scans • Very large databases, rows, or transactions • Transactions other than READ COMMITTED	• Write scaling • 99.999% uptime • Online addition of nodes and online schema operations • Multiple SQL and NoSQL APIs (see MySQL Cluster APIs: Overview and Concepts) • Real-time performance • Limited use of BLOB columns • Foreign keys are supported, although their use may have an impact on performance at high throughput

3.6 Known Limitations of MySQL Cluster

In the sections that follow, we discuss known limitations in current releases of MySQL Cluster as compared with the features available when using the MyISAM and InnoDB storage engines. If you check the “Cluster” category in the MySQL bugs database at http://bugs.mysql.com, you can find known bugs in the following categories under “MySQL Server:” in the MySQL bugs database at http://bugs.mysql.com, which we intend to correct in upcoming releases of MySQL Cluster:

- MySQL Cluster
- Cluster Direct API (NDBAPI)
- Cluster Disk Data
- Cluster Replication
- ClusterJ

This information is intended to be complete with respect to the conditions just set forth. You can report any discrepancies that you encounter to the MySQL bugs database using the instructions given in How to Report Bugs or Problems. If we do not plan to fix the problem in MySQL Cluster NDB 7.3, we will add it to the list.

See Section 3.6.11, “Previous MySQL Cluster Issues Resolved in MySQL Cluster NDB 7.3” for a list of issues in MySQL Cluster NDB 7.2 that have been resolved in MySQL Cluster NDB 7.3..

Note

Limitations and other issues specific to MySQL Cluster Replication are described in Section 8.3, “Known Issues in MySQL Cluster Replication”.

3.6.1 Noncompliance with SQL Syntax in MySQL Cluster

Some SQL statements relating to certain MySQL features produce errors when used with `NDB` tables, as described in the following list:

- **Temporary tables.** Temporary tables are not supported. Trying either to create a temporary table that uses the `NDB` storage engine or to alter an existing temporary table to use `NDB` fails with the error `Table storage engine 'ndbcluster' does not support the create option 'TEMPORARY'`.
- **Indexes and keys in NDB tables.** Keys and indexes on MySQL Cluster tables are subject to the following limitations:
  - **Column width.** Attempting to create an index on an `NDB` table column whose width is greater than 3072 bytes succeeds, but only the first 3072 bytes are actually used for the index. In such cases, a warning `Specified key was too long; max key length is 3072 bytes` is issued, and a `SHOW CREATE TABLE` statement shows the length of the index as 3072.
  - **TEXT and BLOB columns.** You cannot create indexes on `NDB` table columns that use any of the `TEXT` or `BLOB` data types.
  - **FULLTEXT indexes.** The `NDB` storage engine does not support `FULLTEXT` indexes, which are possible for `MyISAM` and (MySQL 5.6.4 and later) `InnoDB` tables only.

    However, you can create indexes on `VARCHAR` columns of `NDB` tables.
  - **USING HASH keys and NULL.** Using nullable columns in unique keys and primary keys means that queries using these columns are handled as full table scans. To work around this issue, make the column `NOT NULL`, or re-create the index without the `USING HASH` option.
  - **Prefixes.** There are no prefix indexes; only entire columns can be indexed. (The size of an `NDB` column index is always the same as the width of the column in bytes, up to and including 3072 bytes, as described earlier in this section. Also see Section 3.6.6, “Unsupported or Missing Features in MySQL Cluster”, for additional information.)
  - **BIT columns.** A `BIT` column cannot be a primary key, unique key, or index, nor can it be part of a composite primary key, unique key, or index.
  - **AUTO_INCREMENT columns.** Like other MySQL storage engines, the `NDB` storage engine can handle a maximum of one `AUTO_INCREMENT` column per table, and this column must be indexed. However, in the case of a MySQL Cluster table with no explicit primary key, an `AUTO_INCREMENT` column is automatically defined and used as a “hidden” primary key. For this reason, you cannot create an `NDB` table having an `AUTO_INCREMENT` column and no explicit primary key.
- **Restrictions on foreign keys.** Support for foreign key constraints in MySQL Cluster NDB 7.3 is comparable to that provided by `InnoDB`, subject to the following restrictions:
  - Every column referenced as a foreign key requires an explicit unique key, if it is not the table's primary key.
  - `ON UPDATE CASCADE` is not supported when the reference is to the parent table's primary key.
  - `SET DEFAULT` is not supported. (Also not supported by `InnoDB`.)

- The `NO ACTION` keywords are accepted but treated as `RESCRICT`. (Also the same as with `InnoDB`.)
- Prior to MySQL Cluster NDB 7.3.5, when creating a table with foreign key referencing an index in another table, it sometimes appeared possible to create the foreign key even if the order of the columns in the indexes did not match, due to the fact that an appropriate error was not always returned internally. A partial fix for this issue in MySQL Cluster NDB 7.3.5 improves the error used internally to work in most cases; however, it is still possible for this situation to occur in the event that the parent index is a unique index. (Bug #18094360)

For more information, see Using FOREIGN KEY Constraints, and FOREIGN KEY Constraints.

- **MySQL Cluster and geometry data types.** Geometry data types (`WKT` and `WKB`) are supported for `NDB` tables. However, spatial indexes are not supported.
- **Character sets and binary log files.** Currently, the `ndb_apply_status` and `ndb_binlog_index` tables are created using the `latin1` (ASCII) character set. Because names of binary logs are recorded in this table, binary log files named using non-Latin characters are not referenced correctly in these tables. This is a known issue, which we are working to fix. (Bug #50226)

  To work around this problem, use only Latin-1 characters when naming binary log files or setting any the `--basedir`, `--log-bin`, or `--log-bin-index` options.
- **Creating NDB tables with user-defined partitioning.** Support for user-defined partitioning in MySQL Cluster is restricted to [`LINEAR`] `KEY` partitioning. Using any other partitioning type with `ENGINE=NDB` or `ENGINE=NDBCLUSTER` in a `CREATE TABLE` statement results in an error.

  It is possible to override this restriction, but doing so is not supported for use in production settings. For details, see User-defined partitioning and the NDB storage engine (MySQL Cluster).

  Default partitioning scheme. All MySQL Cluster tables are by default partitioned by `KEY` using the table's primary key as the partitioning key. If no primary key is explicitly set for the table, the "hidden" primary key automatically created by the `NDB` storage engine is used instead. For additional discussion of these and related issues, see KEY Partitioning.

  `CREATE TABLE` and `ALTER TABLE` statements that would cause a user-partitioned `NDBCLUSTER` table not to meet either or both of the following two requirements are not permitted, and fail with an error:

  1. The table must have an explicit primary key.
  2. All columns listed in the table's partitioning expression must be part of the primary key.

  Exception. If a user-partitioned `NDBCLUSTER` table is created using an empty column-list (that is, using `PARTITION BY [LINEAR] KEY()`), then no explicit primary key is required.

  Maximum number of partitions for NDBCLUSTER tables. The maximum number of partitions that can defined for a `NDBCLUSTER` table when employing user-defined partitioning is 8 per node group. (See Section 3.2, "MySQL Cluster Nodes, Node Groups, Replicas, and Partitions", for more information about MySQL Cluster node groups.

  DROP PARTITION not supported. It is not possible to drop partitions from `NDB` tables using `ALTER TABLE ... DROP PARTITION`. The other partitioning extensions to `ALTER TABLE`—`ADD PARTITION`, `REORGANIZE PARTITION`, and `COALESCE PARTITION`—are supported for Cluster tables, but use copying and so are not optimized. See Management of RANGE and LIST Partitions and ALTER TABLE Syntax.
- **Row-based replication.**

When using row-based replication with MySQL Cluster, binary logging cannot be disabled. That is, the `NDB` storage engine ignores the value of `sql_log_bin`. (Bug #16680)

3.6.2 Limits and Differences of MySQL Cluster from Standard MySQL Limits

In this section, we list limits found in MySQL Cluster that either differ from limits found in, or that are not found in, standard MySQL.

Memory usage and recovery. Memory consumed when data is inserted into an `NDB` table is not automatically recovered when deleted, as it is with other storage engines. Instead, the following rules hold true:

- A `DELETE` statement on an `NDB` table makes the memory formerly used by the deleted rows available for re-use by inserts on the same table only. However, this memory can be made available for general re-use by performing `OPTIMIZE TABLE`.

  A rolling restart of the cluster also frees any memory used by deleted rows. See Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”.

- A `DROP TABLE` or `TRUNCATE TABLE` operation on an `NDB` table frees the memory that was used by this table for re-use by any `NDB` table, either by the same table or by another `NDB` table.

  > **Note**
  >
  > Recall that `TRUNCATE TABLE` drops and re-creates the table. See TRUNCATE TABLE Syntax.

- **Limits imposed by the cluster's configuration.**
  A number of hard limits exist which are configurable, but available main memory in the cluster sets limits. See the complete list of configuration parameters in Section 5.3, “MySQL Cluster Configuration Files”. Most configuration parameters can be upgraded online. These hard limits include:

  - Database memory size and index memory size (`DataMemory` and `IndexMemory`, respectively).

    `DataMemory` is allocated as 32KB pages. As each `DataMemory` page is used, it is assigned to a specific table; once allocated, this memory cannot be freed except by dropping the table.

    See Section 5.3.6, “Defining MySQL Cluster Data Nodes”, for more information.

  - The maximum number of operations that can be performed per transaction is set using the configuration parameters `MaxNoOfConcurrentOperations` and `MaxNoOfLocalOperations`.

    > **Note**
    >
    > Bulk loading, `TRUNCATE TABLE`, and `ALTER TABLE` are handled as special cases by running multiple transactions, and so are not subject to this limitation.

  - Different limits related to tables and indexes. For example, the maximum number of ordered indexes in the cluster is determined by `MaxNoOfOrderedIndexes`, and the maximum number of ordered indexes per table is 16.

- **Node and data object maximums.** The following limits apply to numbers of cluster nodes and metadata objects:

  - The maximum number of data nodes is 48.

    A data node must have a node ID in the range of 1 to 48, inclusive. (Management and API nodes may use node IDs in the range 1 to 255, inclusive.)

- The total maximum number of nodes in a MySQL Cluster is 255. This number includes all SQL nodes (MySQL Servers), API nodes (applications accessing the cluster other than MySQL servers), data nodes, and management servers.
- The maximum number of metadata objects in current versions of MySQL Cluster is 20320. This limit is hard-coded.

See Section 3.6.11, “Previous MySQL Cluster Issues Resolved in MySQL Cluster NDB 7.3”, for more information.

3.6.3 Limits Relating to Transaction Handling in MySQL Cluster

A number of limitations exist in MySQL Cluster with regard to the handling of transactions. These include the following:

- **Transaction isolation level.** The `NDBCLUSTER` storage engine supports only the `READ COMMITTED` transaction isolation level. (`InnoDB`, for example, supports `READ COMMITTED`, `READ UNCOMMITTED`, `REPEATABLE READ`, and `SERIALIZABLE`.) See Section 7.3.4, “MySQL Cluster Backup Troubleshooting”, for information on how this can affect backing up and restoring Cluster databases.)
- **Transactions and BLOB or TEXT columns.** `NDBCLUSTER` stores only part of a column value that uses any of MySQL's `BLOB` or `TEXT` data types in the table visible to MySQL; the remainder of the `BLOB` or `TEXT` is stored in a separate internal table that is not accessible to MySQL. This gives rise to two related issues of which you should be aware whenever executing `SELECT` statements on tables that contain columns of these types:

  1. For any `SELECT` from a MySQL Cluster table: If the `SELECT` includes a `BLOB` or `TEXT` column, the `READ COMMITTED` transaction isolation level is converted to a read with read lock. This is done to guarantee consistency.
  2. For any `SELECT` which uses a unique key lookup to retrieve any columns that use any of the `BLOB` or `TEXT` data types and that is executed within a transaction, a shared read lock is held on the table for the duration of the transaction—that is, until the transaction is either committed or aborted.

     This issue does not occur for queries that use index or table scans, even against `NDB` tables having `BLOB` or `TEXT` columns.

     For example, consider the table `t` defined by the following `CREATE TABLE` statement:

     ```
     CREATE TABLE t (
         a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
         b INT NOT NULL,
         c INT NOT NULL,
         d TEXT,
         INDEX i(b),
         UNIQUE KEY u(c)
     ) ENGINE = NDB,
     ```

     Either of the following queries on `t` causes a shared read lock, because the first query uses a primary key lookup and the second uses a unique key lookup:

     ```
     SELECT * FROM t WHERE a = 1;
     SELECT * FROM t WHERE c = 1;
     ```

     However, none of the four queries shown here causes a shared read lock:

     ```
     SELECT * FROM t WHERE b = 1;
     ```

```
SELECT * FROM t WHERE d = '1';
SELECT * FROM t;
SELECT b,c WHERE a = 1;
```

This is because, of these four queries, the first uses an index scan, the second and third use table scans, and the fourth, while using a primary key lookup, does not retrieve the value of any `BLOB` or `TEXT` columns.

You can help minimize issues with shared read locks by avoiding queries that use unique key lookups that retrieve `BLOB` or `TEXT` columns, or, in cases where such queries are not avoidable, by committing transactions as soon as possible afterward.

- **Rollbacks.** There are no partial transactions, and no partial rollbacks of transactions. A duplicate key or similar error causes the entire transaction to be rolled back.

  This behavior differs from that of other transactional storage engines such as `InnoDB` that may roll back individual statements.

- **Transactions and memory usage.**
  As noted elsewhere in this chapter, MySQL Cluster does not handle large transactions well; it is better to perform a number of small transactions with a few operations each than to attempt a single large transaction containing a great many operations. Among other considerations, large transactions require very large amounts of memory. Because of this, the transactional behavior of a number of MySQL statements is effected as described in the following list:

  - `TRUNCATE TABLE` is not transactional when used on `NDB` tables. If a `TRUNCATE TABLE` fails to empty the table, then it must be re-run until it is successful.

  - `DELETE FROM` (even with no `WHERE` clause) *is* transactional. For tables containing a great many rows, you may find that performance is improved by using several `DELETE FROM ... LIMIT ...` statements to “chunk” the delete operation. If your objective is to empty the table, then you may wish to use `TRUNCATE TABLE` instead.

  - **LOAD DATA statements.** `LOAD DATA INFILE` is not transactional when used on `NDB` tables.

    Important

    When executing a `LOAD DATA INFILE` statement, the `NDB` engine performs commits at irregular intervals that enable better utilization of the communication network. It is not possible to know ahead of time when such commits take place.

  - **ALTER TABLE and transactions.** When copying an `NDB` table as part of an `ALTER TABLE`, the creation of the copy is nontransactional. (In any case, this operation is rolled back when the copy is deleted.)

- **Transactions and the COUNT() function.** When using MySQL Cluster Replication, it is not possible to guarantee the transactional consistency of the `COUNT()` function on the slave. In other words, when performing on the master a series of statements (`INSERT`, `DELETE`, or both) that changes the number of rows in a table within a single transaction, executing `SELECT COUNT(*) FROM *table*` queries on the slave may yield intermediate results. This is due to the fact that `SELECT COUNT(...)` may perform dirty reads, and is not a bug in the `NDB` storage engine. (See Bug #31321 for more information.)

3.6.4 MySQL Cluster Error Handling

Starting, stopping, or restarting a node may give rise to temporary errors causing some transactions to fail. These include the following cases:

- **Temporary errors.** When first starting a node, it is possible that you may see Error 1204 `Temporary failure, distribution changed` and similar temporary errors.

- **Errors due to node failure.** The stopping or failure of any data node can result in a number of different node failure errors. (However, there should be no aborted transactions when performing a planned shutdown of the cluster.)

In either of these cases, any errors that are generated must be handled within the application. This should be done by retrying the transaction.

See also Section 3.6.2, “Limits and Differences of MySQL Cluster from Standard MySQL Limits”.

3.6.5 Limits Associated with Database Objects in MySQL Cluster

Some database objects such as tables and indexes have different limitations when using the `NDBCLUSTER` storage engine:

- **Database and table names.** When using the `NDB` storage engine, the maximum allowed length both for database names and for table names is 63 characters.

  In MySQL Cluster NDB 7.3.8 and later, a statement using a database name or table name longer than this limit fails with an appropriate error. (Bug #19550973)

- **Number of database objects.** The maximum number of *all* `NDB` database objects in a single MySQL Cluster—including databases, tables, and indexes—is limited to 20320.

- **Attributes per table.** The maximum number of attributes (that is, columns and indexes) that can belong to a given table is 512.

- **Attributes per key.** The maximum number of attributes per key is 32.

- **Row size.** The maximum permitted size of any one row is 14000 bytes. Each `BLOB` or `TEXT` column contributes 256 + 8 = 264 bytes to this total.

- **BIT column storage per table.** The maximum combined width for all `BIT` columns used in a given `NDB` table is 4096.

3.6.6 Unsupported or Missing Features in MySQL Cluster

A number of features supported by other storage engines are not supported for `NDB` tables. Trying to use any of these features in MySQL Cluster does not cause errors in or of itself; however, errors may occur in applications that expects the features to be supported or enforced. Statements referencing such features, even if effectively ignored by `NDB`, must be syntactically and otherwise valid.

- **Index prefixes.** Prefixes on indexes are not supported for `NDB` tables. If a prefix is used as part of an index specification in a statement such as `CREATE TABLE`, `ALTER TABLE`, or `CREATE INDEX`, the prefix is not created by `NDB`.

  A statement containing an index prefix, and creating or modifying an `NDB` table, must still be syntactically valid. For example, the following statement always fails with Error 1089 `Incorrect prefix key; the used key part isn't a string, the used length is longer than the key part, or the storage engine doesn't support unique prefix keys`, regardless of storage engine:

  ```
  CREATE TABLE t1 (
      c1 INT NOT NULL,
      c2 VARCHAR(100),
      INDEX i1 (c2(500))
  );
  ```

  This happens on account of the SQL syntax rule that no index may have a prefix larger than itself.

- **Savepoints and rollbacks.** Savepoints and rollbacks to savepoints are ignored as in `MyISAM`.

- **Durability of commits.** There are no durable commits on disk. Commits are replicated, but there is no guarantee that logs are flushed to disk on commit.

- **Replication.** Statement-based replication is not supported. Use `--binlog-format=ROW` (or `--binlog-format=MIXED`) when setting up cluster replication. See Chapter 8, *MySQL Cluster Replication*, for more information.

  Replication using global transaction identifiers (GTIDs) is not compatible with MySQL Cluster, and is not supported in MySQL Cluster NDB 7.3 or MySQL Cluster NDB 7.4. Do not enable GTIDs when using the `NDB` storage engine, as this is very likely to cause problems including failure of MySQL Cluster Replication.

  > **Note**
  >
  > See Section 3.6.3, “Limits Relating to Transaction Handling in MySQL Cluster”, for more information relating to limitations on transaction handling in `NDB`.

3.6.7 Limitations Relating to Performance in MySQL Cluster

The following performance issues are specific to or especially pronounced in MySQL Cluster:

- **Range scans.** There are query performance issues due to sequential access to the `NDB` storage engine; it is also relatively more expensive to do many range scans than it is with either `MyISAM` or `InnoDB`.

- **Reliability of Records in range.** The `Records in range` statistic is available but is not completely tested or officially supported. This may result in nonoptimal query plans in some cases. If necessary, you can employ `USE INDEX` or `FORCE INDEX` to alter the execution plan. See Index Hints, for more information on how to do this.

- **Unique hash indexes.** Unique hash indexes created with `USING HASH` cannot be used for accessing a table if `NULL` is given as part of the key.

3.6.8 Issues Exclusive to MySQL Cluster

The following are limitations specific to the `NDB` storage engine:

- **Machine architecture.** All machines used in the cluster must have the same architecture. That is, all machines hosting nodes must be either big-endian or little-endian, and you cannot use a mixture of both. For example, you cannot have a management node running on a PowerPC which directs a data node that is running on an x86 machine. This restriction does not apply to machines simply running `mysql` or other clients that may be accessing the cluster's SQL nodes.

- **Binary logging.**
  MySQL Cluster has the following limitations or restrictions with regard to binary logging:

  - `sql_log_bin` has no effect on data operations; however, it is supported for schema operations.

  - MySQL Cluster cannot produce a binary log for tables having `BLOB` columns but no primary key.

  - Only the following schema operations are logged in a cluster binary log which is *not* on the `mysqld` executing the statement:

    - `CREATE TABLE`

    - `ALTER TABLE`

    - `DROP TABLE`

    - `CREATE DATABASE` / `CREATE SCHEMA`

    - `DROP DATABASE` / `DROP SCHEMA`

- `CREATE TABLESPACE`
- `ALTER TABLESPACE`
- `DROP TABLESPACE`
- `CREATE LOGFILE GROUP`
- `ALTER LOGFILE GROUP`
- `DROP LOGFILE GROUP`

See also Section 3.6.10, “Limitations Relating to Multiple MySQL Cluster Nodes”.

3.6.9 Limitations Relating to MySQL Cluster Disk Data Storage

Disk Data object maximums and minimums. Disk data objects are subject to the following maximums and minimums:

- Maximum number of tablespaces: 2^{32} (4294967296)
- Maximum number of data files per tablespace: 2^{16} (65536)
- Maximum data file size: The theoretical limit is 64G; however, the practical upper limit is 32G. This is equivalent to 32768 extents of 1M each.

  Since a MySQL Cluster Disk Data table can use at most 1 tablespace, this means that the theoretical upper limit to the amount of data (in bytes) that can be stored on disk by a single `NDB` table is 32G * 65536 = 2251799813685248, or approximately 2 petabytes.

- The theoretical maximum number of extents per tablespace data file is 2^{16} (65536); however, for practical purposes, the recommended maximum number of extents per data file is 2^{15} (32768).

  The minimum and maximum possible sizes of extents for tablespace data files are 32K and 2G, respectively. See CREATE TABLESPACE Syntax, for more information.

Disk Data tables and diskless mode. Use of Disk Data tables is not supported when running the cluster in diskless mode.

3.6.10 Limitations Relating to Multiple MySQL Cluster Nodes

Multiple SQL nodes.
The following are issues relating to the use of multiple MySQL servers as MySQL Cluster SQL nodes, and are specific to the `NDBCLUSTER` storage engine:

- **No distributed table locks.** A `LOCK TABLES` works only for the SQL node on which the lock is issued; no other SQL node in the cluster “sees” this lock. This is also true for a lock issued by any statement that locks tables as part of its operations. (See next item for an example.)
- **ALTER TABLE operations.** `ALTER TABLE` is not fully locking when running multiple MySQL servers (SQL nodes). (As discussed in the previous item, MySQL Cluster does not support distributed table locks.)

Multiple management nodes.
When using multiple management servers:

- If any of the management servers are running on the same host, you must give nodes explicit IDs in connection strings because automatic allocation of node IDs does not work across multiple management servers on the same host. This is not required if every management server resides on a different host.

- When a management server starts, it first checks for any other management server in the same MySQL Cluster, and upon successful connection to the other management server uses its configuration data. This means that the management server `--reload` and `--initial` startup options are ignored unless the management server is the only one running. It also means that, when performing a rolling restart of a MySQL Cluster with multiple management nodes, the management server reads its own configuration file if (and only if) it is the only management server running in this MySQL Cluster. See Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”, for more information.

Multiple network addresses. Multiple network addresses per data node are not supported. Use of these is liable to cause problems: In the event of a data node failure, an SQL node waits for confirmation that the data node went down but never receives it because another route to that data node remains open. This can effectively make the cluster inoperable.

> **Note**
>
> It is possible to use multiple network hardware *interfaces* (such as Ethernet cards) for a single data node, but these must be bound to the same address. This also means that it not possible to use more than one `[tcp]` section per connection in the `config.ini` file. See Section 5.3.9, “MySQL Cluster TCP/IP Connections”, for more information.

3.6.11 Previous MySQL Cluster Issues Resolved in MySQL Cluster NDB 7.3

A number of limitations and related issues that existed in earlier versions of MySQL Cluster have been resolved in MySQL Cluster NDB 7.3. These are described briefly in the following list:

- **Support for foreign keys.** Foreign key constraints are now supported for `NDB` tables, similar to how these are supported by the `InnoDB` storage engine.

  > **Note**
  >
  > Unlike the case with user-partitioned `InnoDB` tables, foreign keys are supported for `NDB` tables that are partitioned by `KEY` or `LINEAR KEY`.

  FOREIGN KEY Constraints, provides more information about foreign key support in MySQL. For more information about the syntax supported by MySQL for foreign keys, see Using FOREIGN KEY Constraints.

Chapter 4 MySQL Cluster Installation

Table of Contents

This section describes the basics for planning, installing, configuring, and running a MySQL Cluster. Whereas the examples in Chapter 5, *Configuration of MySQL Cluster* provide more in-depth information on a variety of clustering options and configuration, the result of following the guidelines and procedures outlined here should be a usable MySQL Cluster which meets the *minimum* requirements for availability and safeguarding of data.

For information about upgrading or downgrading a MySQL Cluster between release versions, see Section 4.8, "Upgrading and Downgrading MySQL Cluster".

This section covers hardware and software requirements; networking issues; installation of MySQL Cluster; basic configuration issues; starting, stopping, and restarting the cluster; loading of a sample database; and performing queries.

MySQL Cluster NDB 7.3 and later provides a MySQL Cluster Auto-Installer, a web-based graphical installer, as part of the MySQL Cluster distribution. The Auto-Installer can be used to perform basic installation and setup of a MySQL Cluster on one (for testing) or more host computers. See Section 4.1, "The MySQL Cluster Auto-Installer", for more information.

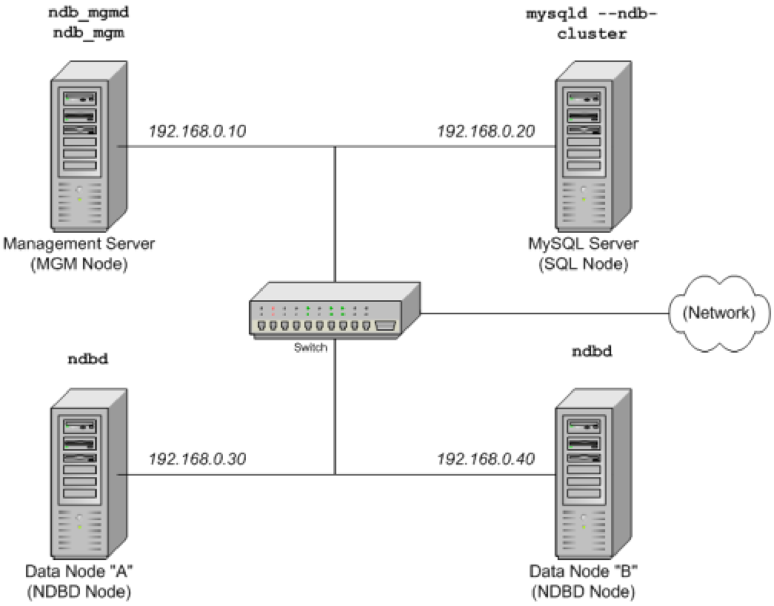
Assumptions. The following sections make a number of assumptions regarding the cluster's physical and network configuration. These assumptions are discussed in the next few paragraphs.

Cluster nodes and host computers. The cluster consists of four nodes, each on a separate host computer, and each with a fixed network address on a typical Ethernet network as shown here:

Node	IP Address
Management node (`mgmd`)	192.168.0.10
SQL node (`mysqld`)	192.168.0.20
Data node "A" (`ndbd`)	192.168.0.30
Data node "B" (`ndbd`)	192.168.0.40

This may be made clearer by the following diagram:

Figure 4.1 MySQL Cluster Multi-Computer Setup



Network addressing. In the interest of simplicity (and reliability), this *How-To* uses only numeric IP addresses. However, if DNS resolution is available on your network, it is possible to use host names in lieu of IP addresses in configuring Cluster. Alternatively, you can use the `hosts` file (typically `/etc/hosts` for Linux and other Unix-like operating systems, `C:\WINDOWS\system32\drivers\etc\hosts` on Windows, or your operating system's equivalent) for providing a means to do host lookup if such is available.

Potential hosts file issues. A common problem when trying to use host names for Cluster nodes arises because of the way in which some operating systems (including some Linux distributions) set up the system's own host name in the `/etc/hosts` during installation. Consider two machines with the host names `ndb1` and `ndb2`, both in the `cluster` network domain. Red Hat Linux (including some derivatives such as CentOS and Fedora) places the following entries in these machines' `/etc/hosts` files:

```
#  ndb1 /etc/hosts:
127.0.0.1   ndb1.cluster ndb1 localhost.localdomain localhost
```

```
#  ndb2 /etc/hosts:
127.0.0.1   ndb2.cluster ndb2 localhost.localdomain localhost
```

SUSE Linux (including OpenSUSE) places these entries in the machines' `/etc/hosts` files:

```
#  ndb1 /etc/hosts:
127.0.0.1       localhost
127.0.0.2       ndb1.cluster ndb1
```

```
#  ndb2 /etc/hosts:
127.0.0.1       localhost
127.0.0.2       ndb2.cluster ndb2
```

In both instances, `ndb1` routes `ndb1.cluster` to a loopback IP address, but gets a public IP address from DNS for `ndb2.cluster`, while `ndb2` routes `ndb2.cluster` to a loopback address and obtains a public address for `ndb1.cluster`. The result is that each data node connects to the management server, but cannot tell when any other data nodes have connected, and so the data nodes appear to hang while starting.

> **Caution**
>
> You cannot mix `localhost` and other host names or IP addresses in `config.ini`. For these reasons, the solution in such cases (other than to use IP addresses for *all* `config.ini` `HostName` entries) is to remove the fully qualified host names from `/etc/hosts` and use these in `config.ini` for all cluster hosts.

Host computer type. Each host computer in our installation scenario is an Intel-based desktop PC running a supported operating system installed to disk in a standard configuration, and running no unnecessary services. The core operating system with standard TCP/IP networking capabilities should be sufficient. Also for the sake of simplicity, we also assume that the file systems on all hosts are set up identically. In the event that they are not, you should adapt these instructions accordingly.

Network hardware. Standard 100 Mbps or 1 gigabit Ethernet cards are installed on each machine, along with the proper drivers for the cards, and that all four hosts are connected through a standard-issue Ethernet networking appliance such as a switch. (All machines should use network cards with the same throughput. That is, all four machines in the cluster should have 100 Mbps cards *or* all four machines should have 1 Gbps cards.) MySQL Cluster works in a 100 Mbps network; however, gigabit Ethernet provides better performance.

> **Important**
>
> MySQL Cluster is *not* intended for use in a network for which throughput is less than 100 Mbps or which experiences a high degree of latency. For this reason (among others), attempting to run a MySQL Cluster over a wide area network such as the Internet is not likely to be successful, and is not supported in production.

Sample data. We use the `world` database which is available for download from the MySQL Web site (see http://dev.mysql.com/doc/index-other.html). We assume that each machine has sufficient memory for running the operating system, required MySQL Cluster processes, and (on the data nodes) storing the database.

For general information about installing MySQL, see Installing and Upgrading MySQL. For information about installation of MySQL Cluster on Linux and other Unix-like operating systems, see Section 4.2, "Installation of MySQL Cluster on Linux". For information about installation of MySQL Cluster on Windows operating systems, see Section 4.3, "Installing MySQL Cluster on Windows".

For general information about MySQL Cluster hardware, software, and networking requirements, see Section 3.3, "MySQL Cluster Hardware, Software, and Networking Requirements".

4.1 The MySQL Cluster Auto-Installer

This section describes the web-based graphical configuration installer included as part of the MySQL Cluster distribution beginning with MySQL Cluster NDB 7.3.1. Topics discussed include an overview of the installer and its parts, software and other requirements for running the installer, navigating the GUI, and using the installer to set up and start or stop a MySQL Cluster on one or more host computers.

4.1.1 MySQL Cluster Auto-Installer Requirements

This section provides information on supported operating platforms and software, required software, and other prerequisites for running the MySQL Cluster Auto-Installer.

Supported platforms. The MySQL Cluster Auto-Installer is available with most MySQL Cluster NDB 7.3 and later distributions for recent versions of Linux, Windows, Solaris, and MacOS X. For more detailed information about platform support for MySQL Cluster and the MySQL Cluster Auto-Installer, see http://www.mysql.com/support/supportedplatforms/cluster.html.

Supported Web browsers. The Web-based installer is supported with recent versions of Firefox and Microsoft Internet Explorer. It should also work with recent versions of Opera, Safari, and Chrome, although we have not thoroughly tested for compability with these browsers.

Required software—setup host. The following software must be installed on the host where the Auto-Installer is run:

- **Python 2.6 or newer.** The Auto-Installer requires the Python interpreter and standard libraries. If these are not already installed on the system, you may be able to add them using the system's package manager. Otherwise, they can be downloaded from http://python.org/download/.
- **Paramiko 1.7.7.1 or newer.** This is required to communicate with remote hosts using SSH. You can download it from http://www.lag.net/paramiko/. Paramiko may also be available from your system's package manager.
- **Pycrypto version 2.6 or newer.** This cryptography module is required by Paramiko. If it is not available using your system's package manage, you can download it from https://www.dlitz.net/software/pycrypto/.

All of the software in the preceding list is included in the Windows version of the configuration tool, and does not need to be installed separately.

The Paramiko and Pycrypto libraries are required only if you intend to deploy MySQL Cluster nodes on remote hosts, and are not needed if all nodes are on the same host where the installer is run.

Required software—remote hosts. The only software required for remote hosts where you wish to deploy MySQL Cluster nodes is the SSH server, which is usually installed by default on Linux and Solaris systems. Several alternatives are available for Windows; for an overview of these, see http://en.wikipedia.org/wiki/Comparison_of_SSH_servers.

An additional requirement when using multiple hosts is that it is possible to authenticate to any of the remote hosts using SSH and the proper keys or user credentials, as discussed in the next few paragraphs:

Authentication and security. Three basic security or authentication mechanisms for remote access are available to the Auto-Installer, which we list and describe here:

- **SSH.** A secure shell connection is used to enable the back end to perform actions on remote hosts. For this reason, an SSH server must be running on the remote host. In addition, the system user running the installer must have access to the remote server, either with a user name and password, or by using public and private keys.

  > **Important**
  >
  > You should never use the system `root` account for remote access, as this is extremely insecure. In addition, `mysqld` cannot normally be started by system `root`. For these and other reasons, you should provide SSH credentials for a regular user account on the target system, and not for system `root`. For more information about this issue, see How to Run MySQL as a Normal User.

- **HTTPS.** Remote communication between the Web browser front end and the back end is not encrypted by default, which means that information such as the user's SSH password is transmitted in clear text that is readable to anyone. For communication from a remote client to be encrypted, the back end must have a certificate, and the front end must communicate with the back end using HTTPS rather than HTTP. Enabling HTTPS is accomplished most easily through issuing a self-signed certificate. Once the certificate is issued, you must make sure that it is used. You can do this by starting `ndb_setup.py` from the command line with the `--use-https` and `--cert-file` options.

- **Certificate-based authentication.** The back end `ndb_setup.py` process can execute commands on the local host as well as remote hosts. This means that anyone connecting to the back end can take charge of how commands are executed. To reject unwanted connections to the back end, a certificate may be required for authentication of the client. In this case, a certificate must be issued by the user, installed in the browser, and made available to the back end for authentication purposes. You can enact this requirement (together with or in place of password or key authentication) by starting `ndb_setup.py` with the `--ca-certs-file` option.

There is no need or requirement for secure authentication when the client browser is running on the same host as the Auto-Installer back end.

See also Section 7.11, “MySQL Cluster Security Issues”, which discusses security considerations to take into account when deploying MySQL Cluster, as well as Security, for more general MySQL security information.

4.1.2 MySQL Cluster Auto-Installer Overview

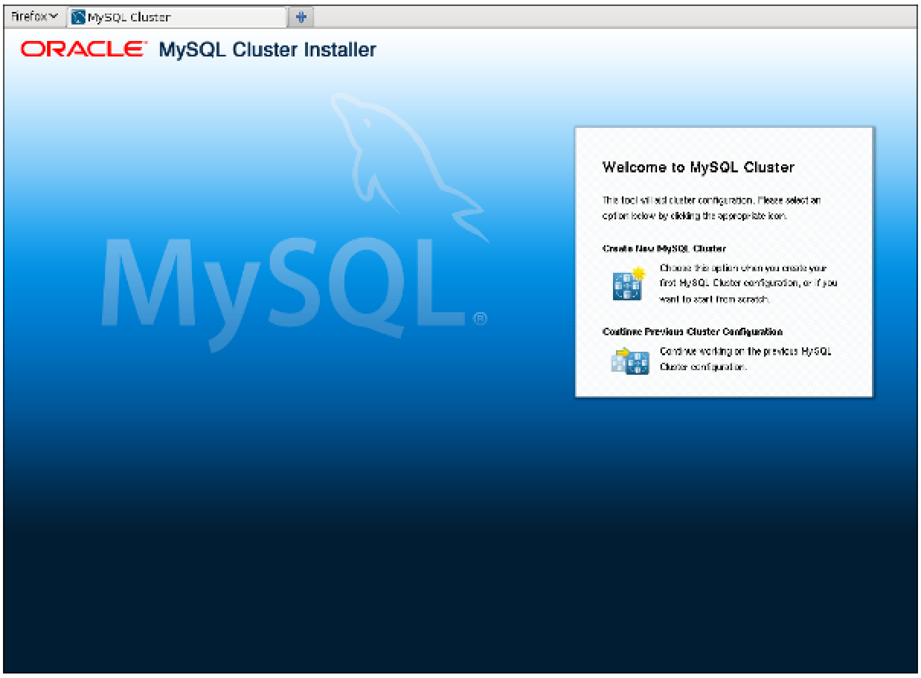
The MySQL Cluster Auto-Installer is made up of two components. The front end is a GUI client implemented as a Web page that loads and runs in a standard Web browser such as Firefox or Microsoft Internet Explorer (see Section 4.1.1, “MySQL Cluster Auto-Installer Requirements”). The back end is a server process (`ndb_setup.py`) that runs on the local machine or on another host to which you have access.

These two components (client and server) communicate with each other using standard HTTP requests and responses. The back end can manage MySQL Cluster software programs on any host where the back end user has granted access. If the MySQL Cluster software is on a different host, the back end relies on SSH for access, using the Paramiko library for executing commands remotely (see Section 4.1.1, “MySQL Cluster Auto-Installer Requirements”).

The remainder of this section is concerned primarily with the Web client. For more information about using the command-line tool, see Section 6.23, “`ndb_setup.py` — Start browser-based Auto-Installer for MySQL Cluster”.

MySQL Cluster Auto-Installer Interface. This section describes the layout and navigation of the MySQL Cluster Auto-Installer, whose Welcome screen looks similar to what is shown here when it is first opened in the Web browser:

Figure 4.2 Welcome Screen For MySQL Cluster Auto-Installer



You can access the installer UI by selecting either of the options **Create New MySQL Cluster** or **Continue Previous Cluster Configuration**. A typical screen in the Auto-Installer includes the following elements:

1. **Display panel.** The central area where data regarding configuration settings and controls for changing them are displayed.

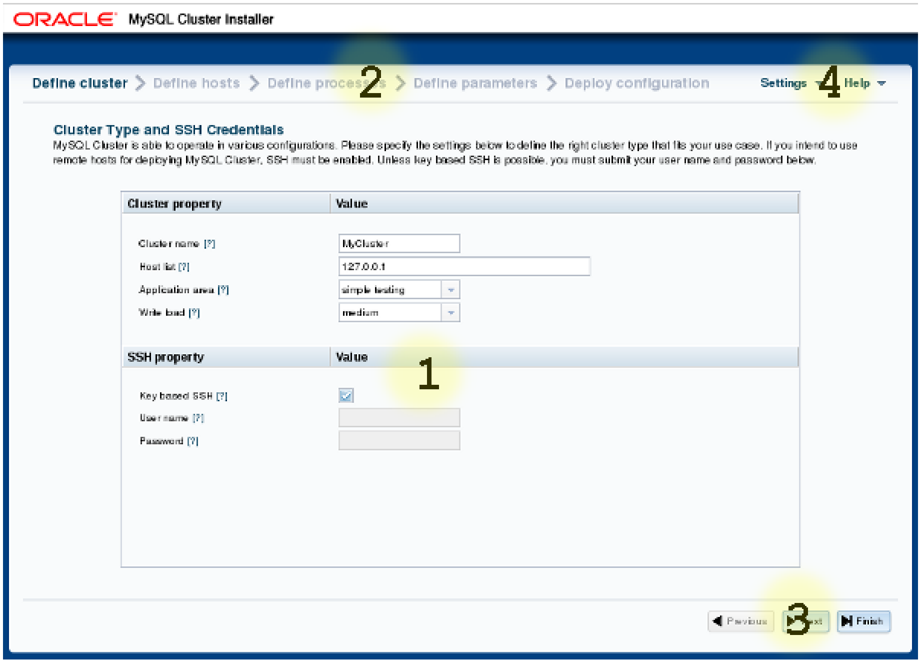
2. **Breadcrumb navigation.** Located in the top left and top center of the GUI, the breadcrumb navigation bar consists of a series of titles linking to screens that correspond to steps in the configuration of a MySQL Cluster. The breadcrumb allows you to jump between these stages in arbitrary order.

3. **Sequential navigation.** This consists of a set of buttons labelled Previous, Next, and Finished, and can be found in the lower right-hand corner of the GUI. The sequential navigation is used to move between steps in the suggested order.

4. **Settings and Help menus.** These menus can be found in the top right corner of the GUI (to the right of the breadcrumb navigation bar). Settings provides a way check and possibly alter configuration settings for the Auto-Installer; Help can be used to access the installer's built-in help files.

The locations of the elements just described are shown here in a typical page in the Auto-Installer; the numbers superimposed thereupon correspond to those used in the preceding list.

Figure 4.3 Layout of the MySQL Cluster Auto-Installer GUI



All of these elements except for the display panel are described in greater detail in the remainder of this section. Section 4.1.3, "Using the MySQL Cluster Auto-Installer", describes the panels shown in the display area as well as the functionality of each panel and the controls it contains.

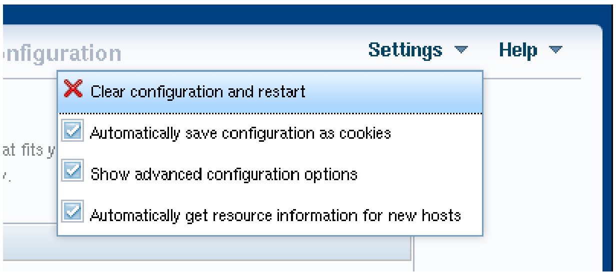
Arbitrary and sequential navigation. The Auto-Installer can display any of a number of pages covering different stages in the setup and configuration of a MySQL Cluster deployment. You can navigate between pages in either of two ways. The first of these is the breadcrumb trail navigation toolbar displaying the titles of the various pages (in which the title of the current page is highlighted and disabled). From these, any desired page, in any desired order, can be reached by selecting the title of the corresponding page. This toolbar is shown here:

Figure 4.4 Detail of MySQL Cluster Auto-Installer breadcrumb navigation, showing page titles/ links

The second navigation mechanism provided by the Auto-Installer consists of the Next, Previous, and Finish sequential navigation buttons at the bottom right of the page. These can be used to move to the next or previous page in predetermined order, or to go to the very last page. The buttons are enabled and disabled as needed, so that you cannot, for example, advance beyond the last page.

Settings and Help menus. These menus are positioned adjacent to one another in the top right corner of the GUI, as shown earlier in this section. The Settings menu is shown here in more detail:

Figure 4.5 MySQL Cluster Auto-Installer Settings menu detail



The entries in the Settings menu are described here, in the following list:

- Clear configuration and restart: Remove all hosts and processes; reset all parameter values to their defaults; start the installer over at the first page.
- Automatically save configuration as cookies: Save your configuration information—such as host names, process data, and parameter values—as a cookie in the browser. When this option is chosen, all information except any SSH password is saved. This means that you can quit and restart the browser, and continue working on the same configuration from where you left off at the end of the previous session).

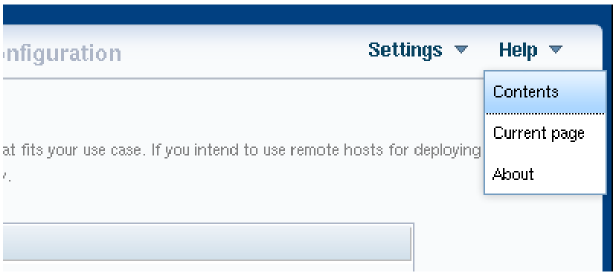
  Since the SSH password is never saved, you must supply this once again at the beginning of a new session, if one is used.
- Show advanced configuration options: Show advanced configuration parameters in the Auto-Installer and make these settable by the user.

  Once set, the advanced parameters continue to be used in the configuration file until they are explicitly changed or reset. This is regardless of whether the advanced parameters are currently visible in the installer; in other words, disabling the menu item does not reset the values of any of these parameters.
- Automatically get resource information for new hosts: Query new hosts automatically for hardware resource information to pre-populate a number of configuration options and values. In this case, the suggested values are not mandatory, but they are used unless explicitly changed using the appropriate editing options in the installer.

As with the installer's navigation elements, one or more of the entries in the Settings menu may be disabled due to choices you have made previously.

The Help menu is shown here, as it appears when expanded:

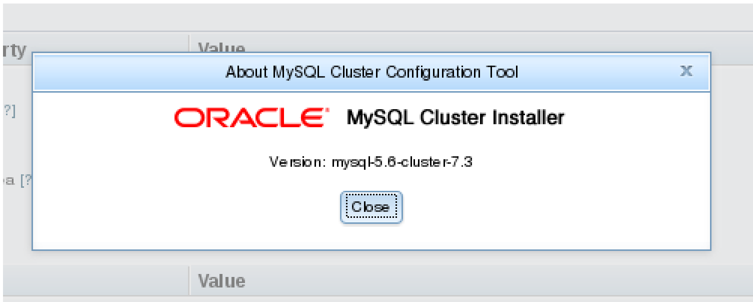
Figure 4.6 The MySQL Cluster Auto-Installer Help menu, expanded



The Help menu provides several options, described in the following list:

- Content: Show the built-in user guide. This is opened in a separate browser window, so that it can be used simultaneously with the installer without interrupting workflow.
- Current page: Open the built-in user guide to the section describing the page currently displayed in the installer.
- About: This will show a small dialog displaying the installer name and the version number of the MySQL Cluster distribution it was supplied with, similar to what is shown here:

  Figure 4.7 The MySQL Cluster Auto-Installer About dialog

The Auto-Installer also provides context-sensitive help in the form of tooltips for most input widgets. One of these tooltips is displayed when the mouse hovers over a widget or the small question mark which can sometimes appear next to a widget label.

In addition, the names of MySQL Cluster configuration parameters are linked to their descriptions in the online MySQL Cluster documentation, so that if you click on the name of a given parameter, the documentation for that parameter is shown in a separate window.

4.1.3 Using the MySQL Cluster Auto-Installer

The MySQL Cluster Auto-Installer consists of several pages, each corresponding to a step in the process used to configure and deploy a MySQL Cluster, and listed here:

- **Welcome**: Begin using the Auto-Installer by choosing either to configure a new MySQL Cluster, or to continue configuring an existing one.
- **Define Cluster**: Set basic information about the cluster as a whole, such as name, hosts, and load type. Here you can also set the SSH authentication type for accessing remote hosts, if needed.
- **Define Hosts**: Identify the hosts where you intend to run MySQL Cluster processes.
- **Define Processes**: Assign one or more processes of a given type or types to each cluster host.
- **Define Attributes**: Set configuration attributes for processes or types of processes.
- **Deploy Cluster**: Deploy the cluster with the configuration set previously; start and stop the deployed cluster.

The following sections describe in greater detail the purpose and function of each of these pages, in the order just listed.

4.1.3.1 Starting the MySQL Cluster Auto-Installer

The Auto-Installer is provided together with the MySQL Cluster software. (See Chapter 4, *MySQL Cluster Installation*.) The present section explains how to start the installer. You can do by invoking the `ndb_setup.py` executable. `ndb_setup.py` is found in the `bin` within the MySQL Cluster installation directory; a typical location might be `/usr/local/mysql/bin` on a Linux system or `C:\Program Files\MySQL\MySQL Server 5.6\bin` on a Windows system, but this can vary according to where the MySQL Cluster software is installed on your system.

On Windows, you can also start the installer by running `setup.bat` in the MySQL Cluster installation directory. When invoked from the command line, it accepts the same options as does `ndb_setup.py`.

`ndb_setup.py` can be started with any of several options that affect its operation, but it is usually sufficient to allow the default settings be used, in which case you can start `ndb_setup.py` by either of the following two methods:

1. Navigate to the MySQL Cluster `bin` directory in a terminal and invoke it from the command line, without any additional arguments or options, like this:

   ```
   shell> ndb_setup
   ```

   This works regardless of operating platform.

2. Navigate to the MySQL Cluster `bin` directory in a file browser (such Windows Explorer on Windows, or Konqueror, Dolphin, or Nautilus on Linux) and activate (usually by double-clicking) the **ndb_setup.py** file icon. This works on Windows, and should work with most common Linux desktops as well.

   On Windows, you can also navigate to the MySQL Cluster installation directory and activate the **setup.bat** file icon.

In either case, once `ndb_setup.py` is invoked, the Auto-Installer's **Welcome** screen should open in the system's default Web browser.

In some cases, you may wish to use non-default settings for the installer, such as specifying a different port for the Auto-Installer's included Web server to run on, in which case you must invoke `ndb_setup.py` with one or more startup options with values overriding the necessary defaults. The same startup options can be used on Windows systems with the `setup.bat` file supplied for such platforms in the MySQL Cluster software distribution. This can be done using the command line, but if you want or need to start the installer from a desktop or file browser while emplying one or more of these options, it is also possible to create a script or batch file containing the proper invocation, then to double-click its file icon in the file browser to start the installer. (On Linux systems, you might also need to make the script file executable first.) For information about advanced startup options for the MySQL Cluster Auto-Installer, see Section 6.23, "`ndb_setup.py` — Start browser-based Auto-Installer for MySQL Cluster".

4.1.3.2 MySQL Cluster Auto-Installer Welcome Screen

The **Welcome** screen is loaded in the default browser when `ndb_setup.py` is invoked, as shown here:

Figure 4.8 The MySQL Cluster Auto-Installer Welcome screen (Closeup)

This screen provides the following two choices for entering the installer, one of which must be selected to continue:

1. **Create New MySQL Cluster**: Start the Auto-Installer with a completely new cluster to be set up and deployed.

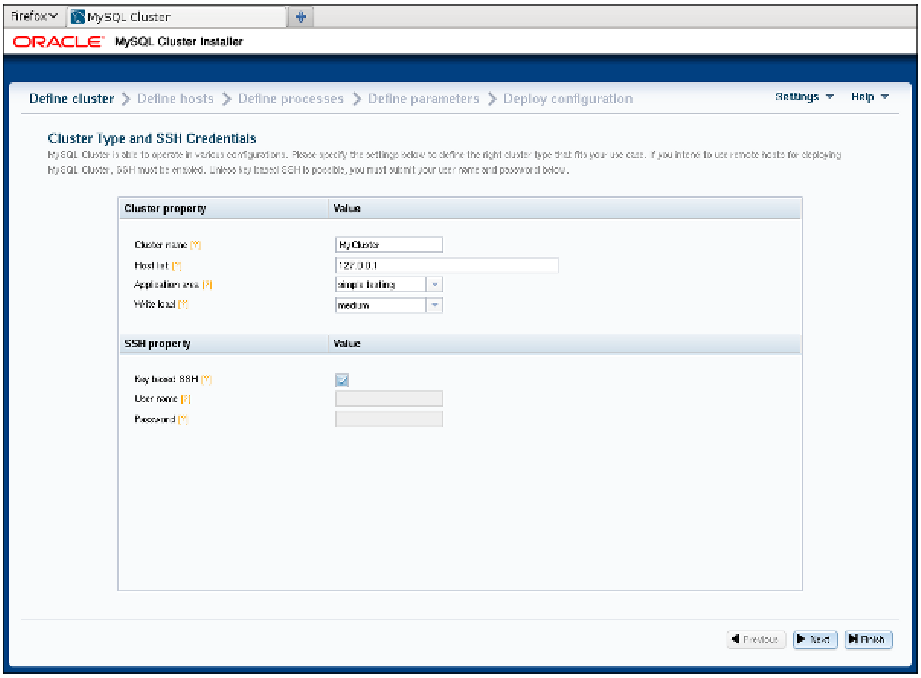
2. **Continue Previous Cluster Configuration**: Start the Auto-Installer at the same point where the previous session ended, with all previous settings preserved.

The second option requires that the browser be able to access its cookies from the previous session, as these provide the mechanism by which configuration and other information generated during a session is stored. In other words, to continue the previous session with the Auto-Installer, you must use the same web browser running on the same host as you did for the previous session.

4.1.3.3 MySQL Cluster Auto-Installer Define Cluster Screen

The **Define Cluster** screen is the first screen to appear following the choice made in the **Welcome screen**, and is used for setting general properties of the cluster. The layout of the **Define Cluster** screen is shown here:

Figure 4.9 The MySQL Cluster Auto-Installer Define Cluster screen



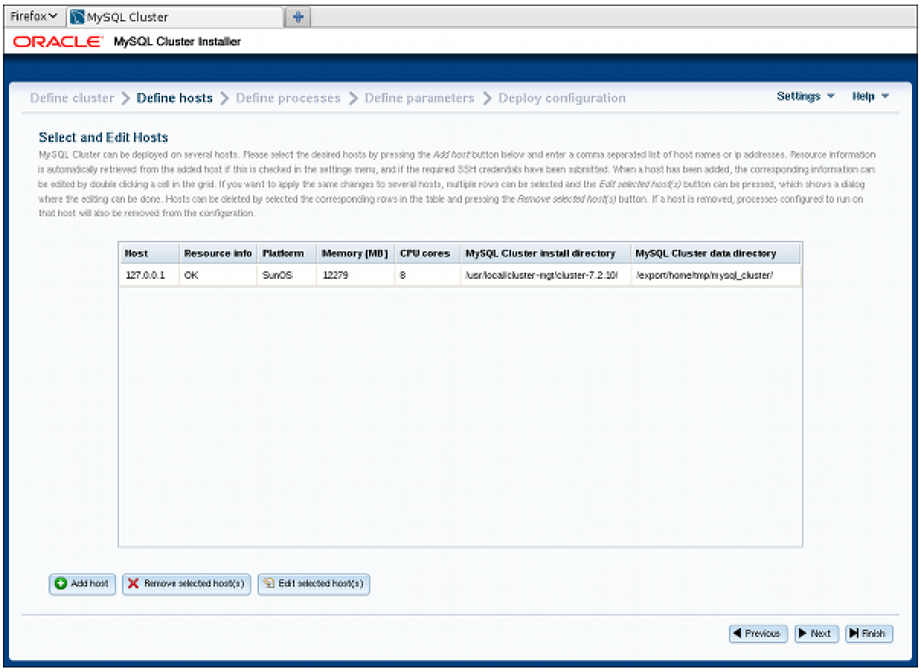
The **Define Cluster** screen allows you to set a number of general properties for the cluster, as described in this list:

- **Cluster name**: A name that identifies the cluster. The default is `MyCluster`.
- **Host list**: A comma-delimited list of one or more hosts where cluster processes should run. By default, this is `127.0.0.1`. If you add remote hosts to the list, you must be able to connect to them using the **SSH Credentials** supplied.
- **Application type**: Choose one of the following:
  1. Simple testing: Minimal resource usage for small-scale testing. This the default. *Not intended for production environments*.
  2. Web: Maximize performance for the given hardware.
  3. Real-time: Maximize performance while maximizing sensitivity to timeouts in order to minimize the time needed to detect failed cluster processes.
- **Write load**: Choose a level for the anticipated number of writes for the cluster as a whole. You can choose any one of the following levels:
  1. Low: The expected load includes fewer than 100 write transactions for second.
  2. Medium: The expected load includes 100 to 1000 write transactions per second.
  3. High: The expected load includes more than 1000 write transactions per second.
- **SSH Credentials**: Choose **Key-Based SSH** or enter **User** and **Password** credentials. The SSH key or a user name with password is required for connecting to any remote hosts specified in the **Host list**. By default, **Key-Based SSH** is selected, and the **User** and **Password** fileds are blank.

4.1.3.4 MySQL Cluster Auto-Installer Define Hosts Screen

The **Define Hosts** screen, shown here, provides a means of viewing and specifying several key properties of each cluster host:

Figure 4.10 MySQL Cluster Define Hosts screen



The hosts currently entered are displayed in the grid with various pieces of information. You can add hosts by clicking the Add hosts button and entering a list of one or more comma-separated host names, IP addresses, or both (as when editing the host list on the **Define Cluster** screen).

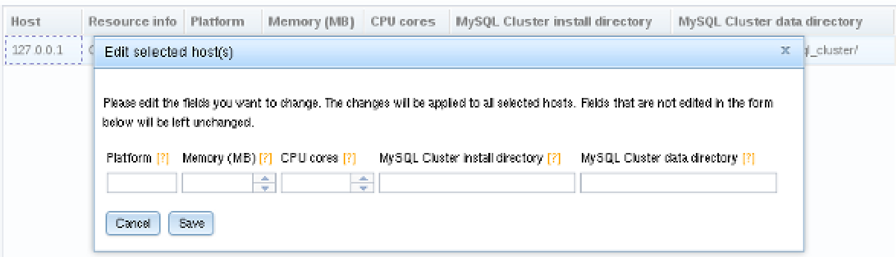
Similarly, you can remove one or more hosts using the button labelled Remove selected host(s). When you remove a host in this fashion, any process which was configured for that host is also removed.

If **Automatically get resource information for new hosts** is checked in the Settings menu, the Auto-Installer attempts to retrieve the platform name, amount of memory, and number of CPU cores and to fill these in automatically. The status of this is displayed in the *Resource info* column. Fetching the information from remote hosts is not instantaneous and may take some time, particularly from remote hosts running Windows.

If the SSH user credentials on the *Define Cluster* screen are changed, the tool tries to refresh the hardware information from any hosts for which information is missing. However, if a given field has already been edited, the user-supplied information is *not* overwritten by any value fetched from that host.

The hardware resource information, platform name, installation directory, and data directory can be edited by the user by clicking the corresponding cell in the grid, by selecting one or more hosts and clicking the button labelled **Edit selected host(s)**. This causes a dialog box to appear, in which these fields can be edited, as shown here:

Figure 4.11 MySQL Cluster Auto-Installer Edit Hosts dialog

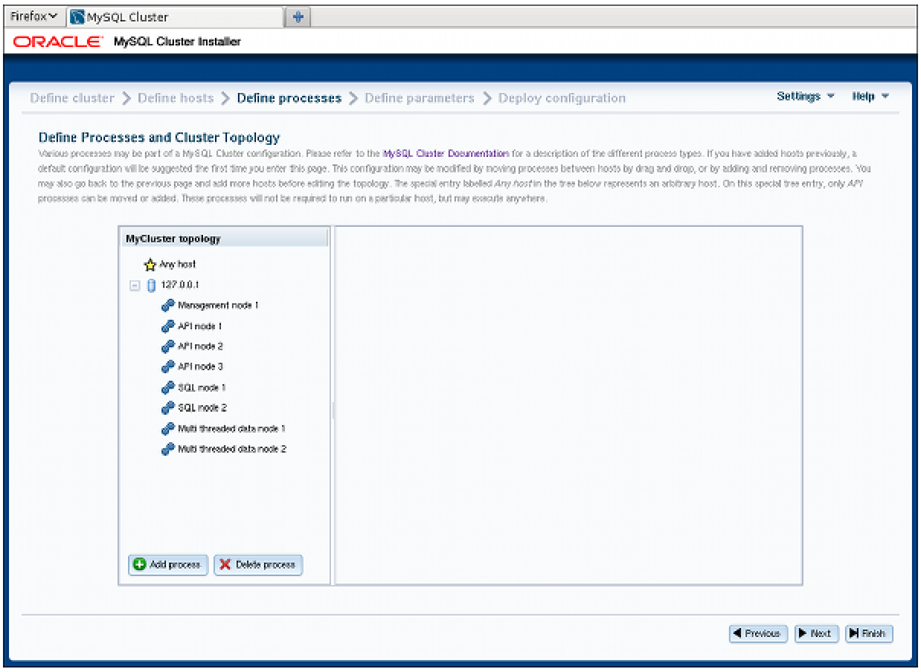


When more than one host is selected, any edited values are applied to all selected hosts.

4.1.3.5 MySQL Cluster Auto-Installer Define Processes Screen

The **Define Processes** screen, shown here, provides a way to assign MySQL Cluster processes (nodes) to cluster hosts:

Figure 4.12 MySQL Cluster Auto-Installer Define Processes dialog



The left-hand portion of this screen contains a process tree showing cluster hosts and processes set up to run on each one. On the right is a panel which displays information about the item currently selected in the tree.

When this screen is accessed for the first time for a given cluster, a default set of processes is defined for you, based on the number of hosts. If you later return to the **Define Hosts** screen, remove all hosts, and add new hosts, this also causes a new default set of processes to be defined.

MySQL Cluster processes are of the following types:

- **Management node.** Performs administrative tasks such as stopping individual data nodes, querying node and cluster status, and making backups. Executable: `ndb_mgmd`.
- **Single-threaded data node.** Stores data and executes queries. Executable: `ndbd`.
- **Multi threaded data node.** Stores data and executes queries with multiple worker threads executing in parallel. Executable: `ndbmtd`.
- **SQL node.** MySQL server for executing SQL queries against `NDB`. Executable: `mysqld`.
- **API node.** A client accessing data in `NDB` by means of the NDB API or other low-level client API, rather than by using SQL. See MySQL Cluster API Developer Guide, for more information.

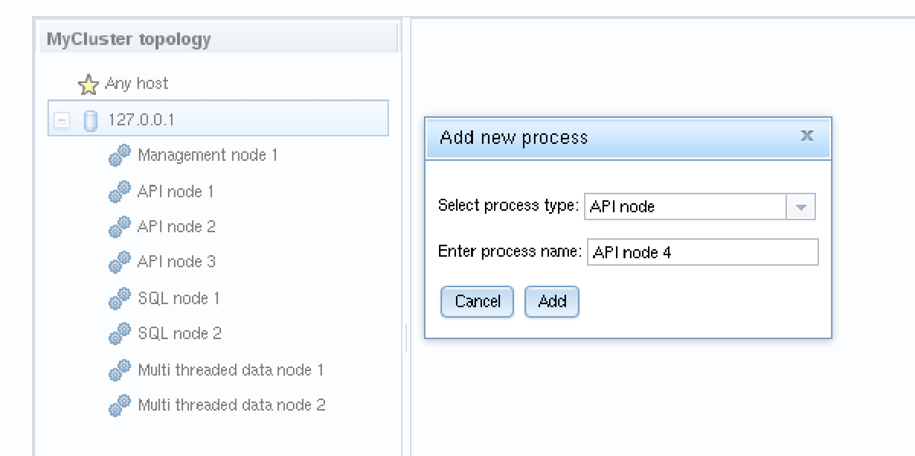
For more information about process (node) types, see Section 3.1, “MySQL Cluster Core Concepts”.

Processes shown in the tree are numbered sequentially by type, for each host—for example, `SQL node 1`, `SQL node 2`, and so on—to simplify identification.

Each management node, data node, or SQL process must be assigned to a specific host, and is not allowed to run on any other host. An API node *may* be assigned to a single host, but this is not required. Instead, you can assign it to the special Any host entry which the tree also contains in addition to any other hosts, and which acts as a placeholder for processes that are allowed to run on any host. *Only API processes may use this Any host entry*.

Adding processes. To add a new process to a given host, either right-click that host's entry in the tree, then select the **Add process** popup when it appears, or select a host in the process tree, and press the Add process button below the process tree. Performing either of these actions opens the add process dialog, as shown here:

Figure 4.13 MySQL Cluster Auto-Installer Add Process Dialog



Here you can select from among the available process types described earlier this section; you can also enter an arbitrary process name to take the place of the suggested value, if desired.

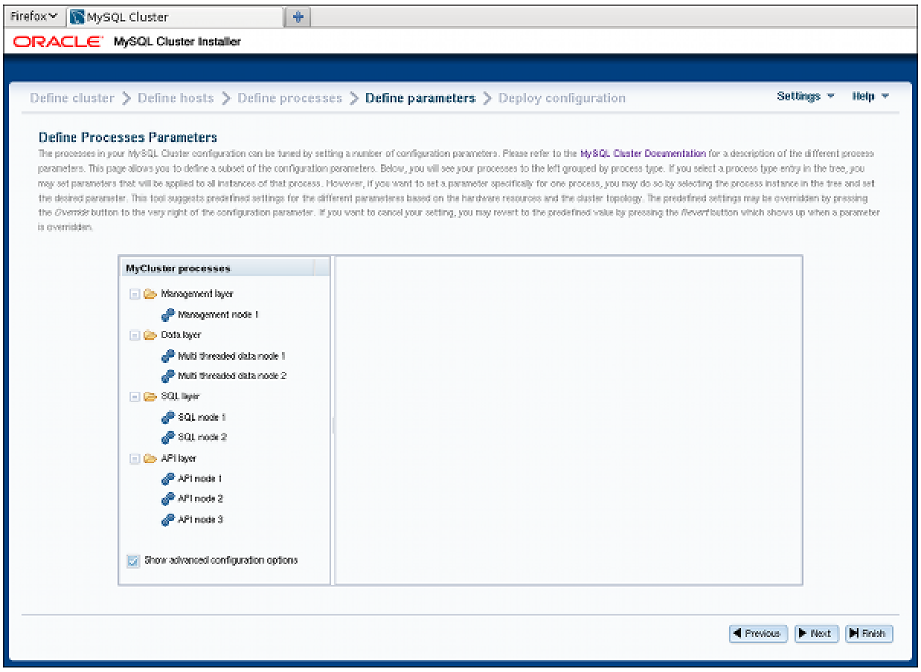
Removing processes. To delete a process, right-click on a process in the tree and select **delete process** from the pop up menu that appears, or select a process, then use the delete process button below the process tree.

When a process is selected in the process tree, information about that process is displayed in the panel to the right of the tree, where you can change the process name and possibly its type. *Important*: Currently, you can change a single-threaded data node (`ndbd`) to a multi-threaded data node (`ndbmtd`), or the reverse, only; no other process type changes are allowed. If you want to make a change between any other process types, you must delete the original process first, then add a new process of the desired type.

4.1.3.6 MySQL Cluster Auto-Installer Define Attributes Screen

This screen has a layout similar to that of the **Define Processes** screen, with a process tree at the left. Unlike that screen's tree, the **Define Attributes** process tree is organized by process or node type, with single-threaded and multi-threaded data nodes considered to be of the same type for this purpose, in groups labelled Management Layer, Data Layer, SQL Layer, and API Layer. A panel to the right of this tree displays information regarding the item currently selected. The **Define Attributes** screen is shown here:

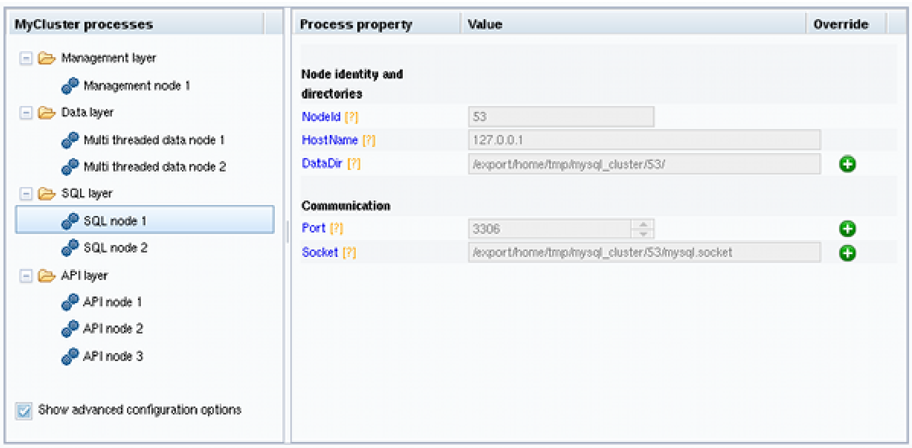
Figure 4.14 MySQL Cluster Auto-Installer Define Attributes screen



A checkbox labelled **Show advanced configuration** is located below the process tree. Checking this box makes advanced options visible in the information pane. These options are set and used whether or not they are visible.

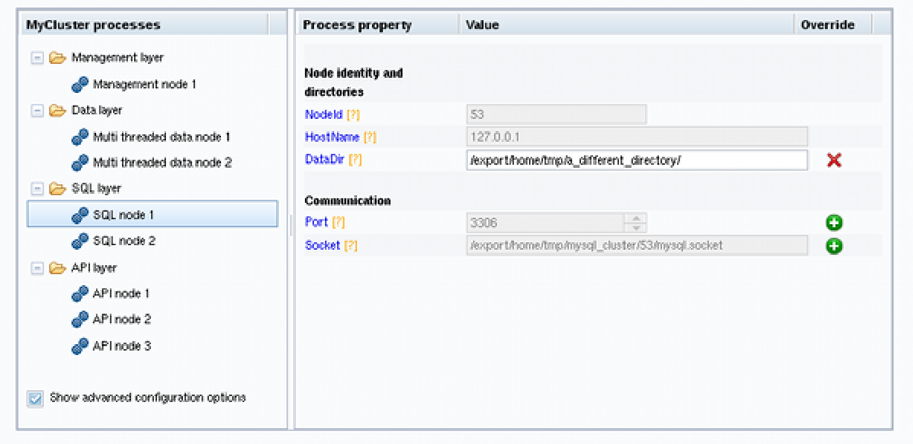
You can edit attributes for a single process by selecting that process from the tree, or for all processes of the same type in the cluster by selecting one of the Layer folders. A per-process value set for a given attribute overrides any per-group setting for that attribute that would otherwise apply to the process in question. An example of such an information panel (for an SQL process) is shown here:

Figure 4.15 Define Attributes Detail With SQL Process Attributes



For some of the attributes shown in the information panel, a button bearing a plus sign is displayed to the right, which means that the value of this attribute can be overridden. This + button activates an input widget for the attribute, enabling you to change its value. When the value has been overridden, this button changes into a button showing an X, as shown here:

Figure 4.16 Define Attributes Detail, Overriding Attribute Default Value



Clicking the X button next to an attribute undoes any changes made to it; it immediately reverts to the predefined value.

All configuration attributes have predefined values calculated by the installer, based such factors as host name, node ID, node type, and so on. In most cases, these values may be left as they are. If you are not familiar with it already, it is highly recommended that you read the applicable documentation before making changes to any of the attribute values. To make finding this information easier, each attribute name shown in the information panel is linked to its description in the online MySQL Cluster documentation.

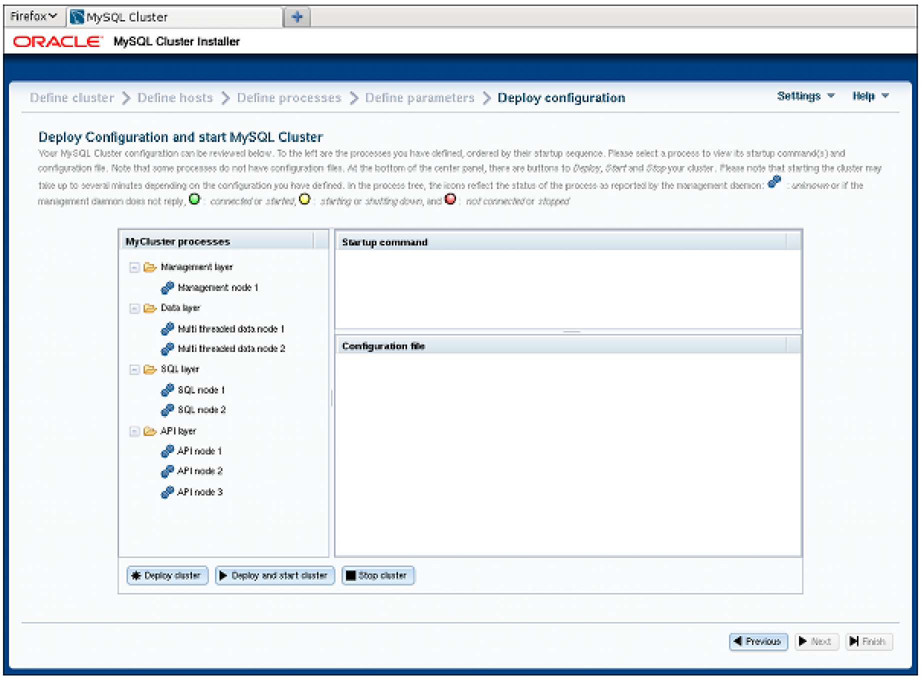
4.1.3.7 MySQL Cluster Auto-Installer Deploy Cluster Screen

This screen allows you to perform the following tasks:

- Review process startup commands and configuration files to be applied
- Distribute configuration files by creating any necessary files and directories on all cluster hosts—that is, *deploy* the cluster as presently configured
- Start and stop the cluster

The **Deploy Cluster** screen is shown here:

Figure 4.17 MySQL Cluster Auto-Installer Deploy Cluster screen



Like the **Define Attributes** screen, this screens features a process tree, organized by process type, on the left hand side. Next to each process is a status icon whose color indicates the current status of the process: green if it is running; yellow if it is starting or stopping; red if the process is stopped.

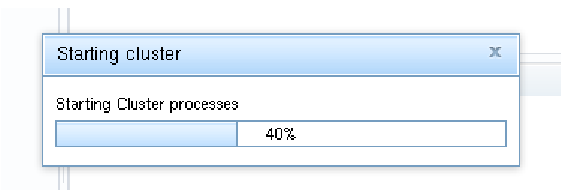
To the right of the process tree are two information panels, the upper panel showing the startup command or commands needed to start the selected process. (For some processes, more than one command may be required—for example, if initialization is necessary.) The lower panel shows the contents of the configuration file, if any, for the given process; currently, the management node process is only type of process having a configuration file. Other process types are configured using command-line parameters when starting the process, or by obtaining configuration information from the management nodes as needed in real time.

Three buttons are located immediately below the process tree. These are labelled as and perform the functions described in the following list:

- Deploy cluster: Verify that the configuration is valid. Create any directories required on the cluster hosts, and distribute the configuration files onto the hosts. A progress bar shows how far the deployment has proceeded.
- Start cluster: The cluster is deployed as with Deploy cluster, after which all cluster processes are started in the correct order.

  Starting these processes may take some time. If the estimated time to completion is too large, the installer provides an opportunity to cancel or to continue of the startup procedure. A progress bar indicates the current status of the startup procedure, as shown here:

Figure 4.18 Progress Bar With Status of Node Startup Process



The process status icons adjoining the process tree mentioned previously also update with the status of each process.

- Stop cluster: After the cluster has been started, you can stop it using the this. As with starting the cluster, cluster shutdown is not instantaneous, and may require some time complete. A progress bar, similar to that displayed during cluster startup, shows the approximate current status of the cluster shutdown procedure, as do the process status icons adjoining the process tree.

Prior to MySQL Cluster NDB 7.3.3, SQL nodes were started with all options employed on the command line. Beginning with MySQL Cluster NDB 7.3.3, the Auto-Installer generates a `my.cnf` file containing the appropriate options for each `mysqld` process in the cluster. (Bug #16994782)

4.2 Installation of MySQL Cluster on Linux

This section covers installation methods for MySQL Cluster on Linux and other Unix-like operating systems. While the next few sections refer to a Linux operating system, the instructions and procedures given there should be easily adaptable to other supported Unix-like platforms. For manual installation and setup instructions specific to Windows systems, see Section 4.3, "Installing MySQL Cluster on Windows".

Each MySQL Cluster host computer must have the correct executable programs installed. A host running an SQL node must have installed on it a MySQL Server binary (`mysqld`). Management nodes require the management server daemon (`ndb_mgmd`); data nodes require the data node daemon (`ndbd` or `ndbmtd`). It is not necessary to install the MySQL Server binary on management node hosts and data node hosts. It is recommended that you also install the management client (`ndb_mgm`) on the management server host.

Installation of MySQL Cluster on Linux can be done using precompiled binaries from Oracle (downloaded as a .tar.gz archive), with RPM packages (also available from Oracle), or from source code. All three of these installation methods are described in the section that follow.

Regardless of the method used, it is still necessary following installation of the MySQL Cluster binaries to create configuration files for all cluster nodes, before you can start the cluster. See Section 4.4, "Initial Configuration of MySQL Cluster".

4.2.1 Installing a MySQL Cluster Binary Release on Linux

This section covers the steps necessary to install the correct executables for each type of Cluster node from precompiled binaries supplied by Oracle.

For setting up a cluster using precompiled binaries, the first step in the installation process for each cluster host is to download the latest MySQL Cluster NDB 7.3 or later binary archive (`mysql-cluster-gpl-7.3.12-linux-i686-glibc23.tar.gz` or `mysql-cluster-gpl-7.4.9-linux-i686-glibc23.tar.gz`) from the MySQL Cluster downloads area. We assume that you have placed this file in each machine's `/var/tmp` directory. (If you do require a custom binary, see Installing MySQL Using a Development Source Tree.)

> **Note**
>
> After completing the installation, do not yet start any of the binaries. We show you how to do so following the configuration of the nodes (see Section 4.4, "Initial Configuration of MySQL Cluster").

SQL nodes. On each of the machines designated to host SQL nodes, perform the following steps as the system `root` user:

1. Check your `/etc/passwd` and `/etc/group` files (or use whatever tools are provided by your operating system for managing users and groups) to see whether there is already a `mysql` group and `mysql` user on the system. Some OS distributions create these as part of the operating system

installation process. If they are not already present, create a new `mysql` user group, and then add a `mysql` user to this group:

```
shell> groupadd mysql
shell> useradd -g mysql -s /bin/false mysql
```

The syntax for `useradd` and `groupadd` may differ slightly on different versions of Unix, or they may have different names such as `adduser` and `addgroup`.

2. Change location to the directory containing the downloaded file, unpack the archive, and create a symbolic link named `mysql` to the `mysql` directory.

   Note

   The actual file and directory names vary according to the MySQL Cluster version number.

```
shell> cd /var/tmp
shell> tar -C /usr/local -xzvf mysql-cluster-gpl-7.4.9-linux2.6.tar.gz
shell> ln -s /usr/local/mysql-cluster-gpl-7.4.9-linux2.6-i686 /usr/local/mysql
```

3. Change location to the `mysql` directory and run the supplied script for creating the system databases:

```
shell> cd mysql
shell> scripts/mysql_install_db --user=mysql
```

4. Set the necessary permissions for the MySQL server and data directories:

```
shell> chown -R root .
shell> chown -R mysql data
shell> chgrp -R mysql .
```

5. Copy the MySQL startup script to the appropriate directory, make it executable, and set it to start when the operating system is booted up:

```
shell> cp support-files/mysql.server /etc/rc.d/init.d/
shell> chmod +x /etc/rc.d/init.d/mysql.server
shell> chkconfig --add mysql.server
```

(The startup scripts directory may vary depending on your operating system and version—for example, in some Linux distributions, it is `/etc/init.d`.)

Here we use Red Hat's `chkconfig` for creating links to the startup scripts; use whatever means is appropriate for this purpose on your platform, such as `update-rc.d` on Debian.

Remember that the preceding steps must be repeated on each machine where an SQL node is to reside.

Data nodes. Installation of the data nodes does not require the `mysqld` binary. Only the MySQL Cluster data node executable `ndbd` (single-threaded) or `ndbmtd` (multi-threaded) is required. These binaries can also be found in the `.tar.gz` archive. Again, we assume that you have placed this archive in `/var/tmp`.

As system `root` (that is, after using `sudo`, `su root`, or your system's equivalent for temporarily assuming the system administrator account's privileges), perform the following steps to install the data node binaries on the data node hosts:

1. Change location to the `/var/tmp` directory, and extract the `ndbd` and `ndbmtd` binaries from the archive into a suitable directory such as `/usr/local/bin`:

```
shell> cd /var/tmp
shell> tar -zxvf mysql-5.6.27-ndb-7.4.9-linux-i686-glibc23.tar.gz
shell> cd mysql-5.6.27-ndb-7.4.9-linux-i686-glibc23
shell> cp bin/ndbd /usr/local/bin/ndbd
shell> cp bin/ndbmtd /usr/local/bin/ndbmtd
```

(You can safely delete the directory created by unpacking the downloaded archive, and the files it contains, from `/var/tmp` once `ndb_mgm` and `ndb_mgmd` have been copied to the executables directory.)

2. Change location to the directory into which you copied the files, and then make both of them executable:

```
shell> cd /usr/local/bin
shell> chmod +x ndb*
```

The preceding steps should be repeated on each data node host.

Although only one of the data node executables is required to run a MySQL Cluster data node, we have shown you how to install both `ndbd` and `ndbmtd` in the preceding instructions. We recommend that you do this when installing or upgrading MySQL Cluster, even if you plan to use only one of them, since this will save time and trouble in the event that you later decide to change from one to the other.

> **Note**
>
> The data directory on each machine hosting a data node is `/usr/local/mysql/data`. This piece of information is essential when configuring the management node. (See Section 4.4, "Initial Configuration of MySQL Cluster".)

Management nodes. Installation of the management node does not require the `mysqld` binary. Only the MySQL Cluster management server (`ndb_mgmd`) is required; you most likely want to install the management client (`ndb_mgm`) as well. Both of these binaries also be found in the `.tar.gz` archive. Again, we assume that you have placed this archive in `/var/tmp`.

As system `root`, perform the following steps to install `ndb_mgmd` and `ndb_mgm` on the management node host:

1. Change location to the `/var/tmp` directory, and extract the `ndb_mgm` and `ndb_mgmd` from the archive into a suitable directory such as `/usr/local/bin`:

```
shell> cd /var/tmp
shell> tar -zxvf mysql-5.6.27-ndb-7.4.9-linux2.6-i686.tar.gz
shell> cd mysql-5.6.27-ndb-7.4.9-linux2.6-i686
shell> cp bin/ndb_mgm* /usr/local/bin
```

(You can safely delete the directory created by unpacking the downloaded archive, and the files it contains, from `/var/tmp` once `ndb_mgm` and `ndb_mgmd` have been copied to the executables directory.)

2. Change location to the directory into which you copied the files, and then make both of them executable:

```
shell> cd /usr/local/bin
shell> chmod +x ndb_mgm*
```

In Section 4.4, "Initial Configuration of MySQL Cluster", we create configuration files for all of the nodes in our example MySQL Cluster.

4.2.2 Installing MySQL Cluster from RPM

This section covers the steps necessary to install the correct executables for each type of MySQL Cluster node using RPM packages supplied by Oracle.

RPMs are available for both 32-bit and 64-bit Linux platforms. The filenames for these RPMs use the following pattern:

```
MySQL-Cluster-component-producttype-ndbversion.distribution.architecture.rpm
component:= {server | client [| other]}
producttype:= {gpl | advanced}
ndbversion:= major.minor.release
distribution:= {sles10 | rhel5 | el6}
architecture:= {i386 | x86_64}
```

The *`component`* can be `server` or `client`. (Other values are possible, but since only the `server` and `client` components are required for a working MySQL Cluster installation, we do not discuss them here.) The *`producttype`* for Community RPMs downloaded from http://dev.mysql.com/downloads/cluster/ is always `gpl`; `advanced` is used to indicate commercial releases. *`ndbversion`* represents the three-part `NDB` storage engine version number in 7.3.*x* or 7.4.*x* format. The *`distribution`* can be one of `sles11` (SUSE Enterprise Linux 11), `rhel5` (Oracle Linux 5, Red Hat Enterprise Linux 4 and 5), or `el6` (Oracle Linux 6, Red Hat Enterprise Linux 6) The *`architecture`* is `i386` for 32-bit RPMs and `x86_64` for 64-bit versions.

For a MySQL Cluster, one and possibly two RPMs are required:

- The `server` RPM (for example, `MySQL-Cluster-server-gpl-7.3.12-1.sles11.i386.rpm` or `MySQL-Cluster-server-gpl-7.4.9-1.sles11.i386.rpm`), which supplies the core files needed to run a MySQL Server with `NDBCLUSTER` storage engine support (that is, as a MySQL Cluster SQL node) as well as all MySQL Cluster executables, including the management node, data node, and `ndb_mgm` client binaries. This RPM is always required for installing MySQL Cluster.

- If you do not have your own client application capable of administering a MySQL server, you should also obtain and install the `client` RPM (for example, `MySQL-Cluster-client-gpl-7.3.12-1.sles11.i386.rpm` or `MySQL-Cluster-client-gpl-7.4.9-1.sles11.i386.rpm`), which supplies the `mysql` client

The MySQL Cluster version number in the RPM file names (shown here as `7.3.12` or `7.4.9`, depending on whether you are installing MySQL Cluster NDB 7.3 or MySQL Cluster NDB 7.4) can vary according to the version which you are actually using. *It is very important that all of the Cluster RPMs to be installed have the same version number*. The *`architecture`* designation should also be appropriate to the machine on which the RPM is to be installed; in particular, you should keep in mind that 64-bit RPMs cannot be used with 32-bit operating system.

Data nodes. On a computer that is to host a cluster data node it is necessary to install only the `server` RPM. To do so, copy this RPM to the data node host, and run the following command as the system root user, replacing the name shown for the RPM as necessary to match that of the RPM downloaded from the MySQL web site:

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.3.12-1.sles11.i386.rpm
```

or

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.4.9-1.sles11.i386.rpm
```

Although this installs all MySQL Cluster binaries, only the program `ndbd` or `ndbmtd` (both in `/usr/sbin`) is actually needed to run a MySQL Cluster data node.

SQL nodes. On each machine to be used for hosting a cluster SQL node, install the `server` RPM by executing the following command as the system root user, replacing the name shown for the RPM as necessary to match the name of the RPM downloaded from the MySQL web site:

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.3.12-1.sles11.i386.rpm
```

or

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.4.9-1.sles11.i386.rpm
```

This installs the MySQL server binary (`mysqld`) with `NDB` storage engine support in the `/usr/sbin` directory, as well as all needed MySQL Server support files. It also installs the `mysql.server` and `mysqld_safe` startup scripts (in `/usr/share/mysql` and `/usr/bin`, respectively). The RPM installer should take care of general configuration issues (such as creating the `mysql` user and group, if needed) automatically.

To administer the SQL node (MySQL server), you should also install the `client` RPM, as shown here:

```
shell> rpm -Uhv MySQL-Cluster-client-gpl-7.3.12-1.sles11.i386.rpm
```

or

```
shell> rpm -Uhv MySQL-Cluster-client-gpl-7.4.9-1.sles11.i386.rpm
```

This installs the `mysql` client program.

Management nodes. To install the MySQL Cluster management server, it is necessary only to use the `server` RPM. Copy this RPM to the computer intended to host the management node, and then install it by running the following command as the system root user (replace the name shown for the RPM as necessary to match that of the `server` RPM downloaded from the MySQL web site):

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.3.12-1.sles11.i386.rpm
```

or

```
shell> rpm -Uhv MySQL-Cluster-server-gpl-7.4.9-1.sles11.i386.rpm
```

Although this RPM installs many other files, only the management server binary `ndb_mgmd` (in the `/usr/sbin` directory) is actually required for running a management node. The `server` RPM also installs `ndb_mgm`, the `NDB` management client.

See Installing MySQL on Linux Using RPM Packages, for general information about installing MySQL using RPMs supplied by Oracle.

After installing from RPM, you still need to configure the cluster as discussed in Section 4.4, "Initial Configuration of MySQL Cluster".

> **Note**
>
> A number of RPMs used by MySQL Cluster NDB 7.1 were made obsolete and discontinued in MySQL Cluster NDB 7.3. These include the former `MySQL-Cluster-clusterj`, `MySQL-Cluster-extra`, `MySQL-Cluster-management`, `MySQL-Cluster-storage`, and `MySQL Cluster-tools` RPMs. The former contents of all of these packages are now included in the `MySQL-Cluster-server` RPM.

4.2.3 Building MySQL Cluster from Source on Linux

This section provides information about compiling MySQL Cluster on Linux and other Unix-like platforms. Building MySQL Cluster from source is similar to building the standard MySQL Server, although it differs in a few key respects discussed here. For general information about building MySQL from source, see Installing MySQL from Source. For information about compiling MySQL Cluster on Windows platforms, see Section 4.3.2, “Compiling and Installing MySQL Cluster from Source on Windows”.

Building MySQL Cluster requires using the MySQL Cluster sources. These are available from the MySQL Cluster downloads page at http://dev.mysql.com/downloads/cluster/. The archived source file should have a name similar to `mysql-cluster-gpl-7.3.12.tar.gz` (MySQL Cluster NDB 7.3) or `mysql-cluster-gpl-7.4.9.tar.gz` (MySQL Cluster NDB 7.4). You can also obtain MySQL development sources from launchpad.net. *Building MySQL Cluster from standard MySQL Server 5.6 sources is not supported*.

The `WITH_NDBCLUSTER_STORAGE_ENGINE` option for `CMake` causes the binaries for the management nodes, data nodes, and other MySQL Cluster programs to be built; it also causes `mysqld` to be compiled with `NDB` storage engine support. This option (or its alias `WITH_NDBCLUSTER`) is required when building MySQL Cluster.

> **Important**
>
> In MySQL Cluster NDB 7.3 and later, the `WITH_NDB_JAVA` option is enabled by default. This means that, by default, if `CMake` cannot find the location of Java on your system, the configuration process fails; if you do not wish to enable Java and ClusterJ support, you must indicate this explicitly by configuring the build using `-DWITH_NDB_JAVA=OFF`. Use `WITH_CLASSPATH` to provide the Java classpath if needed.

For more information about `CMake` options specific to building MySQL Cluster, see Options for Compiling MySQL Cluster.

After you have run `make && make install` (or your system's equivalent), the result is similar to what is obtained by unpacking a precompiled binary to the same location.

Management nodes. When building from source and running the default `make install`, the management server and management client binaries (`ndb_mgmd` and `ndb_mgm`) can be found in `/usr/local/mysql/bin`. Only `ndb_mgmd` is required to be present on a management node host; however, it is also a good idea to have `ndb_mgm` present on the same host machine. Neither of these executables requires a specific location on the host machine's file system.

Data nodes. The only executable required on a data node host is the data node binary `ndbd` or `ndbmtd`. (`mysqld`, for example, does not have to be present on the host machine.) By default, when building from source, this file is placed in the directory `/usr/local/mysql/bin`. For installing on multiple data node hosts, only `ndbd` or `ndbmtd` need be copied to the other host machine or machines. (This assumes that all data node hosts use the same architecture and operating system; otherwise you may need to compile separately for each different platform.) The data node binary need not be in any particular location on the host's file system, as long as the location is known.

When compiling MySQL Cluster from source, no special options are required for building multi-threaded data node binaries. Configuring the build with `NDB` storage engine support causes `ndbmtd` to be built automatically; `make install` places the `ndbmtd` binary in the installation `bin` directory along with `mysqld`, `ndbd`, and `ndb_mgm`.

SQL nodes. If you compile MySQL with clustering support, and perform the default installation (using `make install` as the system `root` user), `mysqld` is placed in `/usr/local/mysql/bin`. Follow the steps given in Installing MySQL from Source to make `mysqld` ready for use. If you want to run multiple SQL nodes, you can use a copy of the same `mysqld` executable and its associated support files on several machines. The easiest way to do this is to copy the entire `/usr/local/mysql` directory and all directories and files contained within it to the other SQL node host or hosts, then repeat the steps from Installing MySQL from Source on each machine. If you configure the build with a nondefault `PREFIX` option, you must adjust the directory accordingly.

In Section 4.4, “Initial Configuration of MySQL Cluster”, we create configuration files for all of the nodes in our example MySQL Cluster.

4.3 Installing MySQL Cluster on Windows

This section describes installation procedures for MySQL Cluster on Windows hosts. MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 binaries for Windows can be obtained from http://dev.mysql.com/downloads/cluster/. For information about installing MySQL Cluster on Windows from a binary release provided by Oracle, see Section 4.3.1, "Installing MySQL Cluster on Windows from a Binary Release".

It is also possible to compile and install MySQL Cluster from source on Windows using Microsoft Visual Studio. For more information, see Section 4.3.2, "Compiling and Installing MySQL Cluster from Source on Windows".

4.3.1 Installing MySQL Cluster on Windows from a Binary Release

This section describes a basic installation of MySQL Cluster on Windows using a binary `no-install` MySQL Cluster release provided by Oracle, using the same 4-node setup outlined in the beginning of this section (see Chapter 4, *MySQL Cluster Installation*), as shown in the following table:

Node	IP Address
Management (MGMD) node	192.168.0.10
MySQL server (SQL) node	192.168.0.20
Data (NDBD) node "A"	192.168.0.30
Data (NDBD) node "B"	192.168.0.40

As on other platforms, the MySQL Cluster host computer running an SQL node must have installed on it a MySQL Server binary (`mysqld.exe`). You should also have the MySQL client (`mysql.exe`) on this host. For management nodes and data nodes, it is not necessary to install the MySQL Server binary; however, each management node requires the management server daemon (`ndb_mgmd.exe`); each data node requires the data node daemon (`ndbd.exe` or `ndbmtd.exe`). For this example, we refer to `ndbd.exe` as the data node executable, but you can install `ndbmtd.exe`, the multi-threaded version of this program, instead, in exactly the same way. You should also install the management client (`ndb_mgm.exe`) on the management server host. This section covers the steps necessary to install the correct Windows binaries for each type of MySQL Cluster node.

> **Note**
>
> As with other Windows programs, MySQL Cluster executables are named with the `.exe` file extension. However, it is not necessary to include the `.exe` extension when invoking these programs from the command line. Therefore, we often simply refer to these programs in this documentation as `mysqld`, `mysql`, `ndb_mgmd`, and so on. You should understand that, whether we refer (for example) to `mysqld` or `mysqld.exe`, either name means the same thing (the MySQL Server program).

For setting up a MySQL Cluster using Oracles's `no-install` binaries, the first step in the installation process is to download the latest MySQL Cluster Windows binary archive from http://dev.mysql.com/downloads/cluster/. This archive has a filename of the form `mysql-cluster-gpl-noinstall-`*`ver`*`-win`*`arch`*`.zip`, where *`ver`* is the `NDB` storage engine version (such as `7.3.1`), and *`arch`* is the architecture (`32` for 32-bit binaries, and `64` for 64-bit binaries). For example, the MySQL Cluster NDB 7.3.1 `no-install` archive for 32-bit Windows systems is named `mysql-cluster-gpl-noinstall-7.3.1-win32.zip`.

You can run 32-bit MySQL Cluster binaries on both 32-bit and 64-bit versions of Windows; however, 64-bit MySQL Cluster binaries can be used only on 64-bit versions of Windows. If you are using a 32-bit version of Windows on a computer that has a 64-bit CPU, then you must use the 32-bit MySQL Cluster binaries.

To minimize the number of files that need to be downloaded from the Internet or copied between machines, we start with the computer where you intend to run the SQL node.

SQL node. We assume that you have placed a copy of the `no-install` archive in the directory `C:\Documents and Settings\username\My Documents\Downloads` on the computer having the IP address 192.168.0.20, where *`username`* is the name of the current user. (You can obtain this name using `ECHO %USERNAME%` on the command line.) To install and run MySQL Cluster executables as Windows services, this user should be a member of the `Administrators` group.

Extract all the files from the archive. The Extraction Wizard integrated with Windows Explorer is adequate for this task. (If you use a different archive program, be sure that it extracts all files and directories from the archive, and that it preserves the archive's directory structure.) When you are asked for a destination directory, enter `C:\`, which causes the Extraction Wizard to extract the archive to the directory `C:\mysql-cluster-gpl-noinstall-ver-winarch`. Rename this directory to `C:\mysql`.

It is possible to install the MySQL Cluster binaries to directories other than `C:\mysql\bin`; however, if you do so, you must modify the paths shown in this procedure accordingly. In particular, if the MySQL Server (SQL node) binary is installed to a location other than `C:\mysql` or `C:\Program Files\MySQL\MySQL Server 5.6`, or if the SQL node's data directory is in a location other than `C:\mysql\data` or `C:\Program Files\MySQL\MySQL Server 5.6\data`, extra configuration options must be used on the command line or added to the `my.ini` or `my.cnf` file when starting the SQL node. For more information about configuring a MySQL Server to run in a nonstandard location, see Installing MySQL on Microsoft Windows Using a noinstall Zip Archive.

For a MySQL Server with MySQL Cluster support to run as part of a MySQL Cluster, it must be started with the options `--ndbcluster` and `--ndb-connectstring`. While you can specify these options on the command line, it is usually more convenient to place them in an option file. To do this, create a new text file in Notepad or another text editor. Enter the following configuration information into this file:

```
[mysqld]
# Options for mysqld process:
ndbcluster                       # run NDB storage engine
ndb-connectstring=192.168.0.10   # location of management server
```

You can add other options used by this MySQL Server if desired (see Creating an Option File), but the file must contain the options shown, at a minimum. Save this file as `C:\mysql\my.ini`. This completes the installation and setup for the SQL node.

Data nodes. A MySQL Cluster data node on a Windows host requires only a single executable, one of either `ndbd.exe` or `ndbmtd.exe`. For this example, we assume that you are using `ndbd.exe`, but the same instructions apply when using `ndbmtd.exe`. On each computer where you wish to run a data node (the computers having the IP addresses 192.168.0.30 and 192.168.0.40), create the directories `C:\mysql`, `C:\mysql\bin`, and `C:\mysql\cluster-data`; then, on the computer where you downloaded and extracted the `no-install` archive, locate `ndbd.exe` in the `C:\mysql\bin` directory. Copy this file to the `C:\mysql\bin` directory on each of the two data node hosts.

To function as part of a MySQL Cluster, each data node must be given the address or hostname of the management server. You can supply this information on the command line using the `--ndb-connectstring` or `-c` option when starting each data node process. However, it is usually preferable to put this information in an option file. To do this, create a new text file in Notepad or another text editor and enter the following text:

```
[mysql_cluster]
# Options for data node process:
ndb-connectstring=192.168.0.10   # location of management server
```

Save this file as `C:\mysql\my.ini` on the data node host. Create another text file containing the same information and save it on as `C:mysql\my.ini` on the other data node host, or copy the my.ini file from the first data node host to the second one, making sure to place the copy in the second data node's `C:\mysql` directory. Both data node hosts are now ready to be used in the MySQL Cluster, which leaves only the management node to be installed and configured.

Management node. The only executable program required on a computer used for hosting a MySQL Cluster management node is the management server program `ndb_mgmd.exe`. However, in order to administer the MySQL Cluster once it has been started, you should also install the MySQL Cluster management client program `ndb_mgm.exe` on the same machine as the management server. Locate these two programs on the machine where you downloaded and extracted the `no-install` archive; this should be the directory `C:\mysql\bin` on the SQL node host. Create the directory `C:\mysql\bin` on the computer having the IP address 192.168.0.10, then copy both programs to this directory.

You should now create two configuration files for use by `ndb_mgmd.exe`:

1. A local configuration file to supply configuration data specific to the management node itself. Typically, this file needs only to supply the location of the MySQL Cluster global configuration file (see item 2).

   To create this file, start a new text file in Notepad or another text editor, and enter the following information:

   ```
   [mysql_cluster]
   # Options for management node process
   config-file=C:/mysql/bin/config.ini
   ```

   Save this file as the text file `C:\mysql\bin\my.ini`.

2. A global configuration file from which the management node can obtain configuration information governing the MySQL Cluster as a whole. At a minimum, this file must contain a section for each node in the MySQL Cluster, and the IP addresses or hostnames for the management node and all data nodes (`HostName` configuration parameter). It is also advisable to include the following additional information:

   - The IP address or hostname of any SQL nodes
   - The data memory and index memory allocated to each data node (`DataMemory` and `IndexMemory` configuration parameters)
   - The number of replicas, using the `NoOfReplicas` configuration parameter (see Section 3.2, “MySQL Cluster Nodes, Node Groups, Replicas, and Partitions”)
   - The directory where each data node stores it data and log file, and the directory where the management node keeps its log files (in both cases, the `DataDir` configuration parameter)

   Create a new text file using a text editor such as Notepad, and input the following information:

   ```
   [ndbd default]
   # Options affecting ndbd processes on all data nodes:
   NoOfReplicas=2                      # Number of replicas
   DataDir=C:/mysql/cluster-data       # Directory for each data node's data files
                                       # Forward slashes used in directory path,
                                       # rather than backslashes. This is correct;
                                       # see Important note in text
   DataMemory=80M    # Memory allocated to data storage
   IndexMemory=18M   # Memory allocated to index storage
                     # For DataMemory and IndexMemory, we have used the
                     # default values. Since the "world" database takes up
                     # only about 500KB, this should be more than enough for
                     # this example Cluster setup.
   [ndb_mgmd]
   # Management process options:
   HostName=192.168.0.10               # Hostname or IP address of management node
   DataDir=C:/mysql/bin/cluster-logs   # Directory for management node log files
   ```

```
[ndbd]
# Options for data node "A":
                                # (one [ndbd] section per data node)
HostName=192.168.0.30           # Hostname or IP address
[ndbd]
# Options for data node "B":
HostName=192.168.0.40           # Hostname or IP address
[mysqld]
# SQL node options:
HostName=192.168.0.20           # Hostname or IP address
```

Save this file as the text file `C:\mysql\bin\config.ini`.

Important

A single backslash character (`\`) cannot be used when specifying directory paths in program options or configuration files used by MySQL Cluster on Windows. Instead, you must either escape each backslash character with a second backslash (`\\`), or replace the backslash with a forward slash character (`/`). For example, the following line from the `[ndb_mgmd]` section of a MySQL Cluster `config.ini` file does not work:

```
DataDir=C:\mysql\bin\cluster-logs
```

Instead, you may use either of the following:

```
DataDir=C:\\mysql\\bin\\cluster-logs  # Escaped backslashes
```

```
DataDir=C:/mysql/bin/cluster-logs     # Forward slashes
```

For reasons of brevity and legibility, we recommend that you use forward slashes in directory paths used in MySQL Cluster program options and configuration files on Windows.

4.3.2 Compiling and Installing MySQL Cluster from Source on Windows

Oracle provides precompiled MySQL Cluster binaries for Windows which should be adequate for most users. However, if you wish, it is also possible to compile MySQL Cluster for Windows from source code. The procedure for doing this is almost identical to the procedure used to compile the standard MySQL Server binaries for Windows, and uses the same tools. However, there are two major differences:

- To build MySQL Cluster, you must use the MySQL Cluster sources, which you can obtain from http://dev.mysql.com/downloads/cluster/.

  Attempting to build MySQL Cluster from the source code for the standard MySQL Server is likely not to be successful, and is not supported by Oracle.

- You must configure the build using the `WITH_NDBCLUSTER_STORAGE_ENGINE` or `WITH_NDBCLUSTER` option in addition to any other build options you wish to use with `CMake`. (`WITH_NDBCLUSTER` is supported as an alias for `WITH_NDBCLUSTER_STORAGE_ENGINE`, and works in exactly the same way.)

Important

In MySQL Cluster NDB 7.3 and later, the `WITH_NDB_JAVA` option is enabled by default. This means that, by default, if `CMake` cannot find the location of Java on your system, the configuration process fails; if you do not wish to enable Java and ClusterJ support, you must indicate this explicitly by configuring the build using `-DWITH_NDB_JAVA=OFF`. (Bug #12379735) Use `WITH_CLASSPATH` to provide the Java classpath if needed.

For more information about CMake options specific to building MySQL Cluster, see Options for Compiling MySQL Cluster.

Once the build process is complete, you can create a Zip archive containing the compiled binaries; Installing MySQL Using a Standard Source Distribution provides the commands needed to perform this task on Windows systems. The MySQL Cluster binaries can be found in the bin directory of the resulting archive, which is equivalent to the no-install archive, and which can be installed and configured in the same manner. For more information, see Section 4.3.1, “Installing MySQL Cluster on Windows from a Binary Release”.

4.3.3 Initial Startup of MySQL Cluster on Windows

Once the MySQL Cluster executables and needed configuration files are in place, performing an initial start of the cluster is simply a matter of starting the MySQL Cluster executables for all nodes in the cluster. Each cluster node process must be started separately, and on the host computer where it resides. The management node should be started first, followed by the data nodes, and then finally by any SQL nodes.

1. On the management node host, issue the following command from the command line to start the management node process. The output should appear similar to what is shown here:

   ```
   C:\mysql\bin> ndb_mgmd
   2010-06-23 07:53:34 [MgmtSrvr] INFO -- NDB Cluster Management Server. mysql-5.6.27-ndb-7.4.9
   2010-06-23 07:53:34 [MgmtSrvr] INFO -- Reading cluster configuration from 'config.ini'
   ```

   The management node process continues to print logging output to the console. This is normal, because the management node is not running as a Windows service. (If you have used MySQL Cluster on a Unix-like platform such as Linux, you may notice that the management node's default behavior in this regard on Windows is effectively the opposite of its behavior on Unix systems, where it runs by default as a Unix daemon process. This behavior is also true of MySQL Cluster data node processes running on Windows.) For this reason, do not close the window in which ndb_mgmd.exe is running; doing so kills the management node process. (See Section 4.3.4, “Installing MySQL Cluster Processes as Windows Services”, where we show how to install and run MySQL Cluster processes as Windows services.)

   The required -f option tells the management node where to find the global configuration file (config.ini). The long form of this option is --config-file.

   > **Important**
   >
   > A MySQL Cluster management node caches the configuration data that it reads from config.ini; once it has created a configuration cache, it ignores the config.ini file on subsequent starts unless forced to do otherwise. This means that, if the management node fails to start due to an error in this file, you must make the management node re-read config.ini after you have corrected any errors in it. You can do this by starting ndb_mgmd.exe with the --reload or --initial option on the command line. Either of these options works to refresh the configuration cache.
   >
   > It is not necessary or advisable to use either of these options in the management node's my.ini file.

   For additional information about options which can be used with ndb_mgmd, see Section 6.4, “ndb_mgmd — The MySQL Cluster Management Server Daemon”, as well as Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

2. On each of the data node hosts, run the command shown here to start the data node processes:

   ```
   C:\mysql\bin> ndbd
   ```

```
2010-06-23 07:53:46 [ndbd] INFO -- Configuration fetched from 'localhost:1186', generation: 1
```

In each case, the first line of output from the data node process should resemble what is shown in the preceding example, and is followed by additional lines of logging output. As with the management node process, this is normal, because the data node is not running as a Windows service. For this reason, do not close the console window in which the data node process is running; doing so kills `ndbd.exe`. (For more information, see Section 4.3.4, "Installing MySQL Cluster Processes as Windows Services".)

3. Do not start the SQL node yet; it cannot connect to the cluster until the data nodes have finished starting, which may take some time. Instead, in a new console window on the management node host, start the MySQL Cluster management client `ndb_mgm.exe`, which should be in `C:\mysql\bin` on the management node host. (Do not try to re-use the console window where `ndb_mgmd.exe` is running by typing **CTRL**+**C**, as this kills the management node.) The resulting output should look like this:

```
C:\mysql\bin> ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm>
```

When the prompt `ndb_mgm>` appears, this indicates that the management client is ready to receive MySQL Cluster management commands. You can observe the status of the data nodes as they start by entering `ALL STATUS` at the management client prompt. This command causes a running report of the data nodes's startup sequence, which should look something like this:

```
ndb_mgm> ALL STATUS
Connected to Management Server at: localhost:1186
Node 2: starting (Last completed phase 3) (mysql-5.6.27-ndb-7.4.9)
Node 3: starting (Last completed phase 3) (mysql-5.6.27-ndb-7.4.9)
Node 2: starting (Last completed phase 4) (mysql-5.6.27-ndb-7.4.9)
Node 3: starting (Last completed phase 4) (mysql-5.6.27-ndb-7.4.9)
Node 2: Started (version 7.4.9)
Node 3: Started (version 7.4.9)
ndb_mgm>
```

Note

Commands issued in the management client are not case-sensitive; we use uppercase as the canonical form of these commands, but you are not required to observe this convention when inputting them into the `ndb_mgm` client. For more information, see Section 7.2, "Commands in the MySQL Cluster Management Client".

The output produced by `ALL STATUS` is likely to vary from what is shown here, according to the speed at which the data nodes are able to start, the release version number of the MySQL Cluster software you are using, and other factors. What is significant is that, when you see that both data nodes have started, you are ready to start the SQL node.

You can leave `ndb_mgm.exe` running; it has no negative impact on the performance of the MySQL Cluster, and we use it in the next step to verify that the SQL node is connected to the cluster after you have started it.

4. On the computer designated as the SQL node host, open a console window and navigate to the directory where you unpacked the MySQL Cluster binaries (if you are following our example, this is `C:\mysql\bin`).

Start the SQL node by invoking `mysqld.exe` from the command line, as shown here:

```
C:\mysql\bin> mysqld --console
```

The `--console` option causes logging information to be written to the console, which can be helpful in the event of problems. (Once you are satisfied that the SQL node is running in a satisfactory manner, you can stop it and restart it out without the `--console` option, so that logging is performed normally.)

In the console window where the management client (`ndb_mgm.exe`) is running on the management node host, enter the `SHOW` command, which should produce output similar to what is shown here:

```
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.30  (Version: 5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=3    @192.168.0.40  (Version: 5.6.27-ndb-7.4.9, Nodegroup: 0)
[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.10  (Version: 5.6.27-ndb-7.4.9)
[mysqld(API)]   1 node(s)
id=4    @192.168.0.20  (Version: 5.6.27-ndb-7.4.9)
```

You can also verify that the SQL node is connected to the MySQL Cluster in the `mysql` client (`mysql.exe`) using the `SHOW ENGINE NDB STATUS` statement.

You should now be ready to work with database objects and data using MySQL Cluster's `NDBCLUSTER` storage engine. See Section 4.6, “MySQL Cluster Example with Tables and Data”, for more information and examples.

You can also install `ndb_mgmd.exe`, `ndbd.exe`, and `ndbmtd.exe` as Windows services. For information on how to do this, see Section 4.3.4, “Installing MySQL Cluster Processes as Windows Services”).

4.3.4 Installing MySQL Cluster Processes as Windows Services

Once you are satisfied that MySQL Cluster is running as desired, you can install the management nodes and data nodes as Windows services, so that these processes are started and stopped automatically whenever Windows is started or stopped. This also makes it possible to control these processes from the command line with the appropriate `NET START` or `NET STOP` command, or using the Windows graphical `Services` utility.

Installing programs as Windows services usually must be done using an account that has Administrator rights on the system.

To install the management node as a service on Windows, invoke `ndb_mgmd.exe` from the command line on the machine hosting the management node, using the `--install` option, as shown here:

```
C:\> C:\mysql\bin\ndb_mgmd.exe --install
Installing service 'MySQL Cluster Management Server'
  as '"C:\mysql\bin\ndbd.exe" "--service=ndb_mgmd"'
Service successfully installed.
```

> **Important**
>
> When installing a MySQL Cluster program as a Windows service, you should always specify the complete path; otherwise the service installation may fail with the error `The system cannot find the file specified`.

The `--install` option must be used first, ahead of any other options that might be specified for `ndb_mgmd.exe`. However, it is preferable to specify such options in an options file instead. If your options file is not in one of the default locations as shown in the output of `ndb_mgmd.exe --help`, you can specify the location using the `--config-file` option.

Now you should be able to start and stop the management server like this:

```
C:\> NET START ndb_mgmd
The MySQL Cluster Management Server service is starting.
The MySQL Cluster Management Server service was started successfully.
C:\> NET STOP ndb_mgmd
The MySQL Cluster Management Server service is stopping..
The MySQL Cluster Management Server service was stopped successfully.
```

You can also start or stop the management server as a Windows service using the descriptive name, as shown here:

```
C:\> NET START 'MySQL Cluster Management Server'
The MySQL Cluster Management Server service is starting.
The MySQL Cluster Management Server service was started successfully.
C:\> NET STOP  'MySQL Cluster Management Server'
The MySQL Cluster Management Server service is stopping..
The MySQL Cluster Management Server service was stopped successfully.
```

However, it is usually simpler to specify a short service name or to permit the default service name to be used when installing the service, and then reference that name when starting or stopping the service. To specify a service name other than `ndb_mgmd`, append it to the `--install` option, as shown in this example:

```
C:\> C:\mysql\bin\ndb_mgmd.exe --install=mgmd1
Installing service 'MySQL Cluster Management Server'
  as '"C:\mysql\bin\ndb_mgmd.exe" "--service=mgmd1"'
Service successfully installed.
```

Now you should be able to start or stop the service using the name you have specified, like this:

```
C:\> NET START mgmd1
The MySQL Cluster Management Server service is starting.
The MySQL Cluster Management Server service was started successfully.
C:\> NET STOP mgmd1
The MySQL Cluster Management Server service is stopping..
The MySQL Cluster Management Server service was stopped successfully.
```

To remove the management node service, invoke `ndb_mgmd.exe` with the `--remove` option, as shown here:

```
C:\> C:\mysql\bin\ndb_mgmd.exe --remove
Removing service 'MySQL Cluster Management Server'
Service successfully removed.
```

If you installed the service using a service name other than the default, you can remove the service by passing this name as the value of the `--remove` option, like this:

```
C:\> C:\mysql\bin\ndb_mgmd.exe --remove=mgmd1
Removing service 'mgmd1'
Service successfully removed.
```

Installation of a MySQL Cluster data node process as a Windows service can be done in a similar fashion, using the `--install` option for `ndbd.exe` (or `ndbmtd.exe`), as shown here:

```
C:\> C:\mysql\bin\ndbd.exe --install
Installing service 'MySQL Cluster Data Node Daemon' as '"C:\mysql\bin\ndbd.exe" "--service=ndbd"'
Service successfully installed.
```

Now you can start or stop the data node using either the default service name or the descriptive name with `net start` or `net stop`, as shown in the following example:

```
C:\> NET START ndbd
The MySQL Cluster Data Node Daemon service is starting.
The MySQL Cluster Data Node Daemon service was started successfully.
C:\> NET STOP ndbd
The MySQL Cluster Data Node Daemon service is stopping..
The MySQL Cluster Data Node Daemon service was stopped successfully.
C:\> NET START 'MySQL Cluster Data Node Daemon'
The MySQL Cluster Data Node Daemon service is starting.
The MySQL Cluster Data Node Daemon service was started successfully.
C:\> NET STOP 'MySQL Cluster Data Node Daemon'
The MySQL Cluster Data Node Daemon service is stopping..
The MySQL Cluster Data Node Daemon service was stopped successfully.
```

To remove the data node service, invoke `ndbd.exe` with the `--remove` option, as shown here:

```
C:\> C:\mysql\bin\ndbd.exe --remove
Removing service 'MySQL Cluster Data Node Daemon'
Service successfully removed.
```

As with `ndb_mgmd.exe` (and `mysqld.exe`), when installing `ndbd.exe` as a Windows service, you can also specify a name for the service as the value of `--install`, and then use it when starting or stopping the service, like this:

```
C:\> C:\mysql\bin\ndbd.exe --install=dnode1
Installing service 'dnode1' as '"C:\mysql\bin\ndbd.exe" "--service=dnode1"'
Service successfully installed.
C:\> NET START dnode1
The MySQL Cluster Data Node Daemon service is starting.
The MySQL Cluster Data Node Daemon service was started successfully.
C:\> NET STOP dnode1
The MySQL Cluster Data Node Daemon service is stopping..
The MySQL Cluster Data Node Daemon service was stopped successfully.
```

If you specified a service name when installing the data node service, you can use this name when removing it as well, by passing it as the value of the `--remove` option, as shown here:

```
C:\> C:\mysql\bin\ndbd.exe --remove=dnode1
Removing service 'dnode1'
Service successfully removed.
```

Installation of the SQL node as a Windows service, starting the service, stopping the service, and removing the service are done in a similar fashion, using `mysqld --install`, `NET START`, `NET STOP`, and `mysqld --remove`. For additional information, see Starting MySQL as a Windows Service.

4.4 Initial Configuration of MySQL Cluster

In this section, we discuss manual configuration of an installed MySQL Cluster by creating and editing configuration files.

MySQL Cluster (NDB versions 7.3 and later) also provides a GUI installer which can be used to perform the configuration without the need to edit text files in a separate application. For more information, see Section 4.1, “The MySQL Cluster Auto-Installer”.

For our four-node, four-host MySQL Cluster (see Cluster nodes and host computers), it is necessary to write four configuration files, one per node host.

- Each data node or SQL node requires a `my.cnf` file that provides two pieces of information: a *connection string* that tells the node where to find the management node, and a line telling the MySQL server on this host (the machine hosting the data node) to enable the `NDBCLUSTER` storage engine.

  For more information on connection strings, see Section 5.3.3, “MySQL Cluster Connection Strings”.

- The management node needs a `config.ini` file telling it how many replicas to maintain, how much memory to allocate for data and indexes on each data node, where to find the data nodes, where to save data to disk on each data node, and where to find any SQL nodes.

Configuring the data nodes and SQL nodes. The `my.cnf` file needed for the data nodes is fairly simple. The configuration file should be located in the `/etc` directory and can be edited using any text editor. (Create the file if it does not exist.) For example:

```
shell> vi /etc/my.cnf
```

> **Note**
>
> We show `vi` being used here to create the file, but any text editor should work just as well.

For each data node and SQL node in our example setup, `my.cnf` should look like this:

```
[mysqld]
# Options for mysqld process:
ndbcluster                      # run NDB storage engine
[mysql_cluster]
# Options for MySQL Cluster processes:
ndb-connectstring=192.168.0.10  # location of management server
```

After entering the preceding information, save this file and exit the text editor. Do this for the machines hosting data node “A”, data node “B”, and the SQL node.

> **Important**
>
> Once you have started a `mysqld` process with the `ndbcluster` and `ndb-connectstring` parameters in the `[mysqld]` and `[mysql_cluster]` sections of the `my.cnf` file as shown previously, you cannot execute any `CREATE TABLE` or `ALTER TABLE` statements without having actually started the cluster. Otherwise, these statements will fail with an error. This is by design.

Configuring the management node. The first step in configuring the management node is to create the directory in which the configuration file can be found and then to create the file itself. For example (running as `root`):

```
shell> mkdir /var/lib/mysql-cluster
shell> cd /var/lib/mysql-cluster
shell> vi config.ini
```

For our representative setup, the `config.ini` file should read as follows:

```
[ndbd default]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2    # Number of replicas
DataMemory=80M    # How much memory to allocate for data storage
IndexMemory=18M   # How much memory to allocate for index storage
                  # For DataMemory and IndexMemory, we have used the
                  # default values. Since the "world" database takes up
                  # only about 500KB, this should be more than enough for
                  # this example Cluster setup.
[tcp default]
# TCP/IP options:
portnumber=2202   # This the default; however, you can use any
                  # port that is free for all the hosts in the cluster
                  # Note: It is recommended that you do not specify the port
                  # number at all and simply allow the default value to be used
                  # instead
[ndb_mgmd]
# Management process options:
```

```
hostname=192.168.0.10           # Hostname or IP address of MGM node
datadir=/var/lib/mysql-cluster  # Directory for MGM node log files
[ndbd]
# Options for data node "A":
                                # (one [ndbd] section per data node)
hostname=192.168.0.30           # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data files
[ndbd]
# Options for data node "B":
hostname=192.168.0.40           # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data files
[mysqld]
# SQL node options:
hostname=192.168.0.20           # Hostname or IP address
                                # (additional mysqld connections can be
                                # specified for this node for various
                                # purposes such as running ndb_restore)
```

> **Note**
>
> The `world` database can be downloaded from http://dev.mysql.com/doc/, where it can be found listed under "Examples".

After all the configuration files have been created and these minimal options have been specified, you are ready to proceed with starting the cluster and verifying that all processes are running. We discuss how this is done in Section 4.5, "Initial Startup of MySQL Cluster".

For more detailed information about the available MySQL Cluster configuration parameters and their uses, see Section 5.3, "MySQL Cluster Configuration Files", and Chapter 5, *Configuration of MySQL Cluster*. For configuration of MySQL Cluster as relates to making backups, see Section 7.3.3, "Configuration for MySQL Cluster Backups".

> **Note**
>
> The default port for Cluster management nodes is 1186; the default port for data nodes is 2202. However, the cluster can automatically allocate ports for data nodes from those that are already free.

4.5 Initial Startup of MySQL Cluster

Starting the cluster is not very difficult after it has been configured. Each cluster node process must be started separately, and on the host where it resides. The management node should be started first, followed by the data nodes, and then finally by any SQL nodes:

1. On the management host, issue the following command from the system shell to start the management node process:

   ```
   shell> ndb_mgmd -f /var/lib/mysql-cluster/config.ini
   ```

   The first time that it is started, `ndb_mgmd` must be told where to find its configuration file, using the `-f` or `--config-file` option. (See Section 6.4, "`ndb_mgmd` — The MySQL Cluster Management Server Daemon", for details.)

   For additional options which can be used with `ndb_mgmd`, see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

2. On each of the data node hosts, run this command to start the `ndbd` process:

   ```
   shell> ndbd
   ```

3. If you used RPM files to install MySQL on the cluster host where the SQL node is to reside, you can (and should) use the supplied startup script to start the MySQL server process on the SQL node.

If all has gone well, and the cluster has been set up correctly, the cluster should now be operational. You can test this by invoking the ndb_mgm management node client. The output should look like that shown here, although you might see some slight differences in the output depending upon the exact version of MySQL that you are using:

```
shell> ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.30  (Version: 5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=3    @192.168.0.40  (Version: 5.6.27-ndb-7.4.9, Nodegroup: 0)
[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.10  (Version: 5.6.27-ndb-7.4.9)
[mysqld(API)]   1 node(s)
id=4    @192.168.0.20  (Version: 5.6.27-ndb-7.4.9)
```

The SQL node is referenced here as `[mysqld(API)]`, which reflects the fact that the `mysqld` process is acting as a MySQL Cluster API node.

> **Note**
>
> The IP address shown for a given MySQL Cluster SQL or other API node in the output of `SHOW` is the address used by the SQL or API node to connect to the cluster data nodes, and not to any management node.

You should now be ready to work with databases, tables, and data in MySQL Cluster. See Section 4.6, “MySQL Cluster Example with Tables and Data”, for a brief discussion.

4.6 MySQL Cluster Example with Tables and Data

> **Note**
>
> The information in this section applies to MySQL Cluster running on both Unix and Windows platforms.

Working with database tables and data in MySQL Cluster is not much different from doing so in standard MySQL. There are two key points to keep in mind:

- For a table to be replicated in the cluster, it must use the `NDBCLUSTER` storage engine. To specify this, use the `ENGINE=NDBCLUSTER` or `ENGINE=NDB` option when creating the table:

  ```
  CREATE TABLE tbl_name (col_name column_definitions) ENGINE=NDBCLUSTER;
  ```

  Alternatively, for an existing table that uses a different storage engine, use `ALTER TABLE` to change the table to use `NDBCLUSTER`:

  ```
  ALTER TABLE tbl_name ENGINE=NDBCLUSTER;
  ```

- Every `NDBCLUSTER` table has a primary key. If no primary key is defined by the user when a table is created, the `NDBCLUSTER` storage engine automatically generates a hidden one. Such a key takes up space just as does any other table index. (It is not uncommon to encounter problems due to insufficient memory for accommodating these automatically created indexes.)

If you are importing tables from an existing database using the output of `mysqldump`, you can open the SQL script in a text editor and add the `ENGINE` option to any table creation statements, or replace any existing `ENGINE` options. Suppose that you have the `world` sample database on another MySQL server that does not support MySQL Cluster, and you want to export the `City` table:

```
shell> mysqldump --add-drop-table world City > city_table.sql
```

The resulting `city_table.sql` file will contain this table creation statement (and the `INSERT` statements necessary to import the table data):

```
DROP TABLE IF EXISTS `City`;
CREATE TABLE `City` (
  `ID` int(11) NOT NULL auto_increment,
  `Name` char(35) NOT NULL default '',
  `CountryCode` char(3) NOT NULL default '',
  `District` char(20) NOT NULL default '',
  `Population` int(11) NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO `City` VALUES (1,'Kabul','AFG','Kabol',1780000);
INSERT INTO `City` VALUES (2,'Qandahar','AFG','Qandahar',237500);
INSERT INTO `City` VALUES (3,'Herat','AFG','Herat',186800);
(remaining INSERT statements omitted)
```

You need to make sure that MySQL uses the `NDBCLUSTER` storage engine for this table. There are two ways that this can be accomplished. One of these is to modify the table definition *before* importing it into the Cluster database. Using the `City` table as an example, modify the `ENGINE` option of the definition as follows:

```
DROP TABLE IF EXISTS `City`;
CREATE TABLE `City` (
  `ID` int(11) NOT NULL auto_increment,
  `Name` char(35) NOT NULL default '',
  `CountryCode` char(3) NOT NULL default '',
  `District` char(20) NOT NULL default '',
  `Population` int(11) NOT NULL default '0',
  PRIMARY KEY  (`ID`)
) ENGINE=NDBCLUSTER DEFAULT CHARSET=latin1;
INSERT INTO `City` VALUES (1,'Kabul','AFG','Kabol',1780000);
INSERT INTO `City` VALUES (2,'Qandahar','AFG','Qandahar',237500);
INSERT INTO `City` VALUES (3,'Herat','AFG','Herat',186800);
(remaining INSERT statements omitted)
```

This must be done for the definition of each table that is to be part of the clustered database. The easiest way to accomplish this is to do a search-and-replace on the file that contains the definitions and replace all instances of `TYPE=engine_name` or `ENGINE=engine_name` with `ENGINE=NDBCLUSTER`. If you do not want to modify the file, you can use the unmodified file to create the tables, and then use `ALTER TABLE` to change their storage engine. The particulars are given later in this section.

Assuming that you have already created a database named `world` on the SQL node of the cluster, you can then use the `mysql` command-line client to read `city_table.sql`, and create and populate the corresponding table in the usual manner:

```
shell> mysql world < city_table.sql
```

It is very important to keep in mind that the preceding command must be executed on the host where the SQL node is running (in this case, on the machine with the IP address `192.168.0.20`).

To create a copy of the entire `world` database on the SQL node, use `mysqldump` on the noncluster server to export the database to a file named `world.sql`; for example, in the `/tmp` directory. Then modify the table definitions as just described and import the file into the SQL node of the cluster like this:

```
shell> mysql world < /tmp/world.sql
```

If you save the file to a different location, adjust the preceding instructions accordingly.

Running SELECT queries on the SQL node is no different from running them on any other instance of a MySQL server. To run queries from the command line, you first need to log in to the MySQL Monitor in the usual way (specify the root password at the Enter password: prompt):

```
shell> mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.6.27-ndb-7.4.9
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

We simply use the MySQL server's root account and assume that you have followed the standard security precautions for installing a MySQL server, including setting a strong root password. For more information, see Securing the Initial MySQL Accounts.

It is worth taking into account that Cluster nodes do *not* make use of the MySQL privilege system when accessing one another. Setting or changing MySQL user accounts (including the root account) effects only applications that access the SQL node, not interaction between nodes. See Section 7.11.2, “MySQL Cluster and MySQL Privileges”, for more information.

If you did not modify the ENGINE clauses in the table definitions prior to importing the SQL script, you should run the following statements at this point:

```
mysql> USE world;
mysql> ALTER TABLE City ENGINE=NDBCLUSTER;
mysql> ALTER TABLE Country ENGINE=NDBCLUSTER;
mysql> ALTER TABLE CountryLanguage ENGINE=NDBCLUSTER;
```

Selecting a database and running a SELECT query against a table in that database is also accomplished in the usual manner, as is exiting the MySQL Monitor:

```
mysql> USE world;
mysql> SELECT Name, Population FROM City ORDER BY Population DESC LIMIT 5;
+-----------+------------+
| Name      | Population |
+-----------+------------+
| Bombay    |   10500000 |
| Seoul     |    9981619 |
| São Paulo |    9968485 |
| Shanghai  |    9696300 |
| Jakarta   |    9604900 |
+-----------+------------+
5 rows in set (0.34 sec)
mysql> \q
Bye
shell>
```

Applications that use MySQL can employ standard APIs to access NDB tables. It is important to remember that your application must access the SQL node, and not the management or data nodes. This brief example shows how we might execute the SELECT statement just shown by using the PHP 5.X mysqli extension running on a Web server elsewhere on the network:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type"
           content="text/html; charset=iso-8859-1">
  <title>SIMPLE mysqli SELECT</title>
</head>
<body>
<?php
  # connect to SQL node:
  $link = new mysqli('192.168.0.20', 'root', 'root_password', 'world');
  # parameters for mysqli constructor are:
```

```
  #   host, user, password, database
  if( mysqli_connect_errno() )
    die("Connect failed: " . mysqli_connect_error());
  $query = "SELECT Name, Population
            FROM City
            ORDER BY Population DESC
            LIMIT 5";
  # if no errors...
  if( $result = $link->query($query) )
  {
?>
<table border="1" width="40%" cellpadding="4" cellspacing ="1">
  <tbody>
  <tr>
    <th width="10%">City</th>
    <th>Population</th>
  </tr>
<?
    # then display the results...
    while($row = $result->fetch_object())
      printf("<tr>\n  <td align=\"center\">%s</td><td>%d</td>\n</tr>\n",
              $row->Name, $row->Population);
?>
  </tbody
</table>
<?
  # ...and verify the number of rows that were retrieved
    printf("<p>Affected rows: %d</p>\n", $link->affected_rows);
  }
  else
    # otherwise, tell us what went wrong
    echo mysqli_error();
  # free the result set and the mysqli connection object
  $result->close();
  $link->close();
?>
</body>
</html>
```

We assume that the process running on the Web server can reach the IP address of the SQL node.

In a similar fashion, you can use the MySQL C API, Perl-DBI, Python-mysql, or MySQL Connectors to perform the tasks of data definition and manipulation just as you would normally with MySQL.

4.7 Safe Shutdown and Restart of MySQL Cluster

To shut down the cluster, enter the following command in a shell on the machine hosting the management node:

```
shell> ndb_mgm -e shutdown
```

The `-e` option here is used to pass a command to the `ndb_mgm` client from the shell. (See Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”, for more information about this option.) The command causes the `ndb_mgm`, `ndb_mgmd`, and any `ndbd` or `ndbmtd` processes to terminate gracefully. Any SQL nodes can be terminated using `mysqladmin shutdown` and other means. On Windows platforms, assuming that you have installed the SQL node as a Windows service, you can use `NET STOP MYSQL`.

To restart the cluster on Unix platforms, run these commands:

- On the management host (`192.168.0.10` in our example setup):

  ```
  shell> ndb_mgmd -f /var/lib/mysql-cluster/config.ini
  ```

- On each of the data node hosts (`192.168.0.30` and `192.168.0.40`):

```
shell> ndbd
```

- Use the `ndb_mgm` client to verify that both data nodes have started successfully.
- On the SQL host (`192.168.0.20`):

```
shell> mysqld_safe &
```

On Windows platforms, assuming that you have installed all MySQL Cluster processes as Windows services using the default service names (see Section 4.3.4, “Installing MySQL Cluster Processes as Windows Services”), you can restart the cluster as follows:

- On the management host (`192.168.0.10` in our example setup), execute the following command:

```
C:\> NET START ndb_mgmd
```

- On each of the data node hosts (`192.168.0.30` and `192.168.0.40`), execute the following command:

```
C:\> NET START ndbd
```

- On the management node host, use the `ndb_mgm` client to verify that the management node and both data nodes have started successfully (see Section 4.3.3, “Initial Startup of MySQL Cluster on Windows”).
- On the SQL node host (`192.168.0.20`), execute the following command:

```
C:\> NET START mysql
```

In a production setting, it is usually not desirable to shut down the cluster completely. In many cases, even when making configuration changes, or performing upgrades to the cluster hardware or software (or both), which require shutting down individual host machines, it is possible to do so without shutting down the cluster as a whole by performing a *rolling restart* of the cluster. For more information about doing this, see Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”.

4.8 Upgrading and Downgrading MySQL Cluster

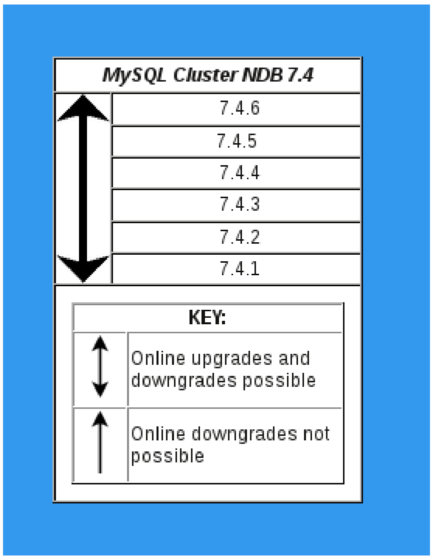
This section provides information about MySQL Cluster software and table file compatibility between different MySQL Cluster NDB 7.3 releases with regard to performing upgrades and downgrades as well as compatibility matrices and notes. You are expected already to be familiar with installing and configuring a MySQL Cluster prior to attempting an upgrade or downgrade. See Chapter 5, *Configuration of MySQL Cluster*.

> **Important**
>
> Only compatibility between MySQL versions with regard to `NDBCLUSTER` is taken into account in this section, and there are likely other issues to be considered. *As with any other MySQL software upgrade or downgrade, you are strongly encouraged to review the relevant portions of the MySQL Manual for the MySQL versions from which and to which you intend to migrate, before attempting an upgrade or downgrade of the MySQL Cluster software*. See Upgrading MySQL.

The tables shown here provide information on MySQL Cluster upgrade and downgrade compatibility among different releases of MySQL Cluster NDB 7.3 and of MySQL Cluster NDB 7.4, respectively. Additional notes about upgrades and downgrades to, from, or within the MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4 release series can be found following the tables.

Upgrades and Downgrades, MySQL Cluster NDB 7.4

Figure 4.19 MySQL Cluster Upgrade and Downgrade Compatibility, MySQL Cluster NDB 7.4



Version support. The following versions of MySQL Cluster are supported for upgrades to MySQL Cluster NDB 7.4 (7.4.4 and later):

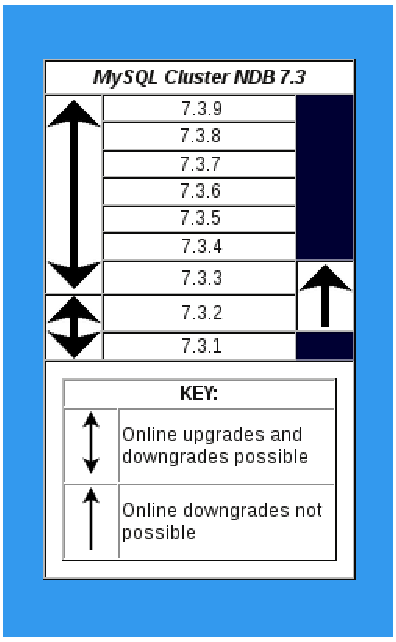
- MySQL Cluster NDB 7.3 GA releases (7.3.2 and later)
- MySQL Cluster NDB 7.2 GA releases (7.2.4 and later)
- MySQL Cluster NDB 7.1 GA releases (7.1.3 and later)
- MySQL Cluster NDB 7.0 GA releases (7.0.5 and later)

Known Issues. Prior to MySQL Cluster NDB 7.4.4, when upgrading from NDB 7.3 to NDB 7.4, the first new data node binary to be started caused the master node (still running NDB 7.3) to fail, then itself failed. (Bug #20608889)

Prior to MySQL Cluster 7.4.3, `mysql_upgrade` failed to drop and recreate `ndbinfo`. (Bug #74863, Bug #20031425) In addition, when running `mysql_upgrade` on a MySQL Cluster SQL node, the expected drop of the `performance_schema` database on this node was instead performed on all SQL nodes connected to the cluster. (Bug #200328691)

Upgrades and Downgrades, MySQL Cluster NDB 7.3

Figure 4.20 MySQL Cluster Upgrade and Downgrade Compatibility, MySQL Cluster NDB 7.3



Version support. The following versions of MySQL Cluster are supported for upgrades to MySQL Cluster NDB 7.3 (7.3.2 and later):

- MySQL Cluster NDB 7.2 GA releases (7.2.4 and later)
- MySQL Cluster NDB 7.1 GA releases (7.1.3 and later)
- MySQL Cluster NDB 7.0 GA releases (7.0.5 and later)
- MySQL Cluster NDB 6.3 GA releases (6.3.8 and later) that can be upgraded to MySQL Cluster NDB 7.1

Known Issues. Prior to MySQL Cluster NDB 7.3.8, `mysql_upgrade` failed to drop and recreate `ndbinfo`. (Bug #74863, Bug #20031425) In addition, when running `mysql_upgrade` on a MySQL Cluster SQL node, the expected drop of the `performance_schema` database on this node was instead performed on all SQL nodes connected to the cluster. (Bug #200328691)

NDB API, ClusterJ, and other applications used with recent releases of MySQL Cluster NDB 6.3 and later should continue to work with MySQL Cluster NDB 7.3.2 and later without rewriting or recompiling.

It is not possible to downgrade online to MySQL Cluster NDB 7.3.2 or earlier from MySQL Cluster NDB 7.3.3 or later. Online upgrades from MySQL Cluster NDB 7.3.2 to later MySQL Cluster NDB 7.3 releases are supported.

For information about upgrades and downgrades in previous MySQL Cluster release series, see http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-upgrade-downgrade-compatibility-6.x.html, http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-upgrade-downgrade-compatibility-7.x.html, and Upgrading and Downgrading MySQL Cluster NDB 7.2

Chapter 5 Configuration of MySQL Cluster

Table of Contents

A MySQL server that is part of a MySQL Cluster differs in one chief respect from a normal (nonclustered) MySQL server, in that it employs the `NDB` storage engine. This engine is also referred to sometimes as `NDBCLUSTER`, although `NDB` is preferred.

To avoid unnecessary allocation of resources, the server is configured by default with the `NDB` storage engine disabled. To enable `NDB`, you must modify the server's `my.cnf` configuration file, or start the server with the `--ndbcluster` option.

This MySQL server is a part of the cluster, so it also must know how to access a management node to obtain the cluster configuration data. The default behavior is to look for the management node on `localhost`. However, should you need to specify that its location is elsewhere, this can be done in `my.cnf`, or with the `mysql` client. Before the `NDB` storage engine can be used, at least one management node must be operational, as well as any desired data nodes.

For more information about `--ndbcluster` and other `mysqld` options specific to MySQL Cluster, see Section 5.3.8.1, “MySQL Server Options for MySQL Cluster”.

In MySQL Cluster NDB 7.3.1 and later, you can use the MySQL Cluster Auto-Installer to set up and deploy a MySQL Cluster on one or more hosts using a browser-based GUI. For more information, see Section 4.1, “The MySQL Cluster Auto-Installer”.

For general information about installing MySQL Cluster, see Chapter 4, *MySQL Cluster Installation*.

5.1 Quick Test Setup of MySQL Cluster

To familiarize you with the basics, we will describe the simplest possible configuration for a functional MySQL Cluster. After this, you should be able to design your desired setup from the information provided in the other relevant sections of this chapter.

First, you need to create a configuration directory such as `/var/lib/mysql-cluster`, by executing the following command as the system `root` user:

```
shell> mkdir /var/lib/mysql-cluster
```

In this directory, create a file named `config.ini` that contains the following information. Substitute appropriate values for `HostName` and `DataDir` as necessary for your system.

```
# file "config.ini" - showing minimal setup consisting of 1 data node,
# 1 management server, and 3 MySQL servers.
# The empty default sections are not required, and are shown only for
# the sake of completeness.
# Data nodes must provide a hostname but MySQL Servers are not required
# to do so.
# If you don't know the hostname for your machine, use localhost.
# The DataDir parameter also has a default value, but it is recommended to
# set it explicitly.
# Note: [db], [api], and [mgm] are aliases for [ndbd], [mysqld], and [ndb_mgmd],
# respectively. [db] is deprecated and should not be used in new installations.
[ndbd default]
NoOfReplicas= 1
[mysqld  default]
[ndb_mgmd default]
[tcp default]
[ndb_mgmd]
HostName= myhost.example.com
[ndbd]
HostName= myhost.example.com
DataDir= /var/lib/mysql-cluster
[mysqld]
[mysqld]
[mysqld]
```

You can now start the `ndb_mgmd` management server. By default, it attempts to read the `config.ini` file in its current working directory, so change location into the directory where the file is located and then invoke `ndb_mgmd`:

```
shell> cd /var/lib/mysql-cluster
shell> ndb_mgmd
```

Then start a single data node by running `ndbd`:

```
shell> ndbd
```

For command-line options which can be used when starting `ndbd`, see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

By default, `ndbd` looks for the management server at `localhost` on port 1186.

> **Note**
>
> If you have installed MySQL from a binary tarball, you will need to specify the path of the `ndb_mgmd` and `ndbd` servers explicitly. (Normally, these will be found in `/usr/local/mysql/bin`.)

Finally, change location to the MySQL data directory (usually `/var/lib/mysql` or `/usr/local/mysql/data`), and make sure that the `my.cnf` file contains the option necessary to enable the NDB storage engine:

```
[mysqld]
ndbcluster
```

You can now start the MySQL server as usual:

```
shell> mysqld_safe --user=mysql &
```

Wait a moment to make sure the MySQL server is running properly. If you see the notice `mysql ended`, check the server's `.err` file to find out what went wrong.

If all has gone well so far, you now can start using the cluster. Connect to the server and verify that the `NDBCLUSTER` storage engine is enabled:

```
shell> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.6.29
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SHOW ENGINES\G
...
*************************** 12. row ***************************
Engine: NDBCLUSTER
Support: YES
Comment: Clustered, fault-tolerant, memory-based tables
*************************** 13. row ***************************
Engine: NDB
Support: YES
Comment: Alias for NDBCLUSTER
...
```

The row numbers shown in the preceding example output may be different from those shown on your system, depending upon how your server is configured.

Try to create an `NDBCLUSTER` table:

```
shell> mysql
mysql> USE test;
Database changed
mysql> CREATE TABLE ctest (i INT) ENGINE=NDBCLUSTER;
Query OK, 0 rows affected (0.09 sec)
mysql> SHOW CREATE TABLE ctest \G
*************************** 1. row ***************************
       Table: ctest
Create Table: CREATE TABLE `ctest` (
  `i` int(11) default NULL
) ENGINE=ndbcluster DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

To check that your nodes were set up properly, start the management client:

```
shell> ndb_mgm
```

Use the `SHOW` command from within the management client to obtain a report on the cluster's status:

```
ndb_mgm> SHOW
Cluster Configuration
---------------------
[ndbd(NDB)]     1 node(s)
id=2    @127.0.0.1  (Version: 5.6.27-ndb-7.4.9, Nodegroup: 0, *)
[ndb_mgmd(MGM)] 1 node(s)
id=1    @127.0.0.1  (Version: 5.6.27-ndb-7.4.9)
[mysqld(API)]   3 node(s)
id=3    @127.0.0.1  (Version: 5.6.27-ndb-7.4.9)
id=4 (not connected, accepting connect from any host)
id=5 (not connected, accepting connect from any host)
```

At this point, you have successfully set up a working MySQL Cluster. You can now store data in the cluster by using any table created with `ENGINE=NDBCLUSTER` or its alias `ENGINE=NDB`.

5.2 Overview of MySQL Cluster Configuration Parameters, Options, and Variables

The next several sections provide summary tables of MySQL Cluster node configuration parameters used in the `config.ini` file to govern various aspects of node behavior, as well as of options and variables read by `mysqld` from a `my.cnf` file or from the command line when run as a MySQL Cluster process. Each of the node parameter tables lists the parameters for a given type (`ndbd`, `ndb_mgmd`, `mysqld`, `computer`, `tcp`, `shm`, or `sci`). All tables include the data type for the parameter, option, or variable, as well as its default, mimimum, and maximum values as applicable.

Considerations when restarting nodes. For node parameters, these tables also indicate what type of restart is required (node restart or system restart)—and whether the restart must be done with `--initial`—to change the value of a given configuration parameter. When performing a node restart or an initial node restart, all of the cluster's data nodes must be restarted in turn (also referred to as a *rolling restart*). It is possible to update cluster configuration parameters marked as `node` online—that is, without shutting down the cluster—in this fashion. An initial node restart requires restarting each `ndbd` process with the `--initial` option.

A system restart requires a complete shutdown and restart of the entire cluster. An initial system restart requires taking a backup of the cluster, wiping the cluster file system after shutdown, and then restoring from the backup following the restart.

In any cluster restart, all of the cluster's management servers must be restarted for them to read the updated configuration parameter values.

> **Important**
>
> Values for numeric cluster parameters can generally be increased without any problems, although it is advisable to do so progressively, making such adjustments in relatively small increments. Many of these can be increased online, using a rolling restart.
>
> However, decreasing the values of such parameters—whether this is done using a node restart, node initial restart, or even a complete system restart of the cluster—is not to be undertaken lightly; it is recommended that you do so only after careful planning and testing. This is especially true with regard to those parameters that relate to memory usage and disk space, such as `MaxNoOfTables`, `MaxNoOfOrderedIndexes`, and `MaxNoOfUniqueHashIndexes`. In addition, it is the generally the case that configuration parameters relating to memory and disk usage can be raised using a simple node restart, but they require an initial node restart to be lowered.

Because some of these parameters can be used for configuring more than one type of cluster node, they may appear in more than one of the tables.

> **Note**
>
> `4294967039` often appears as a maximum value in these tables. This value is defined in the `NDBCLUSTER` sources as `MAX_INT_RNIL` and is equal to `0xFFFFFEFF`, or $2^{32} - 2^{8} - 1$.

5.2.1 MySQL Cluster Data Node Configuration Parameters

The summary table in this section provides information about parameters used in the `[ndbd]` or `[ndbd default]` sections of a `config.ini` file for configuring MySQL Cluster data nodes. For detailed descriptions and other additional information about each of these parameters, see Section 5.3.6, “Defining MySQL Cluster Data Nodes”.

These parameters also apply to `ndbmtd`, the multi-threaded version of `ndbd`. For more information, see Section 6.3, “`ndbmtd` — The MySQL Cluster Data Node Daemon (Multi-Threaded)”.

Restart types. Changes in MySQL Cluster configuration parameters do not take effect until the cluster is restarted. The type of restart required to change a given parameter is indicated in the summary table as follows:

- `N`—Node restart: The parameter can be updated using a rolling restart (see Section 7.5, "Performing a Rolling Restart of a MySQL Cluster").
- `S`—System restart: The cluster must be shut down completely, then restarted, to effect a change in this parameter.
- `I`—Initial restart: Data nodes must be restarted using the `--initial` option.

For more information about restart types, see Section 5.2, "Overview of MySQL Cluster Configuration Parameters, Options, and Variables".

MySQL Cluster NDB 7.3 and later support the addition of new data node groups online, to a running cluster. For more information, see Section 7.13, "Adding MySQL Cluster Data Nodes Online".

Table 5.1 Data Node Configuration Parameters

Parameter Name	Type or Units	Default Value	Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
`Arbitration`	enumeration	Default	Default, Disabled, WaitExternal	N	NDB 7.3.0
`ArbitrationTimeout`	milliseconds	7500	10 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
`BackupDataBufferSize`	bytes	16M	0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
`BackupDataDir`	path	FileSystemPath	...	IN	NDB 7.3.0
`BackupDiskWriteSpeedPct`	percent	50	0 / 90	N	NDB 7.4.8
`BackupLogBufferSize`	bytes	16M	0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
BackupMaxWriteSize	bytes 1M 2K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupMemory	bytes 32M 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupReportFrequency	seconds 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BackupWriteSize	bytes 256K 2K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
BatchSizePerLocalScan	integer 256 1 / 992	N	NDB 7.3.0
BuildIndexThreads	numeric 0 0 / 128	S	NDB 7.3.0
CompressedBackup	boolean false true, false	N	NDB 7.3.0
CompressedLCP	boolean false true, false	N	NDB 7.3.0
ConnectCheckIntervalDelay	milliseconds 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
CrashOnCorruptedTuple	boolean true	S	NDB 7.3.0

Parameter Name	Type or Units	Default Value	Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
			true, false		
DataDir	path	.	...	IN	NDB 7.3.0
DataMemory	bytes	80M	1M / 1024G	N	NDB 7.3.0
DefaultHashMapSize [146]	LDM threads	3840	0 / 3840	N	NDB 7.3.0
DictTrace	bytes	undefined	0 / 100	N	NDB 7.3.0
DiskCheckpointSpeed	bytes	10M	1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
DiskCheckpointSpeedInRestart	bytes	100M	1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
DiskIOThreadPool	threads	2	0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Diskless	true|false (1|0)	false	true, false	IS	NDB 7.3.0
DiskPageBufferEntries	32K pages	10	1 / 1000	N	NDB 7.4.3
DiskPageBufferMemory	bytes	64M	4M / 1T	N	NDB 7.3.0
DiskSyncSize	bytes			N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	4M 32K / 4294967039 (0xFFFFFEFF)		
ExecuteOnComputer	name [none] ...	S	NDB 7.3.0
ExtraSendBufferMemory	bytes 0 0 / 32G	N	NDB 7.3.0
FileSystemPath	path DataDir ...	IN	NDB 7.3.0
FileSystemPathDataFiles	filename [see text] ...	IN	NDB 7.3.0
FileSystemPathDD	filename FileSystemPath ...	IN	NDB 7.3.0
FileSystemPathUndoFiles	filename [see text] ...	IN	NDB 7.3.0
FragmentLogFileSize	bytes 16M 4M / 1G	IN	NDB 7.3.0
HeartbeatIntervalDbApi	milliseconds 1500 100 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
HeartbeatIntervalDbDb	milliseconds 5000 10 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
HeartbeatOrder	numeric 0 0 / 65535	S	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
HostName	name or IP address localhost ...	N	NDB 7.3.0
Id	unsigned [none] 1 / 48	IS	NDB 7.3.0
IndexMemory	bytes 18M 1M / 1T	N	NDB 7.3.0
IndexStatAutoCreate	boolean false false, true	S	NDB 7.3.0
IndexStatAutoUpdate	boolean false false, true	S	NDB 7.3.0
IndexStatSaveScale	percentage 100 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatSaveSize	bytes 32768 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatTriggerPct	percentage 100 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatTriggerScale	percentage 100 0 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
IndexStatUpdateDelay	seconds 60	IN	NDB 7.3.0

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 / 4294967039 (0xFFFFFEFF)		
InitFragmentLogFiles	[see values] SPARSE SPARSE, FULL	IN	NDB 7.3.0
InitialLogFileGroup	string [see text] ...	S	NDB 7.3.0
InitialNoOfOpenFiles	files 27 20 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
InitialTablespace	string [see text] ...	S	NDB 7.3.0
LateAlloc	numeric 1 0 / 1	N	NDB 7.3.0
LcpScanProgressTimeout	second 60 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.3
LockExecuteThreadToCPU	CPU ID 64K 0 / 64K	N	NDB 7.3.0
LockMaintThreadsToCPU	CPU ID [none] 0 / 64K	N	NDB 7.3.0
LockPagesInMainMemory	numeric 0 0 / 2	N	NDB 7.3.0
LogLevelCheckpoint	log level 0 0 / 15	N	NDB 7.3.0
LogLevelCongestion	levelr	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 0 / 15		
LogLevelConnection	integer 0 0 / 15	N	NDB 7.3.0
LogLevelError	integer 0 0 / 15	N	NDB 7.3.0
LogLevelInfo	integer 0 0 / 15	N	NDB 7.3.0
LogLevelNodeRestart	integer 0 0 / 15	N	NDB 7.3.0
LogLevelShutdown	integer 0 0 / 15	N	NDB 7.3.0
LogLevelStartup	integer 1 0 / 15	N	NDB 7.3.0
LogLevelStatistic	integer 0 0 / 15	N	NDB 7.3.0
LongMessageBuffer	bytes 64M 512K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.5
MaxAllocate [146]	unsigned 32M 1M / 1G	N	NDB 7.3.0
MaxBufferedEpochs	epochs 100 0 / 100000	N	NDB 7.3.0
MaxBufferedEpochBytes	bytes 26214400 26214400 (0x01900000) /	N	NDB 7.3.0

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	4294967039 (0xFFFFFEFF)		
MaxDiskWriteSpeed	numeric 20M 1M / 1024G	S	NDB 7.4.1
MaxDiskWriteSpeedOtherNodeRestart	numeric 50M 1M / 1024G	S	NDB 7.4.1
MaxDiskWriteSpeedOwnRestart	numeric 200M 1M / 1024G	S	NDB 7.4.1
MaxDMLOperationsPerTransaction	operations (DML) 4294967295 32 / 4294967295	N	NDB 7.3.0
MaxLCPStartDelay	seconds 0 0 / 600	N	NDB 7.3.0
MaxNoOfAttributes	integer 1000 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentIndexOperations	integer 8K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentOperations	integer 32K 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfConcurrentScans	integer 256 2 / 500	N	NDB 7.3.0
MaxNoOfConcurrentSubOperations	unsigned 256	N	NDB 7.3.0

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 / 4294967039 (0xFFFFFEFF)		
MaxNoOfConcurrentTransactions	integer 4096 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfFiredTriggers	integer 4000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfLocalOperations	integer UNDEFINED 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfLocalScans	integer [see text] 32 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfOpenFiles	unsigned 0 20 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfOrderedIndexes	integer 128 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfSavedMessages	integer 25 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfSubscribers	unsigned 0	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 / 4294967039 (0xFFFFFEFF)		
MaxNoOfSubscriptions	unsigned 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfTables	integer 128 8 / 20320	N	NDB 7.3.0
MaxNoOfTriggers	integer 768 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxNoOfUniqueHashIndexes	integer 64 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxParallelCopyInstances	integer 0 0 / 64	S	NDB 7.4.3
MaxParallelScansPerFragment	bytes 256 1 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MaxStartFailRetries	unsigned 3 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MemReportFrequency	unsigned 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
MinDiskWriteSpeed	numeric	S	NDB 7.4.1

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	10M 1M / 1024G		
MinFreePct	unsigned 5 0 / 100	N	NDB 7.3.0
NodeGroup	 [none] 0 / 65536	IS	NDB 7.3.0
NodeId	unsigned [none] 1 / 48	IS	NDB 7.3.0
NoOfFragmentLogFiles	integer 16 3 / 4294967039 (0xFFFFFEFF)	IN	NDB 7.3.0
NoOfReplicas	integer 2 1 / 4	IS	NDB 7.3.0
Numa	boolean 1 ...	N	NDB 7.3.0
ODirect	boolean false true, false	N	NDB 7.3.0
RealtimeScheduler	boolean false true, false	N	NDB 7.3.0
RedoBuffer	bytes 32M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
RedoOverCommitCounter	numeric 3 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
RedoOverCommitLimit	seconds 20 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ReservedSendBufferMemory	bytes 256K 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
RestartOnErrorInsert	error code 2 0 / 4	N	NDB 7.3.0
SchedulerExecutionTimer	µsec 50 0 / 11000	N	NDB 7.3.0
SchedulerSpinTimer	µsec 0 0 / 500	N	NDB 7.3.0
ServerPort	unsigned [none] 1 / 64K	S	NDB 7.3.0
SharedGlobalMemory	bytes 128M 0 / 64T	N	NDB 7.3.0
StartFailRetryDelay	unsigned 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartFailureTimeout	milliseconds 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartNoNodeGroupTimeout	milliseconds 15000	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 / 4294967039 (0xFFFFFEFF)		
StartPartialTimeout	milliseconds 30000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartPartitionedTimeout	milliseconds 60000 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StartupStatusReportFrequency	seconds 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
StopOnError	boolean 1 0, 1	N	NDB 7.3.0
StringMemory	% or bytes 25 0 / 4294967039 (0xFFFFFEFF)	S	NDB 7.3.0
TcpBind_INADDR_ANY	boolean false true, false	N	NDB 7.3.0
TimeBetweenEpochs	milliseconds 100 0 / 32000	N	NDB 7.3.0
TimeBetweenEpochsTimeout	milliseconds 0 0 / 256000	N	NDB 7.3.0
TimeBetweenGlobalCheckpoints	milliseconds 2000 20 / 32000	N	NDB 7.3.0
TimeBetweenGlobalCheckpointsTimeout	milliseconds	N	NDB 7.3.9

Parameter Name	Type or Units / Default Value / Minimum/Maximum or Permitted Values	Restart Type	In Version ... (and later)
	120000 10 / 4294967039 (0xFFFFFEFF)		
TimeBetweenInactiveTransactionAbortCheck	milliseconds 1000 1000 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TimeBetweenLocalCheckpoints	number of 4-byte words, as a base-2 logarithm 20 0 / 31	N	NDB 7.3.0
TimeBetweenWatchDogCheck	milliseconds 6000 70 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TimeBetweenWatchDogCheckInitial	milliseconds 6000 70 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TotalSendBufferMemory	bytes 0 256K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TransactionBufferMemory	bytes 1M 1K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TransactionDeadlockDetectionTimeout	milliseconds 1200 50 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
TransactionInactiveTimeout	milliseconds	N	NDB 7.3.0

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	[see text] 0 / 4294967039 (0xFFFFFEFF)		
TwoPassInitialNodeRestartCopy	boolean false true, false	N	NDB 7.3.0
UndoDataBuffer	unsigned 16M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
UndoIndexBuffer	unsigned 2M 1M / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0

Table 5.2 Multi-Threaded Data Node Configuration Parameters

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
MaxNoOfExecutionThreads	integer 2 2 / 72	IS	NDB 7.3.3
NoOfFragmentLogParts	numeric 4 4, 8, 12, 16, 24, 32	IN	NDB 7.3.3
ThreadConfig	string '' ...	IS	NDB 7.3.0

5.2.2 MySQL Cluster Management Node Configuration Parameters

The summary table in this section provides information about parameters used in the `[ndb_mgmd]` or `[mgm]` sections of a `config.ini` file for configuring MySQL Cluster management nodes. For detailed descriptions and other additional information about each of these parameters, see Section 5.3.5, “Defining a MySQL Cluster Management Server”.

Restart types. Changes in MySQL Cluster configuration parameters do not take effect until the cluster is restarted. The type of restart required to change a given parameter is indicated in the summary table as follows:

- `N`—Node restart: The parameter can be updated using a rolling restart (see Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”).
- `S`—System restart: The cluster must be shut down completely, then restarted, to effect a change in this parameter.
- `I`—Initial restart: Data nodes must be restarted using the `--initial` option.

For more information about restart types, see Section 5.2, “Overview of MySQL Cluster Configuration Parameters, Options, and Variables”.

Table 5.3 Management Node Configuration Parameters

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
`ArbitrationDelay`	milliseconds 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
`ArbitrationRank`	0-2 1 0 / 2	N	NDB 7.3.0
`DataDir`	path	N	NDB 7.3.0
`ExecuteOnComputer`	name [none] ...	S	NDB 7.3.0
`HeartbeatIntervalMgmdMgmd`	milliseconds 1500 100 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.3
`HeartbeatThreadPriority`	string [none] ...	S	NDB 7.3.0
`HostName`	name or IP address [none] ...	N	NDB 7.3.0
`Id`	unsigned	IS	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	[none] 1 / 255		
LogDestination	{CONSOLE|SYSLOG|FILE} [see text] ...	N	NDB 7.3.0
MaxNoOfSavedEvents	unsigned 100 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
NodeId	unsigned [none] 1 / 255	IS	NDB 7.3.0
PortNumber	unsigned 1186 0 / 64K	S	NDB 7.3.0
PortNumberStats	unsigned [none] 0 / 64K	N	NDB 7.3.0
TotalSendBufferMemory	bytes 0 256K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
wan	boolean false true, false	N	NDB 7.3.0

> **Note**
>
> After making changes in a management node's configuration, it is necessary to perform a rolling restart of the cluster for the new configuration to take effect. See Section 5.3.5, “Defining a MySQL Cluster Management Server”, for more information.
>
> To add new management servers to a running MySQL Cluster, it is also necessary perform a rolling restart of all cluster nodes after modifying any existing `config.ini` files. For more information about issues arising when using multiple management nodes, see Section 3.6.10, “Limitations Relating to Multiple MySQL Cluster Nodes”.

5.2.3 MySQL Cluster SQL Node and API Node Configuration Parameters

The summary table in this section provides information about parameters used in the `[mysqld]` and `[api]` sections of a `config.ini` file for configuring MySQL Cluster SQL nodes and API nodes. For detailed descriptions and other additional information about each of these parameters, see Section 5.3.7, "Defining SQL and Other API Nodes in a MySQL Cluster".

> **Note**
>
> For a discussion of MySQL server options for MySQL Cluster, see Section 5.3.8.1, "MySQL Server Options for MySQL Cluster"; for information about MySQL server system variables relating to MySQL Cluster, see Section 5.3.8.2, "MySQL Cluster System Variables".

Restart types. Changes in MySQL Cluster configuration parameters do not take effect until the cluster is restarted. The type of restart required to change a given parameter is indicated in the summary table as follows:

- `N`—Node restart: The parameter can be updated using a rolling restart (see Section 7.5, "Performing a Rolling Restart of a MySQL Cluster").
- `S`—System restart: The cluster must be shut down completely, then restarted, to effect a change in this parameter.
- `I`—Initial restart: Data nodes must be restarted using the `--initial` option.

For more information about restart types, see Section 5.2, "Overview of MySQL Cluster Configuration Parameters, Options, and Variables".

Table 5.4 SQL Node / API Node Configuration Parameters

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
ArbitrationDelay	milliseconds 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ArbitrationRank	0-2 0 0 / 2	N	NDB 7.3.0
AutoReconnect	boolean false true, false	N	NDB 7.3.0
BatchByteSize	bytes 16K 1024 / 1M	N	NDB 7.3.0
BatchSize	records 256 1 / 992	N	NDB 7.3.0
ConnectBackoffMaxTime	integer 0	N	NDB 7.4.2

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 / 4294967039 (0xFFFFFEFF)		
ConnectionMap	string [none] ...	N	NDB 7.3.0
DefaultHashMapSize	buckets 3840 0 / 3840	N	NDB 7.3.0
DefaultOperationRedoProblemAction	enumeration QUEUE ABORT, QUEUE	S	NDB 7.3.0
EventLogBufferSize	bytes 8192 0 / 64K	S	NDB 7.3.0
ExecuteOnComputer	name [none] ...	S	NDB 7.3.0
ExtraSendBufferMemory	bytes 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
HeartbeatThreadPriority	string [none] ...	S	NDB 7.3.0
HostName	name or IP address [none] ...	N	NDB 7.3.0
Id	unsigned [none] 1 / 255	IS	NDB 7.3.0
MaxScanBatchSize	bytes 256K 32K / 16M	N	NDB 7.3.0
NodeId	unsigned	IS	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	[none] 1 / 255		
StartConnectBackoffMaxTime	integer 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.4.2
TotalSendBufferMemory	bytes 0 256K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
wan	boolean false true, false	N	NDB 7.3.0

Note

To add new SQL or API nodes to the configuration of a running MySQL Cluster, it is necessary to perform a rolling restart of all cluster nodes after adding new `[mysqld]` or `[api]` sections to the `config.ini` file (or files, if you are using more than one management server). This must be done before the new SQL or API nodes can connect to the cluster.

It is *not* necessary to perform any restart of the cluster if new SQL or API nodes can employ previously unused API slots in the cluster configuration to connect to the cluster.

5.2.4 Other MySQL Cluster Configuration Parameters

The summary tables in this section provide information about parameters used in the `[computer]`, `[tcp]`, `[shm]`, and `[sci]` sections of a `config.ini` file for configuring MySQL Cluster management nodes. For detailed descriptions and other additional information about individual parameters, see Section 5.3.9, "MySQL Cluster TCP/IP Connections", Section 5.3.11, "MySQL Cluster Shared-Memory Connections", or Section 5.3.12, "SCI Transport Connections in MySQL Cluster", as appropriate.

Restart types. Changes in MySQL Cluster configuration parameters do not take effect until the cluster is restarted. The type of restart required to change a given parameter is indicated in the summary tables as follows:

- `N`—Node restart: The parameter can be updated using a rolling restart (see Section 7.5, "Performing a Rolling Restart of a MySQL Cluster").
- `S`—System restart: The cluster must be shut down completely, then restarted, to effect a change in this parameter.
- `I`—Initial restart: Data nodes must be restarted using the `--initial` option.

For more information about restart types, see Section 5.2, “Overview of MySQL Cluster Configuration Parameters, Options, and Variables”.

Table 5.5 Computer Configuration Parameters

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
HostName	name or IP address [none] ...	N	NDB 7.3.0
Id	string [none] ...	IS	NDB 7.3.0

Table 5.6 TCP Configuration Parameters

Parameter Name	Type or Units Default Value Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
Checksum	boolean false true, false	N	NDB 7.3.0
Group	unsigned 55 0 / 200	N	NDB 7.3.0
NodeId1	numeric [none] ...	N	NDB 7.3.0
NodeId2	numeric [none] ...	N	NDB 7.3.0
NodeIdServer	numeric [none] ...	N	NDB 7.3.0
OverloadLimit	bytes 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
PortNumber	unsigned	S	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	[none] 0 / 64K		
Proxy	string [none] ...	N	NDB 7.3.0
ReceiveBufferMemory	bytes 2M 16K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
SendBufferMemory	unsigned 2M 256K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
SendSignalId	boolean [see text] true, false	N	NDB 7.3.0
TCP_MAXSEG_SIZE	unsigned 0 0 / 2G	N	NDB 7.3.0
TCP_RCV_BUF_SIZE	unsigned 0 0 / 2G	N	NDB 7.3.1
TCP_SND_BUF_SIZE	unsigned 0 0 / 2G	N	NDB 7.3.1
TcpBind_INADDR_ANY	boolean false true, false	N	NDB 7.3.0

Table 5.7 Shared Memory Configuration Parameters

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
Checksum	boolean	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	true true, false		
Group	unsigned 35 0 / 200	N	NDB 7.3.0
NodeId1	numeric [none] ...	N	NDB 7.3.0
NodeId2	numeric [none] ...	N	NDB 7.3.0
NodeIdServer	numeric [none] ...	N	NDB 7.3.0
OverloadLimit	bytes 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
PortNumber	unsigned [none] 0 / 64K	S	NDB 7.3.0
SendSignalId	boolean false true, false	N	NDB 7.3.0
ShmKey	unsigned [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
ShmSize	bytes 1M 64K / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Signum	unsigned [none]	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
	0 / 4294967039 (0xFFFFFEFF)		

Table 5.8 SCI Configuration Parameters

Parameter Name	Type or Units / Default Value / Minimum/ Maximum or Permitted Values	Restart Type	In Version ... (and later)
Checksum	boolean false true, false	N	NDB 7.3.0
Group	unsigned 15 0 / 200	N	NDB 7.3.0
Host1SciId0	unsigned [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Host1SciId1	unsigned 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Host2SciId0	unsigned [none] 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
Host2SciId1	unsigned 0 0 / 4294967039 (0xFFFFFEFF)	N	NDB 7.3.0
NodeId1	numeric [none] ...	N	NDB 7.3.0
NodeId2	numeric	N	NDB 7.3.0

Parameter Name	Type or Units / Default Value / Minimum/Maximum or Permitted Values	Restart Type	In Version ... (and later)
	[none]		
	...		
NodeIdServer	numeric	N	NDB 7.3.0
	[none]		
	...		
OverloadLimit	bytes	N	NDB 7.3.0
	0		
	0 / 4294967039 (0xFFFFFEFF)		
PortNumber	unsigned	S	NDB 7.3.0
	[none]		
	0 / 64K		
SendLimit	unsigned	N	NDB 7.3.0
	8K		
	128 / 32K		
SendSignalId	boolean	N	NDB 7.3.0
	true		
	true, false		
SharedBufferSize	unsigned	N	NDB 7.3.0
	10M		
	64K / 4294967039 (0xFFFFFEFF)		

5.2.5 MySQL Cluster mysqld Option and Variable Reference

The following table provides a list of the command-line options, server and status variables applicable within `mysqld` when it is running as an SQL node in a MySQL Cluster. For a table showing *all* command-line options, server and status variables available for use with `mysqld`, see Server Option and Variable Reference.

Table 5.9 MySQL Server Options and Variables for MySQL Cluster: MySQL Cluster NDB 7.3-7.4

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Com_show_ndb_status		
No	No	Yes
No	Both	No
DESCRIPTION: Count of SHOW NDB STATUS statements		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
create_old_temporals		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Use pre-5.6.4 storage format for temporal types when creating tables. Intended for use in replication and upgrades/downgrades between NDB 7.2 and NDB 7.3/7.4.		
Handler_discover		
No	No	Yes
No	Both	No
DESCRIPTION: Number of times that tables have been discovered		
have_ndbcluster		
No	Yes	No
No	Global	No
DESCRIPTION: Whether mysqld supports NDB Cluster tables (set by --ndbcluster option)		
ndb-batch-size		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Size (in bytes) to use for NDB transaction batches		
ndb-blob-read-batch-bytes		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Specifies size in bytes that large BLOB reads should be batched into. 0 = no limit.		
ndb-blob-write-batch-bytes		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Specifies size in bytes that large BLOB writes should be batched into. 0 = no limit.		
ndb-cluster-connection-pool		
Yes	Yes	Yes
Yes	Global	No
DESCRIPTION: Number of connections to the cluster used by MySQL		
ndb-connectstring		
Yes	No	No
Yes		No
DESCRIPTION: Point to the management server that distributes the cluster configuration		
ndb-deferred-constraints		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Specifies that constraint checks on unique indexes (where these are supported) should be deferred until commit time. Not normally needed or used; for testing purposes only.		
ndb-distribution		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Default distribution for new tables in NDBCLUSTER (KEYHASH or LINHASH, default is KEYHASH)		
`ndb-log-apply-status`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Cause a MySQL server acting as a slave to log mysql.ndb_apply_status updates received from its immediate master in its own binary log, using its own server ID. Effective only if the server is started with the --ndbcluster option.		
`ndb-log-empty-epochs`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: When enabled, causes epochs in which there were no changes to be written to the ndb_apply_status and ndb_binlog_index tables, even when --log-slave-updates is enabled.		
`ndb-log-exclusive-reads`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Log primary key reads with exclusive locks; allow conflict resolution based on read conflicts.		
`ndb-log-orig`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Log originating server id and epoch in mysql.ndb_binlog_index table.		
`ndb-log-transaction-id`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Write NDB transaction IDs in the binary log. Requires --log-bin-v1-events=OFF.		
`ndb-log-update-as-write [474]`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Toggles logging of updates on the master between updates (OFF) and writes (ON)		
`ndb-mgmd-host`		
Yes	No	No
Yes		No
DESCRIPTION: Set the host (and port, if desired) for connecting to management server		
`ndb-nodeid`		
Yes	No	Yes
Yes	Global	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: MySQL Cluster node ID for this MySQL server		
`ndb-recv-thread-activation-threshold`		
Yes	No	No
Yes		No
DESCRIPTION: Activation threshold when receive thread takes over the polling of the cluster connection (measured in concurrently active threads)		
`ndb-recv-thread-cpu-mask`		
Yes	No	No
Yes		No
DESCRIPTION: CPU mask for locking receiver threads to specific CPUs; specified as hexadecimal. See documentation for details.		
`ndb-transid-mysql-connection-map`		
Yes	No	No
No		No
DESCRIPTION: Enable or disable the ndb_transid_mysql_connection_map plugin; that is, enable or disable the INFORMATION_SCHEMA table having that name.		
`ndb-wait-connected`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Time (in seconds) for the MySQL server to wait for connection to cluster management and data nodes before accepting MySQL client connections.		
`ndb-wait-setup`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Time (in seconds) for the MySQL server to wait for NDB engine setup to complete.		
`Ndb_api_bytes_received_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Amount of data (in bytes) received from the data nodes by this MySQL Server (SQL node).		
`Ndb_api_bytes_received_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Amount of data (in bytes) received from the data nodes in this client session.		
`Ndb_api_bytes_received_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Amount of data (in bytes) received from the data nodes by this slave.		
`Ndb_api_bytes_sent_count`		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
No	No	Yes
No	Global	No
DESCRIPTION: Amount of data (in bytes) sent to the data nodes by this MySQL Server (SQL node).		
`Ndb_api_bytes_sent_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Amount of data (in bytes) sent to the data nodes in this client session.		
`Ndb_api_bytes_sent_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Amount of data (in bytes) sent to the data nodes by this slave.		
`Ndb_api_event_bytes_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of bytes of events received by this MySQL Server (SQL node).		
`Ndb_api_event_bytes_count_injector`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of bytes of events received by the NDB binary log injector thread.		
`Ndb_api_event_data_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of row change events received by this MySQL Server (SQL node).		
`Ndb_api_event_data_count_injector`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of row change events received by the NDB binary log injector thread.		
`Ndb_api_event_nondata_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of events received, other than row change events, by this MySQL Server (SQL node).		
`Ndb_api_event_nondata_count_injector`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of events received, other than row change events, by the NDB binary log injector thread.		
`Ndb_api_pk_op_count`		
No	No	Yes

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
No	Global	No
DESCRIPTION: Number of operations based on or using primary keys by this MySQL Server (SQL node).		
Ndb_api_pk_op_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of operations based on or using primary keys in this client session.		
Ndb_api_pk_op_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of operations based on or using primary keys by this slave.		
Ndb_api_pruned_scan_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of scans that have been pruned to a single partition by this MySQL Server (SQL node).		
Ndb_api_pruned_scan_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of scans that have been pruned to a single partition in this client session.		
Ndb_api_pruned_scan_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of scans that have been pruned to a single partition by this slave.		
Ndb_api_range_scan_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of range scans that have been started by this MySQL Server (SQL node).		
Ndb_api_range_scan_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of range scans that have been started in this client session.		
Ndb_api_range_scan_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of range scans that have been started by this slave.		
Ndb_api_read_row_count		
No	No	Yes
No	Global	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: Total number of rows that have been read by this MySQL Server (SQL node).		
Ndb_api_read_row_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Total number of rows that have been read in this client session.		
Ndb_api_read_row_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Total number of rows that have been read by this slave.		
Ndb_api_scan_batch_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of batches of rows received by this MySQL Server (SQL node).		
Ndb_api_scan_batch_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of batches of rows received in this client session.		
Ndb_api_scan_batch_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of batches of rows received by this slave.		
Ndb_api_table_scan_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of table scans that have been started, including scans of internal tables, by this MySQL Server (SQL node).		
Ndb_api_table_scan_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of table scans that have been started, including scans of internal tables, in this client session.		
Ndb_api_table_scan_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of table scans that have been started, including scans of internal tables, by this slave.		
Ndb_api_trans_abort_count		
No	No	Yes
No	Global	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: Number of transactions aborted by this MySQL Server (SQL node).		
`Ndb_api_trans_abort_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Number of transactions aborted in this client session.		
`Ndb_api_trans_abort_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions aborted by this slave.		
`Ndb_api_trans_close_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions aborted (may be greater than the sum of TransCommitCount and TransAbortCount) by this MySQL Server (SQL node).		
`Ndb_api_trans_close_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Number of transactions aborted (may be greater than the sum of TransCommitCount and TransAbortCount) in this client session.		
`Ndb_api_trans_close_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions aborted (may be greater than the sum of TransCommitCount and TransAbortCount) by this slave.		
`Ndb_api_trans_commit_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions committed by this MySQL Server (SQL node).		
`Ndb_api_trans_commit_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Number of transactions committed in this client session.		
`Ndb_api_trans_commit_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions committed by this slave.		
`Ndb_api_trans_local_read_row_count`		
No	No	Yes
No	Global	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: Total number of rows that have been read by this MySQL Server (SQL node).		
`Ndb_api_trans_local_read_row_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Total number of rows that have been read in this client session.		
`Ndb_api_trans_local_read_row_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Total number of rows that have been read by this slave.		
`Ndb_api_trans_start_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions started by this MySQL Server (SQL node).		
`Ndb_api_trans_start_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Number of transactions started in this client session.		
`Ndb_api_trans_start_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions started by this slave.		
`Ndb_api_uk_op_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of operations based on or using unique keys by this MySQL Server (SQL node).		
`Ndb_api_uk_op_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Number of operations based on or using unique keys in this client session.		
`Ndb_api_uk_op_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of operations based on or using unique keys by this slave.		
`Ndb_api_wait_exec_complete_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times thread has been blocked while waiting for execution of an operation to complete by this MySQL Server (SQL node).		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Ndb_api_wait_exec_complete_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of times thread has been blocked while waiting for execution of an operation to complete in this client session.		
Ndb_api_wait_exec_complete_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times thread has been blocked while waiting for execution of an operation to complete by this slave.		
Ndb_api_wait_meta_request_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times thread has been blocked waiting for a metadata-based signal by this MySQL Server (SQL node).		
Ndb_api_wait_meta_request_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Number of times thread has been blocked waiting for a metadata-based signal in this client session.		
Ndb_api_wait_meta_request_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times thread has been blocked waiting for a metadata-based signal by this slave.		
Ndb_api_wait_nanos_count		
No	No	Yes
No	Global	No
DESCRIPTION: Total time (in nanoseconds) spent waiting for some type of signal from the data nodes by this MySQL Server (SQL node).		
Ndb_api_wait_nanos_count_session		
No	No	Yes
No	Session	No
DESCRIPTION: Total time (in nanoseconds) spent waiting for some type of signal from the data nodes in this client session.		
Ndb_api_wait_nanos_count_slave		
No	No	Yes
No	Global	No
DESCRIPTION: Total time (in nanoseconds) spent waiting for some type of signal from the data nodes by this slave.		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
`Ndb_api_wait_scan_result_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times thread has been blocked while waiting for a scan-based signal by this MySQL Server (SQL node).		
`Ndb_api_wait_scan_result_count_session`		
No	No	Yes
No	Session	No
DESCRIPTION: Number of times thread has been blocked while waiting for a scan-based signal in this client session.		
`Ndb_api_wait_scan_result_count_slave`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times thread has been blocked while waiting for a scan-based signal by this slave.		
`ndb_autoincrement_prefetch_sz`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: NDB auto-increment prefetch size		
`ndb_cache_check_time`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Number of milliseconds between checks of cluster SQL nodes made by the MySQL query cache		
`ndb_clear_apply_status`		
Yes	Yes	No
No	Global	Yes
DESCRIPTION: Causes RESET SLAVE to clear all rows from the ndb_apply_status table. ON by default.		
`Ndb_cluster_node_id`		
No	No	Yes
No	Both	No
DESCRIPTION: If the server is acting as a MySQL Cluster node, then the value of this variable its node ID in the cluster		
`Ndb_config_from_host`		
No	No	Yes
No	Both	No
DESCRIPTION: The host name or IP address of the Cluster management server. Formerly Ndb_connected_host		
`Ndb_config_from_port`		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
No	No	Yes
No	Both	No
DESCRIPTION: The port for connecting to Cluster management server. Formerly Ndb_connected_port		
Ndb_conflict_fn_epoch		
No	No	Yes
No	Global	No
DESCRIPTION: Number of rows that have been found in conflict by the NDB$EPOCH() conflict detection function		
Ndb_conflict_fn_epoch2		
No	No	Yes
No	Global	No
DESCRIPTION: Number of rows that have been found in conflict by the NDB$EPOCH2() conflict detection function		
Ndb_conflict_fn_epoch2_trans		
No	No	Yes
No	Global	No
DESCRIPTION: Number of rows that have been found in conflict by the NDB$EPOCH2_TRANS() conflict detection function		
Ndb_conflict_fn_epoch_trans		
No	No	Yes
No	Global	No
DESCRIPTION: Number of rows that have been found in conflict by the NDB$EPOCH_TRANS() conflict detection function		
Ndb_conflict_fn_max		
No	No	Yes
No	Global	No
DESCRIPTION: If the server is part of a MySQL Cluster involved in cluster replication, the value of this variable indicates the number of times that conflict resolution based on "greater timestamp wins" has been applied		
Ndb_conflict_fn_max_del_win		
No	No	Yes
No	Global	No
DESCRIPTION: Number of times that conflict resolution based on outcome of NDB $MAX_DELETE_WIN() has been applied.		
Ndb_conflict_fn_old		
No	No	Yes
No	Global	No
DESCRIPTION: If the server is part of a MySQL Cluster involved in cluster replication, the value of this variable indicates the number of times that "same timestamp wins" conflict resolution has been applied		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Ndb_conflict_last_conflict_epoch		
No	Yes	No
No	Global	No
DESCRIPTION: Most recent NDB epoch on this slave in which a conflict was detected.		
Ndb_conflict_last_stable_epoch		
No	No	Yes
No	Global	No
DESCRIPTION: Number of rows found to be in conflict by a transactional conflict function		
Ndb_conflict_reflected_op_discard_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of reflected operations that were not applied due an error during execution.		
Ndb_conflict_reflected_op_prepare_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of reflected operations received that have been prepared for execution.		
Ndb_conflict_refresh_op_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of refresh operations that have been prepared.		
Ndb_conflict_trans_conflict_commit_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of epoch transactions committed after requiring transactional conflict handling.		
Ndb_conflict_trans_detect_iter_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of internal iterations required to commit an epoch transaction. Should be (slightly) greater than or equal to Ndb_conflict_trans_conflict_commit_count.		
Ndb_conflict_trans_reject_count		
No	No	Yes
No	Global	No
DESCRIPTION: Number of transactions rejected after being found in conflict by a transactional conflict function.		
Ndb_conflict_trans_row_conflict_count		
No	No	Yes
No	Global	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: Number of rows found in conflict by a transactional conflict function. Includes any rows included in or dependent on conflicting transactions.		
`Ndb_conflict_trans_row_reject_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Total number of rows realigned after being found in conflict by a transactional conflict function. Includes Ndb_conflict_trans_row_conflict_count and any rows included in or dependent on conflicting transactions.		
`ndb_deferred_constraints`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Specifies that constraint checks should be deferred (where these are supported). Not normally needed or used; for testing purposes only.		
`ndb_distribution`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Default distribution for new tables in NDBCLUSTER (KEYHASH or LINHASH, default is KEYHASH)		
`Ndb_conflict_delete_delete_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of delete-delete conflicts detected (delete operation is applied, but row does not exist)		
`ndb_eventbuffer_free_percent`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Percentage of free memory that should be available in event buffer before resumption of buffering, after reaching limit set by ndb_eventbuffer_max_alloc.		
`ndb_eventbuffer_max_alloc`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Maximum memory that can be allocated for buffering events by the NDB API. Defaults to 0 (no limit).		
`Ndb_execute_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Provides the number of round trips to the NDB kernel made by operations		
`ndb_extra_logging`		
Yes	Yes	No
Yes	Global	Yes

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: Controls logging of MySQL Cluster schema, connection, and data distribution events in the MySQL error log		
`ndb_force_send`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Forces sending of buffers to NDB immediately, without waiting for other threads		
`ndb_index_stat_cache_entries`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Sets the granularity of the statistics by determining the number of starting and ending keys		
`ndb_index_stat_enable`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Use NDB index statistics in query optimization		
`ndb_index_stat_option`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Comma-separated list of tunable options for NDB index statistics; the list should contain no spaces		
`ndb_index_stat_update_freq`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: How often to query data nodes instead of the statistics cache		
`ndb_join_pushdown`		
No	Yes	No
No	Both	Yes
DESCRIPTION: Enables pushing down of joins to data nodes		
`ndb_log_apply_status`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Whether or not a MySQL server acting as a slave logs mysql.ndb_apply_status updates received from its immediate master in its own binary log, using its own server ID.		
`ndb_log_bin`		
Yes	Yes	No
No	Both	Yes
DESCRIPTION: Write updates to NDB tables in the binary log. Effective only if binary logging is enabled with --log-bin.		
`ndb_log_binlog_index`		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Yes	Yes	No
No	Global	Yes
DESCRIPTION: Insert mapping between epochs and binary log positions into the ndb_binlog_index table. Defaults to ON. Effective only if binary logging is enabled on the server.		
`ndb_log_empty_epochs`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: When enabled, epochs in which there were no changes are written to the ndb_apply_status and ndb_binlog_index tables, even when log_slave_updates is enabled.		
`ndb_log_exclusive_reads`		
Yes	Yes	No
Yes	Both	Yes
DESCRIPTION: Log primary key reads with exclusive locks; allow conflict resolution based on read conflicts.		
`ndb_log_orig`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Whether the id and epoch of the originating server are recorded in the mysql.ndb_binlog_index table. Set using the --ndb-log-orig option when starting mysqld.		
`ndb_log_transaction_id`		
No	Yes	No
No	Global	No
DESCRIPTION: Whether NDB transaction IDs are written into the binary log. (Read-only.)		
`ndb_log_updated_only [474]`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Log complete rows (ON) or updates only (OFF)		
`Ndb_number_of_data_nodes`		
No	No	Yes
No	Global	No
DESCRIPTION: If the server is part of a MySQL Cluster, the value of this variable is the number of data nodes in the cluster		
`ndb_optimization_delay`		
No	Yes	No
No	Global	Yes
DESCRIPTION: Sets the number of milliseconds to wait between processing sets of rows by OPTIMIZE TABLE on NDB tables.		
`ndb_optimized_node_selection`		
Yes	Yes	No
Yes	Global	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
DESCRIPTION: Determines how an SQL node chooses a cluster data node to use as transaction coordinator		
`Ndb_pruned_scan_count`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of scans executed by NDB since the cluster was last started where partition pruning could be used		
`Ndb_pushed_queries_defined`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of joins that API nodes have attempted to push down to the data nodes		
`Ndb_pushed_queries_dropped`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of joins that API nodes have tried to push down, but failed		
`Ndb_pushed_queries_executed`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of joins successfully pushed down and executed on the data nodes		
`Ndb_pushed_reads`		
No	No	Yes
No	Global	No
DESCRIPTION: Number of reads executed on the data nodes by pushed-down joins		
`ndb_recv_thread_activation_threshold`		
No	No	No
No		No
DESCRIPTION: Activation threshold when receive thread takes over the polling of the cluster connection (measured in concurrently active threads)		
`ndb_recv_thread_cpu_mask`		
No	Yes	No
No	Global	Yes
DESCRIPTION: CPU mask for locking receiver threads to specific CPUs; specified as hexadecimal. See documentation for details.		
`ndb_report_thresh_binlog_epoch_slip`		
Yes	No	No
Yes		No
DESCRIPTION: This is a threshold on the number of epochs to be behind before reporting binary log status		
`ndb_report_thresh_binlog_mem_usage`		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Yes	No	No
Yes		No
DESCRIPTION: This is a threshold on the percentage of free memory remaining before reporting binary log status		
`Ndb_scan_count`		
No	No	Yes
No	Global	No
DESCRIPTION: The total number of scans executed by NDB since the cluster was last started		
`ndb_show_foreign_key_mock_tables`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Show the mock tables used to support foreign_key_checks=0.		
`ndb_slave_conflict_role`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Role for slave to play in conflict detection and resolution. Value is one of PRIMARY, SECONDARY, PASS, or NONE (default). Can be changed only when slave SQL thread is stopped. See documentation for further information.		
`Ndb_slave_max_replicated_epoch`		
No	Yes	No
No	Global	No
DESCRIPTION: The most recently committed NDB epoch on this slave. When this value is greater than or equal to Ndb_conflict_last_conflict_epoch, no conflicts have yet been detected.		
`ndb_table_no_logging`		
No	Yes	No
No	Session	Yes
DESCRIPTION: NDB tables created when this setting is enabled are not checkpointed to disk (although table schema files are created). The setting in effect when the table is created with or altered to use NDBCLUSTER persists for the lifetime of the table.		
`ndb_table_temporary`		
No	Yes	No
No	Session	Yes
DESCRIPTION: NDB tables are not persistent on disk: no schema files are created and the tables are not logged		
`ndb_use_exact_count`		
No	Yes	No
No	Both	Yes
DESCRIPTION: Use exact row count when planning queries		
`ndb_use_transactions`		
Yes	Yes	No

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Yes	Both	Yes
DESCRIPTION: Forces NDB to use a count of records during SELECT COUNT(*) query planning to speed up this type of query		
ndb_version		
No	Yes	No
No	Global	No
DESCRIPTION: Shows build and NDB engine version as an integer.		
ndb_version_string		
No	Yes	No
No	Global	No
DESCRIPTION: Shows build information including NDB engine version in ndb-x.y.z format.		
ndbcluster		
Yes	No	No
Yes		No
DESCRIPTION: Enable NDB Cluster (if this version of MySQL supports it) Disabled by `--skip-ndbcluster`		
ndbinfo_database		
No	Yes	No
No	Global	No
DESCRIPTION: The name used for the NDB information database; read only.		
ndbinfo_max_bytes		
Yes	Yes	No
No	Both	Yes
DESCRIPTION: Used for debugging only.		
ndbinfo_max_rows		
Yes	Yes	No
No	Both	Yes
DESCRIPTION: Used for debugging only.		
ndbinfo_offline		
No	Yes	No
No	Global	Yes
DESCRIPTION: Put the ndbinfo database into offline mode, in which no rows are returned from tables or views.		
ndbinfo_show_hidden		
Yes	Yes	No
No	Both	Yes
DESCRIPTION: Whether to show ndbinfo internal base tables in the mysql client. The default is OFF.		
ndbinfo_table_prefix		

Option or Variable Name		
Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
Notes		
Yes	Yes	No
No	Both	Yes
DESCRIPTION: The prefix to use for naming ndbinfo internal base tables		
`ndbinfo_version`		
No	Yes	No
No	Global	No
DESCRIPTION: The version of the ndbinfo engine; read only.		
`server-id-bits`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: Sets the number of least significant bits in the server_id actually used for identifying the server, permitting NDB API applications to store application data in the most significant bits. server_id must be less than 2 to the power of this value.		
`server_id_bits`		
Yes	Yes	No
Yes	Global	No
DESCRIPTION: The effective value of server_id if the server was started with the --server-id-bits option set to a nondefault value.		
`slave_allow_batching`		
Yes	Yes	No
Yes	Global	Yes
DESCRIPTION: Turns update batching on and off for a replication slave		
`transaction_allow_batching`		
No	Yes	No
No	Session	Yes
DESCRIPTION: Allows batching of statements within a transaction. Disable AUTOCOMMIT to use.		

5.3 MySQL Cluster Configuration Files

Configuring MySQL Cluster requires working with two files:

- `my.cnf`: Specifies options for all MySQL Cluster executables. This file, with which you should be familiar with from previous work with MySQL, must be accessible by each executable running in the cluster.

- `config.ini`: This file, sometimes known as the *global configuration file*, is read only by the MySQL Cluster management server, which then distributes the information contained therein to all processes participating in the cluster. `config.ini` contains a description of each node involved in the cluster. This includes configuration parameters for data nodes and configuration parameters for connections between all nodes in the cluster. For a quick reference to the sections that can appear in this file, and what sorts of configuration parameters may be placed in each section, see Sections of the `config.ini` File.

Caching of configuration data. In MySQL Cluster NDB 7.3 and later, `NDB` uses *stateful configuration*. Rather than reading the global configuration file every time the management server is restarted, the management server caches the configuration the first time it is started, and thereafter, the global configuration file is read only when one of the following conditions is true:

- **The management server is started using the --initial option.** In this case, the global configuration file is re-read, any existing cache files are deleted, and the management server creates a new configuration cache.
- **The management server is started using the --reload option.** In this case, the management server compares its cache with the global configuration file. If they differ, the management server creates a new configuration cache; any existing configuration cache is preserved, but not used. If the management server's cache and the global configuration file contain the same configuration data, then the existing cache is used, and no new cache is created.
- **The management server is started using a --config-cache option.** This option can be used to force the management server to bypass configuration caching altogether. In this case, the management server ignores any configuration files that may be present, always reading its configuration data from the `config.ini` file instead.
- **No configuration cache is found.** In this case, the management server reads the global configuration file and creates a cache containing the same configuration data as found in the file.

Configuration cache files. The management server by default creates configuration cache files in a directory named `mysql-cluster` in the MySQL installation directory. (If you build MySQL Cluster from source on a Unix system, the default location is `/usr/local/mysql-cluster`.) This can be overridden at runtime by starting the management server with the `--configdir` option. Configuration cache files are binary files named according to the pattern `ndb_`*`node_id`*`_config.bin.`*`seq_id`*, where *`node_id`* is the management server's node ID in the cluster, and *`seq_id`* is a cache idenitifer. Cache files are numbered sequentially using *`seq_id`*, in the order in which they are created. The management server uses the latest cache file as determined by the *`seq_id`*.

> **Note**
>
> It is possible to roll back to a previous configuration by deleting later configuration cache files, or by renaming an earlier cache file so that it has a higher *`seq_id`*. However, since configuration cache files are written in a binary format, you should not attempt to edit their contents by hand.

For more information about the `--configdir`, `--config-cache`, `--initial`, and `--reload` options for the MySQL Cluster management server, see Section 6.4, “ndb_mgmd — The MySQL Cluster Management Server Daemon”.

We are continuously making improvements in Cluster configuration and attempting to simplify this process. Although we strive to maintain backward compatibility, there may be times when introduce an incompatible change. In such cases we will try to let Cluster users know in advance if a change is not backward compatible. If you find such a change and we have not documented it, please report it in the MySQL bugs database using the instructions given in How to Report Bugs or Problems.

5.3.1 MySQL Cluster Configuration: Basic Example

To support MySQL Cluster, you will need to update `my.cnf` as shown in the following example. You may also specify these parameters on the command line when invoking the executables.

> **Note**
>
> The options shown here should not be confused with those that are used in `config.ini` global configuration files. Global configuration options are discussed later in this section.

```
# my.cnf
# example additions to my.cnf for MySQL Cluster
# (valid in MySQL 5.6)
```

```
# enable ndbcluster storage engine, and provide connection string for
# management server host (default port is 1186)
[mysqld]
ndbcluster
ndb-connectstring=ndb_mgmd.mysql.com
# provide connection string for management server host (default port: 1186)
[ndbd]
connect-string=ndb_mgmd.mysql.com
# provide connection string for management server host (default port: 1186)
[ndb_mgm]
connect-string=ndb_mgmd.mysql.com
# provide location of cluster configuration file
[ndb_mgmd]
config-file=/etc/config.ini
```

(For more information on connection strings, see Section 5.3.3, “MySQL Cluster Connection Strings”.)

```
# my.cnf
# example additions to my.cnf for MySQL Cluster
# (will work on all versions)
# enable ndbcluster storage engine, and provide connection string for management
# server host to the default port 1186
[mysqld]
ndbcluster
ndb-connectstring=ndb_mgmd.mysql.com:1186
```

> **Important**
>
> Once you have started a `mysqld` process with the `NDBCLUSTER` and `ndb-connectstring` parameters in the `[mysqld]` in the `my.cnf` file as shown previously, you cannot execute any `CREATE TABLE` or `ALTER TABLE` statements without having actually started the cluster. Otherwise, these statements will fail with an error. *This is by design*.

You may also use a separate `[mysql_cluster]` section in the cluster `my.cnf` file for settings to be read and used by all executables:

```
# cluster-specific settings
[mysql_cluster]
ndb-connectstring=ndb_mgmd.mysql.com:1186
```

For additional `NDB` variables that can be set in the `my.cnf` file, see Section 5.3.8.2, “MySQL Cluster System Variables”.

The MySQL Cluster global configuration file is by convention named `config.ini` (but this is not required). If needed, it is read by `ndb_mgmd` at startup and can be placed in any location that can be read by it. The location and name of the configuration are specified using `--config-file=`*`path_name`* with `ndb_mgmd` on the command line. This option has no default value, and is ignored if `ndb_mgmd` uses the configuration cache.

The global configuration file for MySQL Cluster uses INI format, which consists of sections preceded by section headings (surrounded by square brackets), followed by the appropriate parameter names and values. One deviation from the standard INI format is that the parameter name and value can be separated by a colon (“`:`”) as well as the equal sign (“`=`”); however, the equal sign is preferred. Another deviation is that sections are not uniquely identified by section name. Instead, unique sections (such as two different nodes of the same type) are identified by a unique ID specified as a parameter within the section.

Default values are defined for most parameters, and can also be specified in `config.ini`. To create a default value section, simply add the word `default` to the section name. For example, an `[ndbd]` section contains parameters that apply to a particular data node, whereas an `[ndbd default]` section contains parameters that apply to all data nodes. Suppose that all data nodes should use the same data memory size. To configure them all, create an `[ndbd default]` section that contains a `DataMemory` line to specify the data memory size.

> **Note**
>
> In some older releases of MySQL Cluster, there was no default value for `NoOfReplicas`, which always had to be specified explicitly in the `[ndbd default]` section. Although this parameter now has a default value of 2, which is the recommended setting in most common usage scenarios, it is still recommended practice to set this parameter explicitly.

The global configuration file must define the computers and nodes involved in the cluster and on which computers these nodes are located. An example of a simple configuration file for a cluster consisting of one management server, two data nodes and two MySQL servers is shown here:

```
# file "config.ini" - 2 data nodes and 2 SQL nodes
# This file is placed in the startup directory of ndb_mgmd (the
# management server)
# The first MySQL Server can be started from any host. The second
# can be started only on the host mysqld_5.mysql.com
[ndbd default]
NoOfReplicas= 2
DataDir= /var/lib/mysql-cluster
[ndb_mgmd]
Hostname= ndb_mgmd.mysql.com
DataDir= /var/lib/mysql-cluster
[ndbd]
HostName= ndbd_2.mysql.com
[ndbd]
HostName= ndbd_3.mysql.com
[mysqld]
[mysqld]
HostName= mysqld_5.mysql.com
```

> **Note**
>
> The preceding example is intended as a minimal starting configuration for purposes of familiarization with MySQL Cluster, and is almost certain not to be sufficient for production settings. See Section 5.3.2, “Recommended Starting Configuration for MySQL Cluster”, which provides a more complete example starting configuration.

Each node has its own section in the `config.ini` file. For example, this cluster has two data nodes, so the preceding configuration file contains two `[ndbd]` sections defining these nodes.

> **Note**
>
> Do not place comments on the same line as a section heading in the `config.ini` file; this causes the management server not to start because it cannot parse the configuration file in such cases.

Sections of the config.ini File

There are six different sections that you can use in the `config.ini` configuration file, as described in the following list:

- `[computer]`: Defines cluster hosts. This is not required to configure a viable MySQL Cluster, but be may used as a convenience when setting up a large cluster. See Section 5.3.4, “Defining Computers in a MySQL Cluster”, for more information.

- `[ndbd]`: Defines a cluster data node (`ndbd` process). See Section 5.3.6, “Defining MySQL Cluster Data Nodes”, for details.

- `[mysqld]`: Defines the cluster's MySQL server nodes (also called SQL or API nodes). For a discussion of SQL node configuration, see Section 5.3.7, “Defining SQL and Other API Nodes in a MySQL Cluster”.

- `[mgm]` or `[ndb_mgmd]`: Defines a cluster management server (MGM) node. For information concerning the configuration of management nodes, see Section 5.3.5, “Defining a MySQL Cluster Management Server”.
- `[tcp]`: Defines a TCP/IP connection between cluster nodes, with TCP/IP being the default connection protocol. Normally, `[tcp]` or `[tcp default]` sections are not required to set up a MySQL Cluster, as the cluster handles this automatically; however, it may be necessary in some situations to override the defaults provided by the cluster. See Section 5.3.9, “MySQL Cluster TCP/IP Connections”, for information about available TCP/IP configuration parameters and how to use them. (You may also find Section 5.3.10, “MySQL Cluster TCP/IP Connections Using Direct Connections” to be of interest in some cases.)
- `[shm]`: Defines shared-memory connections between nodes. In MySQL 5.6, it is enabled by default, but should still be considered experimental. For a discussion of SHM interconnects, see Section 5.3.11, “MySQL Cluster Shared-Memory Connections”.
- `[sci]`:Defines *Scalable Coherent Interface* connections between cluster data nodes. Such connections require software which, while freely available, is not part of the MySQL Cluster distribution, as well as specialized hardware. See Section 5.3.12, “SCI Transport Connections in MySQL Cluster” for detailed information about SCI interconnects.

You can define `default` values for each section. All Cluster parameter names are case-insensitive, which differs from parameters specified in `my.cnf` or `my.ini` files.

5.3.2 Recommended Starting Configuration for MySQL Cluster

Achieving the best performance from a MySQL Cluster depends on a number of factors including the following:

- MySQL Cluster software version
- Numbers of data nodes and SQL nodes
- Hardware
- Operating system
- Amount of data to be stored
- Size and type of load under which the cluster is to operate

Therefore, obtaining an optimum configuration is likely to be an iterative process, the outcome of which can vary widely with the specifics of each MySQL Cluster deployment. Changes in configuration are also likely to be indicated when changes are made in the platform on which the cluster is run, or in applications that use the MySQL Cluster's data. For these reasons, it is not possible to offer a single configuration that is ideal for all usage scenarios. However, in this section, we provide a recommended base configuration.

Starting config.ini file. The following `config.ini` file is a recommended starting point for configuring a cluster running MySQL Cluster NDB 7.3 or later:

```
# TCP PARAMETERS
[tcp default]
SendBufferMemory=2M
ReceiveBufferMemory=2M
# Increasing the sizes of these 2 buffers beyond the default values
# helps prevent bottlenecks due to slow disk I/O.
# MANAGEMENT NODE PARAMETERS
[ndb_mgmd default]
DataDir=path/to/management/server/data/directory
# It is possible to use a different data directory for each management
# server, but for ease of administration it is preferable to be
# consistent.
[ndb_mgmd]
```

```
HostName=management-server-A-hostname
# NodeId=management-server-A-nodeid
[ndb_mgmd]
HostName=management-server-B-hostname
# NodeId=management-server-B-nodeid
# Using 2 management servers helps guarantee that there is always an
# arbitrator in the event of network partitioning, and so is
# recommended for high availability. Each management server must be
# identified by a HostName. You may for the sake of convenience specify
# a NodeId for any management server, although one will be allocated
# for it automatically; if you do so, it must be in the range 1-255
# inclusive and must be unique among all IDs specified for cluster
# nodes.
# DATA NODE PARAMETERS
[ndbd default]
NoOfReplicas=2
# Using 2 replicas is recommended to guarantee availability of data;
# using only 1 replica does not provide any redundancy, which means
# that the failure of a single data node causes the entire cluster to
# shut down. We do not recommend using more than 2 replicas, since 2 is
# sufficient to provide high availability, and we do not currently test
# with greater values for this parameter.
LockPagesInMainMemory=1
# On Linux and Solaris systems, setting this parameter locks data node
# processes into memory. Doing so prevents them from swapping to disk,
# which can severely degrade cluster performance.
DataMemory=3072M
IndexMemory=384M
# The values provided for DataMemory and IndexMemory assume 4 GB RAM
# per data node. However, for best results, you should first calculate
# the memory that would be used based on the data you actually plan to
# store (you may find the ndb_size.pl utility helpful in estimating
# this), then allow an extra 20% over the calculated values. Naturally,
# you should ensure that each data node host has at least as much
# physical memory as the sum of these two values.
# ODirect=1
# Enabling this parameter causes NDBCLUSTER to try using O_DIRECT
# writes for local checkpoints and redo logs; this can reduce load on
# CPUs. We recommend doing so when using MySQL Cluster on systems running
# Linux kernel 2.6 or later.
NoOfFragmentLogFiles=300
DataDir=path/to/data/node/data/directory
MaxNoOfConcurrentOperations=100000
SchedulerSpinTimer=400
SchedulerExecutionTimer=100
RealTimeScheduler=1
# Setting these parameters allows you to take advantage of real-time scheduling
# of NDB threads to achieve increased throughput when using ndbd. They
# are not needed when using ndbmtd; in particular, you should not set
# RealTimeScheduler for ndbmtd data nodes.
TimeBetweenGlobalCheckpoints=1000
TimeBetweenEpochs=200
DiskCheckpointSpeed=10M
DiskCheckpointSpeedInRestart=100M
RedoBuffer=32M
# CompressedLCP=1
# CompressedBackup=1
# Enabling CompressedLCP and CompressedBackup causes, respectively, local
checkpoint files and backup files to be compressed, which can result in a space
savings of up to 50% over noncompressed LCPs and backups.
# MaxNoOfLocalScans=64
MaxNoOfTables=1024
MaxNoOfOrderedIndexes=256
[ndbd]
HostName=data-node-A-hostname
# NodeId=data-node-A-nodeid
LockExecuteThreadToCPU=1
LockMaintThreadsToCPU=0
# On systems with multiple CPUs, these parameters can be used to lock NDBCLUSTER
# threads to specific CPUs
[ndbd]
HostName=data-node-B-hostname
```

```
# NodeId=data-node-B-nodeid
LockExecuteThreadToCPU=1
LockMaintThreadsToCPU=0
# You must have an [ndbd] section for every data node in the cluster;
# each of these sections must include a HostName. Each section may
# optionally include a NodeId for convenience, but in most cases, it is
# sufficient to allow the cluster to allocate node IDs dynamically. If
# you do specify the node ID for a data node, it must be in the range 1
# to 48 inclusive and must be unique among all IDs specified for
# cluster nodes.
# SQL NODE / API NODE PARAMETERS
[mysqld]
# HostName=sql-node-A-hostname
# NodeId=sql-node-A-nodeid
[mysqld]
[mysqld]
# Each API or SQL node that connects to the cluster requires a [mysqld]
# or [api] section of its own. Each such section defines a connection
# “slot”; you should have at least as many of these sections in the
# config.ini file as the total number of API nodes and SQL nodes that
# you wish to have connected to the cluster at any given time. There is
# no performance or other penalty for having extra slots available in
# case you find later that you want or need more API or SQL nodes to
# connect to the cluster at the same time.
# If no HostName is specified for a given [mysqld] or [api] section,
# then any API or SQL node may use that slot to connect to the
# cluster. You may wish to use an explicit HostName for one connection slot
# to guarantee that an API or SQL node from that host can always
# connect to the cluster. If you wish to prevent API or SQL nodes from
# connecting from other than a desired host or hosts, then use a
# HostName for every [mysqld] or [api] section in the config.ini file.
# You can if you wish define a node ID (NodeId parameter) for any API or
# SQL node, but this is not necessary; if you do so, it must be in the
# range 1 to 255 inclusive and must be unique among all IDs specified
# for cluster nodes.
```

Recommended my.cnf options for SQL nodes. MySQL Servers acting as MySQL Cluster SQL nodes must always be started with the `--ndbcluster` and `--ndb-connectstring` options, either on the command line or in `my.cnf`. In addition, set the following options for all `mysqld` processes in the cluster, unless your setup requires otherwise:

- `--ndb-use-exact-count=0`
- `--ndb-index-stat-enable=0`
- `--ndb-force-send=1`
- `--engine-condition-pushdown=1`

5.3.3 MySQL Cluster Connection Strings

With the exception of the MySQL Cluster management server (`ndb_mgmd`), each node that is part of a MySQL Cluster requires a *connection string* that points to the management server's location. This connection string is used in establishing a connection to the management server as well as in performing other tasks depending on the node's role in the cluster. The syntax for a connection string is as follows:

```
[nodeid=node_id, ]host-definition[, host-definition[, ...]]
host-definition:
    host_name[:port_number]
```

`node_id` is an integer greater than or equal to 1 which identifies a node in `config.ini`. *`host_name`* is a string representing a valid Internet host name or IP address. *`port_number`* is an integer referring to a TCP/IP port number.

```
example 1 (long):    "nodeid=2,myhost1:1100,myhost2:1100,192.168.0.3:1200"
```

```
example 2 (short):   "myhost1"
```

`localhost:1186` is used as the default connection string value if none is provided. If *`port_num`* is omitted from the connection string, the default port is 1186. This port should always be available on the network because it has been assigned by IANA for this purpose (see http://www.iana.org/assignments/port-numbers for details).

By listing multiple host definitions, it is possible to designate several redundant management servers. A MySQL Cluster data or API node attempts to contact successive management servers on each host in the order specified, until a successful connection has been established.

It is also possible to specify in a connection string one or more bind addresses to be used by nodes having multiple network interfaces for connecting to management servers. A bind address consists of a hostname or network address and an optional port number. This enhanced syntax for connection strings is shown here:

```
[nodeid=node_id, ]
    [bind-address=host-definition, ]
    host-definition[; bind-address=host-definition]
    host-definition[; bind-address=host-definition]
    [, ...]]
host-definition:
    host_name[:port_number]
```

If a single bind address is used in the connection string *prior* to specifying any management hosts, then this address is used as the default for connecting to any of them (unless overridden for a given management server; see later in this section for an example). For example, the following connection string causes the node to use `192.168.178.242` regardless of the management server to which it connects:

```
bind-address=192.168.178.242, poseidon:1186, perch:1186
```

If a bind address is specified *following* a management host definition, then it is used only for connecting to that management node. Consider the following connection string:

```
poseidon:1186;bind-address=localhost, perch:1186;bind-address=192.168.178.242
```

In this case, the node uses `localhost` to connect to the management server running on the host named `poseidon` and `192.168.178.242` to connect to the management server running on the host named `perch`.

You can specify a default bind address and then override this default for one or more specific management hosts. In the following example, `localhost` is used for connecting to the management server running on host `poseidon`; since `192.168.178.242` is specified first (before any management server definitions), it is the default bind address and so is used for connecting to the management servers on hosts `perch` and `orca`:

```
bind-address=192.168.178.242,poseidon:1186;bind-address=localhost,perch:1186,orca:2200
```

There are a number of different ways to specify the connection string:

- Each executable has its own command-line option which enables specifying the management server at startup. (See the documentation for the respective executable.)
- It is also possible to set the connection string for all nodes in the cluster at once by placing it in a `[mysql_cluster]` section in the management server's `my.cnf` file.
- For backward compatibility, two other options are available, using the same syntax:
  1. Set the `NDB_CONNECTSTRING` environment variable to contain the connection string.

2. Write the connection string for each executable into a text file named `Ndb.cfg` and place this file in the executable's startup directory.

However, these are now deprecated and should not be used for new installations.

The recommended method for specifying the connection string is to set it on the command line or in the `my.cnf` file for each executable.

5.3.4 Defining Computers in a MySQL Cluster

The `[computer]` section has no real significance other than serving as a way to avoid the need of defining host names for each node in the system. All parameters mentioned here are required.

- `Id`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	[none]	...	IS

This is a unique identifier, used to refer to the host computer elsewhere in the configuration file.

> **Important**
>
> The computer ID is *not* the same as the node ID used for a management, API, or data node. Unlike the case with node IDs, you cannot use `NodeId` in place of `Id` in the `[computer]` section of the `config.ini` file.

- `HostName`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

This is the computer's hostname or IP address.

5.3.5 Defining a MySQL Cluster Management Server

The `[ndb_mgmd]` section is used to configure the behavior of the management server. If multiple management servers are employed, you can specify parameters common to all of them in an `[ndb_mgmd default]` section. `[mgm]` and `[mgm default]` are older aliases for these, supported for backward compatibility.

All parameters in the following list are optional and assume their default values if omitted.

> **Note**
>
> If neither the `ExecuteOnComputer` nor the `HostName` parameter is present, the default value `localhost` will be assumed for both.

- `Id`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 255	IS

Each node in the cluster has a unique identity. For a management node, this is represented by an integer value in the range 1 to 255, inclusive. This ID is used by all internal cluster messages for addressing the node, and so must be unique for each MySQL Cluster node, regardless of the type of node.

> **Note**
>
> Data node IDs must be less than 49. If you plan to deploy a large number of data nodes, it is a good idea to limit the node IDs for management nodes (and API nodes) to values greater than 48.

The use of the `Id` parameter for identifying management nodes is deprecated in favor of `NodeId`. Although `Id` continues to be supported for backward compatibility, it now generates a warning and is subject to removal in a future version of MySQL Cluster.

- `NodeId`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 255	IS

Each node in the cluster has a unique identity. For a management node, this is represented by an integer value in the range 1 to 255 inclusive. This ID is used by all internal cluster messages for addressing the node, and so must be unique for each MySQL Cluster node, regardless of the type of node.

> **Note**
>
> Data node IDs must be less than 49. If you plan to deploy a large number of data nodes, it is a good idea to limit the node IDs for management nodes (and API nodes) to values greater than 48.

`NodeId` is the preferred parameter name to use when identifying management nodes. Although the older `Id` continues to be supported for backward compatibility, it is now deprecated and generates a warning when used; it is also subject to removal in a future MySQL Cluster release.

- `ExecuteOnComputer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name	[none]	...	S

This refers to the `Id` set for one of the computers defined in a `[computer]` section of the `config.ini` file.

- `PortNumber`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	1186	0 - 64K	S

This is the port number on which the management server listens for configuration requests and management commands.

- `HostName`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

Specifying this parameter defines the hostname of the computer on which the management node is to reside. To specify a hostname other than `localhost`, either this parameter or `ExecuteOnComputer` is required.

- `LogDestination`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	{CONSOLE|SYSLOG|FILE}	[see text]	...	N

This parameter specifies where to send cluster logging information. There are three options in this regard—`CONSOLE`, `SYSLOG`, and `FILE`—with `FILE` being the default:

- `CONSOLE` outputs the log to `stdout`:

```
CONSOLE
```

- `SYSLOG` sends the log to a `syslog` facility, possible values being one of `auth`, `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `news`, `syslog`, `user`, `uucp`, `local0`, `local1`, `local2`, `local3`, `local4`, `local5`, `local6`, or `local7`.

> **Note**
>
> Not every facility is necessarily supported by every operating system.

```
SYSLOG:facility=syslog
```

- `FILE` pipes the cluster log output to a regular file on the same machine. The following values can be specified:

  - `filename`: The name of the log file.

    In MySQL Cluster NDB 7.3 and later, the default log file name used in such cases is `ndb_`*`nodeid`*`_cluster.log` (in some older versions, the log file's default name, used if `FILE` was specified without also setting `filename`, was `logger.log`.).

  - `maxsize`: The maximum size (in bytes) to which the file can grow before logging rolls over to a new file. When this occurs, the old log file is renamed by appending `.`*`N`* to the file name, where *`N`* is the next number not yet used with this name.

  - `maxfiles`: The maximum number of log files.

```
FILE:filename=cluster.log,maxsize=1000000,maxfiles=6
```

  The default value for the `FILE` parameter is `FILE:filename=ndb_`*`node_id`*`_cluster.log,maxsize=1000000,maxfiles=6`, where *`node_id`* is the ID of the node.

It is possible to specify multiple log destinations separated by semicolons as shown here:

```
CONSOLE;SYSLOG:facility=local0;FILE:filename=/var/log/mgmd
```

- `ArbitrationRank`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	0-2	1	0 - 2	N

This parameter is used to define which nodes can act as arbitrators. Only management nodes and SQL nodes can be arbitrators. `ArbitrationRank` can take one of the following values:

- `0`: The node will never be used as an arbitrator.

- `1`: The node has high priority; that is, it will be preferred as an arbitrator over low-priority nodes.
- `2`: Indicates a low-priority node which be used as an arbitrator only if a node with a higher priority is not available for that purpose.

Normally, the management server should be configured as an arbitrator by setting its `ArbitrationRank` to 1 (the default for management nodes) and those for all SQL nodes to 0 (the default for SQL nodes).

You can disable arbitration completely either by setting `ArbitrationRank` to 0 on all management and SQL nodes, or by setting the `Arbitration` parameter in the `[ndbd default]` section of the `config.ini` global configuration file. Setting `Arbitration` causes any settings for `ArbitrationRank` to be disregarded.

- `ArbitrationDelay`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	0	0 - 4294967039 (0xFFFFFEFF)	N

An integer value which causes the management server's responses to arbitration requests to be delayed by that number of milliseconds. By default, this value is 0; it is normally not necessary to change it.

- `DataDir`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	path	.	...	N

This specifies the directory where output files from the management server will be placed. These files include cluster log files, process output files, and the daemon's process ID (PID) file. (For log files, this location can be overridden by setting the `FILE` parameter for `LogDestination` as discussed previously in this section.)

The default value for this parameter is the directory in which `ndb_mgmd` is located.

- `PortNumberStats`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	0 - 64K	N

This parameter specifies the port number used to obtain statistical information from a MySQL Cluster management server. It has no default value.

- `Wan`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Use WAN TCP setting as default.

- `HeartbeatThreadPriority`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	[none]	...	S

Set the scheduling policy and priority of heartbeat threads for management and API nodes.

The syntax for setting this parameter is shown here:

```
HeartbeatThreadPriority = policy[, priority]
policy:
  {FIFO | RR}
```

When setting this parameter, you must specify a policy. This is one of `FIFO` (first in, first out) or `RR` (round robin). The policy value is followed optionally by the priority (an integer).

- `TotalSendBufferMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	0	256K - 4294967039 (0xFFFFFEFF)	N

This parameter is used to determine the total amount of memory to allocate on this node for shared send buffer memory among all configured transporters.

If this parameter is set, its minimum permitted value is 256KB; 0 indicates that the parameter has not been set. For more detailed information, see Section 5.3.13, "Configuring MySQL Cluster Send Buffer Parameters".

- `HeartbeatIntervalMgmdMgmd`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.3	milliseconds	1500	100 - 4294967039 (0xFFFFFEFF)	N

Specify the interval between heartbeat messages used to determine whether another management node is on contact with this one. The management node waits after 3 of these intervals to declare the connection dead; thus, the default setting of 1500 milliseconds causes the management node to wait for approximately 1600 ms before timing out.

This parameter was added in MySQL Cluster NDB 7.3.3. (Bug #16426805)

> **Note**
>
> After making changes in a management node's configuration, it is necessary to perform a rolling restart of the cluster for the new configuration to take effect.
>
> To add new management servers to a running MySQL Cluster, it is also necessary to perform a rolling restart of all cluster nodes after modifying any existing `config.ini` files. For more information about issues arising when using multiple management nodes, see Section 3.6.10, "Limitations Relating to Multiple MySQL Cluster Nodes".

5.3.6 Defining MySQL Cluster Data Nodes

The `[ndbd]` and `[ndbd default]` sections are used to configure the behavior of the cluster's data nodes.

`[ndbd]` and `[ndbd default]` are always used as the section names whether you are using `ndbd` or `ndbmtd` binaries for the data node processes.

There are many parameters which control buffer sizes, pool sizes, timeouts, and so forth. The only mandatory parameter is either one of `ExecuteOnComputer` or `HostName`; this must be defined in the local `[ndbd]` section.

The parameter `NoOfReplicas` should be defined in the `[ndbd default]` section, as it is common to all Cluster data nodes. It is not strictly necessary to set `NoOfReplicas`, but it is good practice to set it explicitly.

Most data node parameters are set in the `[ndbd default]` section. Only those parameters explicitly stated as being able to set local values are permitted to be changed in the `[ndbd]` section. Where present, `HostName`, `NodeId` and `ExecuteOnComputer` *must* be defined in the local `[ndbd]` section, and not in any other section of `config.ini`. In other words, settings for these parameters are specific to one data node.

For those parameters affecting memory usage or buffer sizes, it is possible to use `K`, `M`, or `G` as a suffix to indicate units of 1024, 1024×1024, or 1024×1024×1024. (For example, `100K` means 100 × 1024 = 102400.) Parameter names and values are currently case-sensitive.

Information about configuration parameters specific to MySQL Cluster Disk Data tables can be found later in this section (see Disk Data Configuration Parameters).

All of these parameters also apply to `ndbmtd` (the multi-threaded version of `ndbd`). Three additional data node configuration parameters—`MaxNoOfExecutionThreads`, `ThreadConfig`, and `NoOfFragmentLogParts`—apply to `ndbmtd` only; these have no effect when used with `ndbd`. For more information, see Multi-Threading Configuration Parameters (ndbmtd). See also Section 6.3, “`ndbmtd` — The MySQL Cluster Data Node Daemon (Multi-Threaded)”.

Identifying data nodes. The `NodeId` or `Id` value (that is, the data node identifier) can be allocated on the command line when the node is started or in the configuration file.

- `Id`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 48	IS

  A unique node ID is used as the node's address for all cluster internal messages. For data nodes, this is an integer in the range 1 to 48 inclusive. Each node in the cluster must have a unique identifier.

  `NodeId` is the preferred parameter name to use when identifying data nodes. Although the older `Id` is still supported for backward compatibility, it is now deprecated, and generates a warning when used. `Id` is also subject to removal in a future MySQL Cluster release.

- `NodeId`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 48	IS

  A unique node ID is used as the node's address for all cluster internal messages. For data nodes, this is an integer in the range 1 to 48 inclusive. Each node in the cluster must have a unique identifier.

  `NodeId` is the preferred parameter name to use when identifying data nodes. Although `Id` continues to be supported for backward compatibility, it is now deprecated, generates a warning when used, and is subject to removal in a future version of MySQL Cluster.

- `ExecuteOnComputer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name	[none]	...	S

  This refers to the `Id` set for one of the computers defined in a `[computer]` section.

- `HostName`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	localhost	...	N

Specifying this parameter defines the hostname of the computer on which the data node is to reside. To specify a hostname other than `localhost`, either this parameter or `ExecuteOnComputer` is required.

- `ServerPort`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 64K	S

Each node in the cluster uses a port to connect to other nodes. By default, this port is allocated dynamically in such a way as to ensure that no two nodes on the same host computer receive the same port number, so it should normally not be necessary to specify a value for this parameter.

However, if you need to be able to open specific ports in a firewall to permit communication between data nodes and API nodes (including SQL nodes), you can set this parameter to the number of the desired port in an `[ndbd]` section or (if you need to do this for multiple data nodes) the `[ndbd default]` section of the `config.ini` file, and then open the port having that number for incoming connections from SQL nodes, API nodes, or both.

> **Note**
>
> Connections from data nodes to management nodes is done using the `ndb_mgmd` management port (the management server's `PortNumber`; see Section 5.3.5, "Defining a MySQL Cluster Management Server") so outgoing connections to that port from any data nodes should always be permitted.

- `TcpBind_INADDR_ANY`

Setting this parameter to `TRUE` or `1` binds `IP_ADDR_ANY` so that connections can be made from anywhere (for autogenerated connections). The default is `FALSE` (`0`).

- `NodeGroup`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0		[none]	0 - 65536	IS

This parameter can be used to assign a data node to a specific node group. It is read only when the cluster is started for the first time, and cannot be used to reassign a data node to a different node group online. It is generally not desirable to use this parameter in the `[ndbd default]` section of the `config.ini` file, and care must be taken not to assign nodes to node groups in such a way that an invalid numbers of nodes are assigned to any node groups.

The `NodeGroup` parameter is chiefly intended for use in adding a new node group to a running MySQL Cluster without having to perform a rolling restart. For this purpose, you should set it to 65536 (the maximum value). You are not required to set a `NodeGroup` value for all cluster data nodes, only for those nodes which are to be started and added to the cluster as a new node group at a later time. For more information, see Section 7.13.3, "Adding MySQL Cluster Data Nodes Online: Detailed Example".

- `NoOfReplicas`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	2	1 - 4	IS

This global parameter can be set only in the `[ndbd default]` section, and defines the number of replicas for each table stored in the cluster. This parameter also specifies the size of node groups. A node group is a set of nodes all storing the same information.

Node groups are formed implicitly. The first node group is formed by the set of data nodes with the lowest node IDs, the next node group by the set of the next lowest node identities, and so on. By way of example, assume that we have 4 data nodes and that `NoOfReplicas` is set to 2. The four data nodes have node IDs 2, 3, 4 and 5. Then the first node group is formed from nodes 2 and 3, and the second node group by nodes 4 and 5. It is important to configure the cluster in such a manner that nodes in the same node groups are not placed on the same computer because a single hardware failure would cause the entire cluster to fail.

If no node IDs are provided, the order of the data nodes will be the determining factor for the node group. Whether or not explicit assignments are made, they can be viewed in the output of the management client's `SHOW` command.

The default value for `NoOfReplicas` is 2, which is the recommended setting in most common usage scenarios.

The maximum possible value is 4; *currently, only the values 1 and 2 are actually supported*.

> **Important**
>
> Setting `NoOfReplicas` to 1 means that there is only a single copy of all Cluster data; in this case, the loss of a single data node causes the cluster to fail because there are no additional copies of the data stored by that node.

The value for this parameter must divide evenly into the number of data nodes in the cluster. For example, if there are two data nodes, then `NoOfReplicas` must be equal to either 1 or 2, since 2/3 and 2/4 both yield fractional values; if there are four data nodes, then `NoOfReplicas` must be equal to 1, 2, or 4.

- `DataDir`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	path	.	...	IN

This parameter specifies the directory where trace files, log files, pid files and error logs are placed.

The default is the data node process working directory.

- `FileSystemPath`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	path	DataDir	...	IN

This parameter specifies the directory where all files created for metadata, REDO logs, UNDO logs (for Disk Data tables), and data files are placed. The default is the directory specified by `DataDir`.

> **Note**
>
> This directory must exist before the `ndbd` process is initiated.

The recommended directory hierarchy for MySQL Cluster includes `/var/lib/mysql-cluster`, under which a directory for the node's file system is created. The name of this subdirectory contains the node ID. For example, if the node ID is 2, this subdirectory is named `ndb_2_fs`.

- `BackupDataDir`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	path	[see text]	...	IN

This parameter specifies the directory in which backups are placed.

> **Important**
>
> The string '/BACKUP' is always appended to this value. For example, if you set the value of BackupDataDir to /var/lib/cluster-data, then all backups are stored under /var/lib/cluster-data/BACKUP. This also means that the *effective* default backup location is the directory named BACKUP under the location specified by the FileSystemPath parameter.

Data Memory, Index Memory, and String Memory

DataMemory and IndexMemory are [ndbd] parameters specifying the size of memory segments used to store the actual records and their indexes. In setting values for these, it is important to understand how DataMemory and IndexMemory are used, as they usually need to be updated to reflect actual usage by the cluster:

- DataMemory

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	80M	1M - 1024G	N

This parameter defines the amount of space (in bytes) available for storing database records. The entire amount specified by this value is allocated in memory, so it is extremely important that the machine has sufficient physical memory to accommodate it.

The memory allocated by DataMemory is used to store both the actual records and indexes. There is a 16-byte overhead on each record; an additional amount for each record is incurred because it is stored in a 32KB page with 128 byte page overhead (see below). There is also a small amount wasted per page due to the fact that each record is stored in only one page.

For variable-size table attributes, the data is stored on separate data pages, allocated from DataMemory. Variable-length records use a fixed-size part with an extra overhead of 4 bytes to reference the variable-size part. The variable-size part has 2 bytes overhead plus 2 bytes per attribute.

The maximum record size is 14000 bytes.

The memory space defined by DataMemory is also used to store ordered indexes, which use about 10 bytes per record. Each table row is represented in the ordered index. A common error among users is to assume that all indexes are stored in the memory allocated by IndexMemory, but this is not the case: Only primary key and unique hash indexes use this memory; ordered indexes use the memory allocated by DataMemory. However, creating a primary key or unique hash index also creates an ordered index on the same keys, unless you specify USING HASH in the index creation statement. This can be verified by running ndb_desc -d *db_name table_name* in the management client.

MySQL Cluster can use a maximum of 512 MB for hash indexes per partition, which means in some cases it is possible to get Table is full errors in MySQL client applications even when ndb_mgm -e "ALL REPORT MEMORYUSAGE" shows significant free DataMemory. This can also pose a problem with data node restarts on nodes that are heavily loaded with data. You can force NDB to create extra partitions for MySQL Cluster tables and thus have more memory available for hash indexes by using the MAX_ROWS option for CREATE TABLE. In general, setting MAX_ROWS to twice the number of rows that you expect to store in the table should be sufficient. You can also use the MinFreePct configuration parameter to help avoid problems with node restarts. (Bug #13436216)

The memory space allocated by DataMemory consists of 32KB pages, which are allocated to table fragments. Each table is normally partitioned into the same number of fragments as there are data nodes in the cluster. Thus, for each node, there are the same number of fragments as are set in NoOfReplicas.

Once a page has been allocated, it is currently not possible to return it to the pool of free pages, except by deleting the table. (This also means that DataMemory pages, once allocated to a given table, cannot be used by other tables.) Performing a data node recovery also compresses the partition because all records are inserted into empty partitions from other live nodes.

The DataMemory memory space also contains UNDO information: For each update, a copy of the unaltered record is allocated in the DataMemory. There is also a reference to each copy in the ordered table indexes. Unique hash indexes are updated only when the unique index columns are updated, in which case a new entry in the index table is inserted and the old entry is deleted upon commit. For this reason, it is also necessary to allocate enough memory to handle the largest transactions performed by applications using the cluster. In any case, performing a few large transactions holds no advantage over using many smaller ones, for the following reasons:

- Large transactions are not any faster than smaller ones
- Large transactions increase the number of operations that are lost and must be repeated in event of transaction failure
- Large transactions use more memory

The default value for DataMemory is 80MB; the minimum is 1MB. There is no maximum size, but in reality the maximum size has to be adapted so that the process does not start swapping when the limit is reached. This limit is determined by the amount of physical RAM available on the machine and by the amount of memory that the operating system may commit to any one process. 32-bit operating systems are generally limited to 2−4GB per process; 64-bit operating systems can use more. For large databases, it may be preferable to use a 64-bit operating system for this reason.

- IndexMemory

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	18M	1M - 1T	N

This parameter controls the amount of storage used for hash indexes in MySQL Cluster. Hash indexes are always used for primary key indexes, unique indexes, and unique constraints. When defining a primary key or a unique index, two indexes are created, one of which is a hash index used for all tuple accesses as well as lock handling. This index is also used to enforce unique constraints.

You can estimate the size of a hash index using this formula:

```
            size  = ( (fragments * 32K) + (rows * 18) )
            * replicas
```

fragments is the number of fragments, *replicas* is the number of replicas (normally 2), and *rows* is the number of rows. If a table has one million rows, 8 fragments, and 2 replicas, the expected index memory usage is calculated as shown here:

```
((8 * 32K) + (1000000 * 18)) * 2 = ((8 * 32768) + (1000000 * 18)) * 2
= (262144 + 18000000) * 2
= 18262144 * 2 = 36524288 bytes = ~35MB
```

In MySQL Cluster NDB 7.2 and later, index statistics (when enabled) for ordered indexes are stored in the mysql.ndb_index_stat_sample table. Since this table has a hash index, this adds to index memory usage. An upper bound to the number of rows for a given ordered index can be calculated as follows:

```
sample_size= key_size + ((key_attributes + 1) * 4)

sample_rows = IndexStatSaveSize
              * ((0.01 * IndexStatSaveScale * log2(rows * sample_size)) + 1)
              / sample_size
```

In the preceding formula, *key_size* is the size of the ordered index key in bytes, *key_attributes* is the number ot attributes in the ordered index key, and *rows* is the number of rows in the base table.

Assume that table t1 has 1 million rows and an ordered index named ix1 on two four-byte integers. Assume in addition that IndexStatSaveSize and IndexStatSaveScale are set to their default values (32K and 100, respectively). Using the previous 2 formulas, we can calculate as follows:

```
sample_size = 8  + ((1 + 2) * 4) = 20 bytes

sample_rows = 32K
              * ((0.01 * 100 * log2(1000000*20)) + 1)
              / 20
              = 32768 * ( (1 * ~16.811) +1) / 20
              = 32768 * ~17.811 / 20
              = ~29182 rows
```

The expected index memory usage is thus 2 * 18 * 29182 = ~1050550 bytes.

The default value for IndexMemory is 18MB. The minimum is 1MB.

- StringMemory

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	% or bytes	25	0 - 4294967039 (0xFFFFFEFF)	S

This parameter determines how much memory is allocated for strings such as table names, and is specified in an [ndbd] or [ndbd default] section of the config.ini file. A value between 0 and 100 inclusive is interpreted as a percent of the maximum default value, which is calculated based on a number of factors including the number of tables, maximum table name size, maximum size of .FRM files, MaxNoOfTriggers, maximum column name size, and maximum default column value.

A value greater than 100 is interpreted as a number of bytes.

The default value is 25—that is, 25 percent of the default maximum.

Under most circumstances, the default value should be sufficient, but when you have a great many Cluster tables (1000 or more), it is possible to get Error 773 Out of string memory, please modify StringMemory config parameter: Permanent error: Schema error, in which case you should increase this value. 25 (25 percent) is not excessive, and should prevent this error from recurring in all but the most extreme conditions.

The following example illustrates how memory is used for a table. Consider this table definition:

```
CREATE TABLE example (
  a INT NOT NULL,
  b INT NOT NULL,
  c INT NOT NULL,
  PRIMARY KEY(a),
  UNIQUE(b)
```

```
) ENGINE=NDBCLUSTER;
```

For each record, there are 12 bytes of data plus 12 bytes overhead. Having no nullable columns saves 4 bytes of overhead. In addition, we have two ordered indexes on columns `a` and `b` consuming roughly 10 bytes each per record. There is a primary key hash index on the base table using roughly 29 bytes per record. The unique constraint is implemented by a separate table with `b` as primary key and `a` as a column. This other table consumes an additional 29 bytes of index memory per record in the `example` table as well 8 bytes of record data plus 12 bytes of overhead.

Thus, for one million records, we need 58MB for index memory to handle the hash indexes for the primary key and the unique constraint. We also need 64MB for the records of the base table and the unique index table, plus the two ordered index tables.

You can see that hash indexes takes up a fair amount of memory space; however, they provide very fast access to the data in return. They are also used in MySQL Cluster to handle uniqueness constraints.

The only partitioning algorithm is hashing and ordered indexes are local to each node. Thus, ordered indexes cannot be used to handle uniqueness constraints in the general case.

An important point for both `IndexMemory` and `DataMemory` is that the total database size is the sum of all data memory and all index memory for each node group. Each node group is used to store replicated information, so if there are four nodes with two replicas, there will be two node groups. Thus, the total data memory available is 2 × `DataMemory` for each data node.

It is highly recommended that `DataMemory` and `IndexMemory` be set to the same values for all nodes. Data distribution is even over all nodes in the cluster, so the maximum amount of space available for any node can be no greater than that of the smallest node in the cluster.

`DataMemory` and `IndexMemory` can be changed, but decreasing either of these can be risky; doing so can easily lead to a node or even an entire MySQL Cluster that is unable to restart due to there being insufficient memory space. Increasing these values should be acceptable, but it is recommended that such upgrades are performed in the same manner as a software upgrade, beginning with an update of the configuration file, and then restarting the management server followed by restarting each data node in turn.

MinFreePct. A proportion (5% by default) of data node resources including `DataMemory` and `IndexMemory` is kept in reserve to insure that the data node does not exhaust its memory when performing a restart. This can be adjusted using the `MinFreePct` data node configuration parameter (default 5).

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	5	0 - 100	N

Updates do not increase the amount of index memory used. Inserts take effect immediately; however, rows are not actually deleted until the transaction is committed.

Transaction parameters. The next few `[ndbd]` parameters that we discuss are important because they affect the number of parallel transactions and the sizes of transactions that can be handled by the system. `MaxNoOfConcurrentTransactions` sets the number of parallel transactions possible in a node. `MaxNoOfConcurrentOperations` sets the number of records that can be in update phase or locked simultaneously.

Both of these parameters (especially `MaxNoOfConcurrentOperations`) are likely targets for users setting specific values and not using the default value. The default value is set for systems using small transactions, to ensure that these do not use excessive memory.

`MaxDMLOperationsPerTransaction` sets the maximum number of DML operations that can be performed in a given transaction.

- `MaxNoOfConcurrentTransactions`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	4096	32 - 4294967039 (0xFFFFFEFF)	N

Each cluster data node requires a transaction record for each active transaction in the cluster. The task of coordinating transactions is distributed among all of the data nodes. The total number of transaction records in the cluster is the number of transactions in any given node times the number of nodes in the cluster.

Transaction records are allocated to individual MySQL servers. Each connection to a MySQL server requires at least one transaction record, plus an additional transaction object per table accessed by that connection. This means that a reasonable minimum for the total number of transactions in the cluster can be expressed as

```
MinTotalNoOfConcurrentTransactions =
    (maximum number of tables accessed in any single transaction + 1)
    * number of SQL nodes
```

Suppose that there are 10 SQL nodes using the cluster. A single join involving 10 tables requires 11 transaction records; if there are 10 such joins in a transaction, then 10 * 11 = 110 transaction records are required for this transaction, per MySQL server, or 110 * 10 = 1100 transaction records total. Each data node can be expected to handle MinTotalNoOfConcurrentTransactions / number of data nodes. For a MySQL Cluster having 4 data nodes, this would mean setting `MaxNoOfConcurrentTransactions` on each data node to 1100 / 4 = 275. In addition, you should provide for failure recovery by ensuring that a single node group can accommodate all concurrent transactions; in other words, that each data node's MaxNoOfConcurrentTransactions is sufficient to cover a number of transactions equal to MinTotalNoOfConcurrentTransactions / number of node groups. If this cluster has a single node group, then `MaxNoOfConcurrentTransactions` should be set to 1100 (the same as the total number of concurrent transactions for the entire cluster).

In addition, each transaction involves at least one operation; for this reason, the value set for `MaxNoOfConcurrentTransactions` should always be no more than the value of `MaxNoOfConcurrentOperations`.

This parameter must be set to the same value for all cluster data nodes. This is due to the fact that, when a data node fails, the oldest surviving node re-creates the transaction state of all transactions that were ongoing in the failed node.

It is possible to change this value using a rolling restart, but the amount of traffic on the cluster must be such that no more transactions occur than the lower of the old and new levels while this is taking place.

The default value is 4096.

- `MaxNoOfConcurrentOperations`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	32K	32 - 4294967039 (0xFFFFFEFF)	N

It is a good idea to adjust the value of this parameter according to the size and number of transactions. When performing transactions which involve only a few operations and records, the default value for this parameter is usually sufficient. Performing large transactions involving many records usually requires that you increase its value.

Records are kept for each transaction updating cluster data, both in the transaction coordinator and in the nodes where the actual updates are performed. These records contain state information needed to find UNDO records for rollback, lock queues, and other purposes.

This parameter should be set at a minimum to the number of records to be updated simultaneously in transactions, divided by the number of cluster data nodes. For example, in a cluster which has four data nodes and which is expected to handle one million concurrent updates using transactions, you should set this value to 1000000 / 4 = 250000. To help provide resiliency against failures, it is suggested that you set this parameter to a value that is high enough to permit an individual data node to handle the load for its node group. In other words, you should set the value equal to `total number of concurrent operations / number of node groups`. (In the case where there is a single node group, this is the same as the total number of concurrent operations for the entire cluster.)

Because each transaction always involves at least one operation, the value of `MaxNoOfConcurrentOperations` should always be greater than or equal to the value of `MaxNoOfConcurrentTransactions`.

Read queries which set locks also cause operation records to be created. Some extra space is allocated within individual nodes to accommodate cases where the distribution is not perfect over the nodes.

When queries make use of the unique hash index, there are actually two operation records used per record in the transaction. The first record represents the read in the index table and the second handles the operation on the base table.

The default value is 32768.

This parameter actually handles two values that can be configured separately. The first of these specifies how many operation records are to be placed with the transaction coordinator. The second part specifies how many operation records are to be local to the database.

A very large transaction performed on an eight-node cluster requires as many operation records in the transaction coordinator as there are reads, updates, and deletes involved in the transaction. However, the operation records of the are spread over all eight nodes. Thus, if it is necessary to configure the system for one very large transaction, it is a good idea to configure the two parts separately. `MaxNoOfConcurrentOperations` will always be used to calculate the number of operation records in the transaction coordinator portion of the node.

It is also important to have an idea of the memory requirements for operation records. These consume about 1KB per record.

- `MaxNoOfLocalOperations`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	UNDEFINED	32 - 4294967039 (0xFFFFFEFF)	N

By default, this parameter is calculated as 1.1 × `MaxNoOfConcurrentOperations`. This fits systems with many simultaneous transactions, none of them being very large. If there is a need to handle one very large transaction at a time and there are many nodes, it is a good idea to override the default value by explicitly specifying this parameter.

- `MaxDMLOperationsPerTransaction`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	operations (DML)	4294967295	32 - 4294967295	N

This parameter limits the size of a transaction. The transaction is aborted if it requires more than this many DML operations. The minimum number of operations per transaction is 32; however, you can set MaxDMLOperationsPerTransaction to 0 to disable any limitation on the number of DML operations per transaction. The maximum (and default) is 4294967295.

Transaction temporary storage. The next set of [ndbd] parameters is used to determine temporary storage when executing a statement that is part of a Cluster transaction. All records are released when the statement is completed and the cluster is waiting for the commit or rollback.

The default values for these parameters are adequate for most situations. However, users with a need to support transactions involving large numbers of rows or operations may need to increase these values to enable better parallelism in the system, whereas users whose applications require relatively small transactions can decrease the values to save memory.

- MaxNoOfConcurrentIndexOperations

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	8K	0 - 4294967039 (0xFFFFFEFF)	N

For queries using a unique hash index, another temporary set of operation records is used during a query's execution phase. This parameter sets the size of that pool of records. Thus, this record is allocated only while executing a part of a query. As soon as this part has been executed, the record is released. The state needed to handle aborts and commits is handled by the normal operation records, where the pool size is set by the parameter MaxNoOfConcurrentOperations.

The default value of this parameter is 8192. Only in rare cases of extremely high parallelism using unique hash indexes should it be necessary to increase this value. Using a smaller value is possible and can save memory if the DBA is certain that a high degree of parallelism is not required for the cluster.

- MaxNoOfFiredTriggers

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	4000	0 - 4294967039 (0xFFFFFEFF)	N

The default value of MaxNoOfFiredTriggers is 4000, which is sufficient for most situations. In some cases it can even be decreased if the DBA feels certain the need for parallelism in the cluster is not high.

A record is created when an operation is performed that affects a unique hash index. Inserting or deleting a record in a table with unique hash indexes or updating a column that is part of a unique hash index fires an insert or a delete in the index table. The resulting record is used to represent this index table operation while waiting for the original operation that fired it to complete. This operation is short-lived but can still require a large number of records in its pool for situations with many parallel write operations on a base table containing a set of unique hash indexes.

- TransactionBufferMemory

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	1M	1K - 4294967039 (0xFFFFFEFF)	N

The memory affected by this parameter is used for tracking operations fired when updating index tables and reading unique indexes. This memory is used to store the key and column information for these operations. It is only very rarely that the value for this parameter needs to be altered from the default.

The default value for `TransactionBufferMemory` is 1MB.

Normal read and write operations use a similar buffer, whose usage is even more short-lived. The compile-time parameter `ZATTRBUF_FILESIZE` (found in `ndb/src/kernel/blocks/Dbtc/Dbtc.hpp`) set to 4000 × 128 bytes (500KB). A similar buffer for key information, `ZDATABUF_FILESIZE` (also in `Dbtc.hpp`) contains 4000 × 16 = 62.5KB of buffer space. `Dbtc` is the module that handles transaction coordination.

Scans and buffering. There are additional `[ndbd]` parameters in the `Dblqh` module (in `ndb/src/kernel/blocks/Dblqh/Dblqh.hpp`) that affect reads and updates. These include `ZATTRINBUF_FILESIZE`, set by default to 10000 × 128 bytes (1250KB) and `ZDATABUF_FILE_SIZE`, set by default to 10000*16 bytes (roughly 156KB) of buffer space. To date, there have been neither any reports from users nor any results from our own extensive tests suggesting that either of these compile-time limits should be increased.

- `MaxNoOfConcurrentScans`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	256	2 - 500	N

This parameter is used to control the number of parallel scans that can be performed in the cluster. Each transaction coordinator can handle the number of parallel scans defined for this parameter. Each scan query is performed by scanning all partitions in parallel. Each partition scan uses a scan record in the node where the partition is located, the number of records being the value of this parameter times the number of nodes. The cluster should be able to sustain `MaxNoOfConcurrentScans` scans concurrently from all nodes in the cluster.

Scans are actually performed in two cases. The first of these cases occurs when no hash or ordered indexes exists to handle the query, in which case the query is executed by performing a full table scan. The second case is encountered when there is no hash index to support the query but there is an ordered index. Using the ordered index means executing a parallel range scan. The order is kept on the local partitions only, so it is necessary to perform the index scan on all partitions.

The default value of `MaxNoOfConcurrentScans` is 256. The maximum value is 500.

- `MaxNoOfLocalScans`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	[see text]	32 - 4294967039 (0xFFFFFEFF)	N

Specifies the number of local scan records if many scans are not fully parallelized. When the number of local scan records is not provided, it is calculated as shown here:

```
4 * MaxNoOfConcurrentScans * [# data nodes] + 2
```

The minimum value is 32.

- `BatchSizePerLocalScan`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	256	1 - 992	N

This parameter is used to calculate the number of lock records used to handle concurrent scan operations.

`BatchSizePerLocalScan` has a strong connection to the `BatchSize` defined in the SQL nodes.

- `LongMessageBuffer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	64M	512K - 4294967039 (0xFFFFFEFF)	N
NDB 7.3.1	bytes	4M	512K - 4294967039 (0xFFFFFEFF)	N
NDB 7.3.5	bytes	64M	512K - 4294967039 (0xFFFFFEFF)	N

This is an internal buffer used for passing messages within individual nodes and between nodes. The default is 64MB. (Prior to MySQL Cluster NDB 7.3.5, this was 4MB.)

This parameter seldom needs to be changed from the default.

- `MaxParallelCopyInstances`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.3	integer	0	0 - 64	S

This parameter sets the parallelization used in the copy phase of a node restart or system restart, when a node that is currently just starting is synchronised with a node that already has current data by copying over any changed records from the node that is up to date. Because full parallelism in such cases can lead to overload situations, `MaxParallelCopyInstances` was introduced in MySQL Cluster NDB 7.4.3 to provide a means to decrease it. This parameter's default value 0. This value means that the effective parallelism is equal to the number of LDM instances in the node just starting as well as the node updating it.

- `MaxParallelScansPerFragment`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	256	1 - 4294967039 (0xFFFFFEFF)	N

It is possible to configure the maximum number of parallel scans (`TUP` scans and `TUX` scans) allowed before they begin queuing for serial handling. You can increase this to take advantage of any unused CPU when performing large number of scans in parallel and improve their performance.

The default value for this parameter in MySQL Cluster NDB and later 7.3 is 256.

Memory Allocation

`MaxAllocate` [146]

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	32M	1M - 1G	N

This is the maximum size of the memory unit to use when allocating memory for tables. In cases where `NDB` gives `Out of memory` errors, but it is evident by examining the cluster logs or the output of `DUMP 1000` (see DUMP 1000) that all available memory has not yet been used, you can increase the value of this parameter (or `MaxNoOfTables`, or both) to cause `NDB` to make sufficient memory available.

Hash Map Size

`DefaultHashMapSize` [146]

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	LDM threads	3840	0 - 3840	N

MySQL Cluster NDB 7.2.7 and later use a larger default table hash map size (3840) than in previous releases (240). Beginning with MySQL Cluster NDB 7.2.11, the size of the table hash maps used by `NDB` is configurable using this parameter; previously this value was hard-coded. `DefaultHashMapSize` can take any of three possible values (0, 240, 3840). These values and their effects are described in the following table:

Value	Description / Effect
`0`	Use the lowest value set, if any, for this parameter among all data nodes and API nodes in the cluster; if it is not set on any data or API node, use the default value.
`240`	Original hash map size, used by default in all MySQL Cluster releases prior to MySQL Cluster NDB 7.2.7.
`3840`	Larger hash map size as used by default in MySQL Cluster NDB 7.2.7 and later

The primary intended use for this parameter is to facilitate upgrades and especially downgrades between MySQL Cluster NDB 7.2.7 and later MySQL Cluster versions, in which the larger hash map size (3840) is the default, and earlier releases (in which the default was 240), due to the fact that this change is not otherwise backward compatible (Bug #14800539). By setting this parameter to 240 prior to performing an upgrade from an older version where this value is in use, you can cause the cluster to continue using the smaller size for table hash maps, in which case the tables remain compatible with earlier versions following the upgrade. `DefaultHashMapSize` can be set for individual data nodes, API nodes, or both, but setting it once only, in the `[ndbd default]` section of the `config.ini` file, is the recommended practice.

After increasing this parameter, to have existing tables to take advantage of the new size, you can run `ALTER TABLE ... REORGANIZE PARTITION` on them, after which they can use the larger hash map size. This is in addition to performing a rolling restart, which makes the larger hash maps available to new tables, but does not enable existing tables to use them.

Decreasing this parameter online after any tables have been created or modified with `DefaultHashMapSize` equal to 3840 is not currently supported.

Logging and checkpointing. The following `[ndbd]` parameters control log and checkpoint behavior.

- `NoOfFragmentLogFiles`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	16	3 - 4294967039 (0xFFFFFEFF)	IN

  This parameter sets the number of REDO log files for the node, and thus the amount of space allocated to REDO logging. Because the REDO log files are organized in a ring, it is extremely important that the first and last log files in the set (sometimes referred to as the “head” and “tail” log files, respectively) do not meet. When these approach one another too closely, the node begins aborting all transactions encompassing updates due to a lack of room for new log records.

  A `REDO` log record is not removed until the required number of local checkpoints has been completed since that log record was inserted. (In MySQL Cluster NDB 7.3 and later, only 2 local checkpoints are necessary). Checkpointing frequency is determined by its own set of configuration parameters discussed elsewhere in this chapter.

  The default parameter value is 16, which by default means 16 sets of 4 16MB files for a total of 1024MB. The size of the individual log files is configurable using the `FragmentLogFileSize` parameter. In scenarios requiring a great many updates, the value for `NoOfFragmentLogFiles` may need to be set as high as 300 or even higher to provide sufficient space for REDO logs.

If the checkpointing is slow and there are so many writes to the database that the log files are full and the log tail cannot be cut without jeopardizing recovery, all updating transactions are aborted with internal error code 410 (`Out of log file space temporarily`). This condition prevails until a checkpoint has completed and the log tail can be moved forward.

> **Important**
>
> This parameter cannot be changed "on the fly"; you must restart the node using `--initial`. If you wish to change this value for all data nodes in a running cluster, you can do so using a rolling node restart (using `--initial` when starting each data node).

- `FragmentLogFileSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	16M	4M - 1G	IN

Setting this parameter enables you to control directly the size of redo log files. This can be useful in situations when MySQL Cluster is operating under a high load and it is unable to close fragment log files quickly enough before attempting to open new ones (only 2 fragment log files can be open at one time); increasing the size of the fragment log files gives the cluster more time before having to open each new fragment log file. The default value for this parameter is 16M.

For more information about fragment log files, see the description for `NoOfFragmentLogFiles`.

- `InitFragmentLogFiles`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	[see values]	SPARSE	SPARSE, FULL	IN

By default, fragment log files are created sparsely when performing an initial start of a data node—that is, depending on the operating system and file system in use, not all bytes are necessarily written to disk. However, it is possible to override this behavior and force all bytes to be written, regardless of the platform and file system type being used, by means of this parameter. `InitFragmentLogFiles` takes either of two values:

- `SPARSE`. Fragment log files are created sparsely. This is the default value.
- `FULL`. Force all bytes of the fragment log file to be written to disk.

Depending on your operating system and file system, setting `InitFragmentLogFiles=FULL` may help eliminate I/O errors on writes to the REDO log.

- `MaxNoOfOpenFiles`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	20 - 4294967039 (0xFFFFFEFF)	N

This parameter sets a ceiling on how many internal threads to allocate for open files. *Any situation requiring a change in this parameter should be reported as a bug*.

The default value is 0. However, the minimum value to which this parameter can be set is 20.

- `InitialNoOfOpenFiles`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	files	27	20 - 4294967039 (0xFFFFFEFF)	N

This parameter sets the initial number of internal threads to allocate for open files.

The default value is 27.

- `MaxNoOfSavedMessages`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	25	0 - 4294967039 (0xFFFFFEFF)	N

This parameter sets the maximum number of trace files that are kept before overwriting old ones. Trace files are generated when, for whatever reason, the node crashes.

The default is 25 trace files.

- `MaxLCPStartDelay`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	seconds	0	0 - 600	N

In parallel data node recovery, only table data is actually copied and synchronized in parallel; synchronization of metadata such as dictionary and checkpoint information is done in a serial fashion. In addition, recovery of dictionary and checkpoint information cannot be executed in parallel with performing of local checkpoints. This means that, when starting or restarting many data nodes concurrently, data nodes may be forced to wait while a local checkpoint is performed, which can result in longer node recovery times.

It is possible to force a delay in the local checkpoint to permit more (and possibly all) data nodes to complete metadata synchronization; once each data node's metadata synchronization is complete, all of the data nodes can recover table data in parallel, even while the local checkpoint is being executed. To force such a delay, set `MaxLCPStartDelay`, which determines the number of seconds the cluster can wait to begin a local checkpoint while data nodes continue to synchronize metadata. This parameter should be set in the `[ndbd default]` section of the `config.ini` file, so that it is the same for all data nodes. The maximum value is 600; the default is 0.

- `LcpScanProgressTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.3	second	60	0 - 4294967039 (0xFFFFFEFF)	N

A local checkpoint fragment scan watchdog checks periodically for no progress in each fragment scan performed as part of a local checkpoint, and shuts down the node if there is no progress after a given amount of time has elapsed. Prior to MySQL Cluster NDB 7.3.3, this interval is always 60 seconds (Bug #16630410). In MySQL Cluster NDB 7.3.3 and later, this interval can be set using the `LcpScanProgressTimeout` data node configuration parameter, which sets the maximum time for which the local checkpoint can be stalled before the LCP fragment scan watchdog shuts down the node.

The default value is 60 seconds (providing compatibility with previous releases). Setting this parameter to 0 disables the LCP fragment scan watchdog altogether.

Metadata objects. The next set of `[ndbd]` parameters defines pool sizes for metadata objects, used to define the maximum number of attributes, tables, indexes, and trigger objects used by indexes, events, and replication between clusters.

> **Note**
>
> These act merely as “suggestions” to the cluster, and any that are not specified revert to the default values shown.

- `MaxNoOfAttributes`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	1000	32 - 4294967039 (0xFFFFFEFF)	N

This parameter sets a suggested maximum number of attributes that can be defined in the cluster; like `MaxNoOfTables`, it is not intended to function as a hard upper limit.

(In older MySQL Cluster releases, this parameter was sometimes treated as a hard limit for certain operations. This caused problems with MySQL Cluster Replication, when it was possible to create more tables than could be replicated, and sometimes led to confusion when it was possible [or not possible, depending on the circumstances] to create more than `MaxNoOfAttributes` attributes.)

The default value is 1000, with the minimum possible value being 32. The maximum is 4294967039. Each attribute consumes around 200 bytes of storage per node due to the fact that all metadata is fully replicated on the servers.

When setting `MaxNoOfAttributes`, it is important to prepare in advance for any `ALTER TABLE` statements that you might want to perform in the future. This is due to the fact, during the execution of `ALTER TABLE` on a Cluster table, 3 times the number of attributes as in the original table are used, and a good practice is to permit double this amount. For example, if the MySQL Cluster table having the greatest number of attributes (*`greatest_number_of_attributes`*) has 100 attributes, a good starting point for the value of `MaxNoOfAttributes` would be `6 * greatest_number_of_attributes = 600`.

You should also estimate the average number of attributes per table and multiply this by `MaxNoOfTables`. If this value is larger than the value obtained in the previous paragraph, you should use the larger value instead.

Assuming that you can create all desired tables without any problems, you should also verify that this number is sufficient by trying an actual `ALTER TABLE` after configuring the parameter. If this is not successful, increase `MaxNoOfAttributes` by another multiple of `MaxNoOfTables` and test it again.

- `MaxNoOfTables`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	128	8 - 20320	N

A table object is allocated for each table and for each unique hash index in the cluster. This parameter sets a suggested maximum number of table objects for the cluster as a whole; like `MaxNoOfAttributes`, it is not intended to function as a hard upper limit.

(In older MySQL Cluster releases, this parameter was sometimes treated as a hard limit for certain operations. This caused problems with MySQL Cluster Replication, when it was possible to create more tables than could be replicated, and sometimes led to confusion when it was possible [or not possible, depending on the circumstances] to create more than `MaxNoOfTables` tables.)

For each attribute that has a `BLOB` data type an extra table is used to store most of the `BLOB` data. These tables also must be taken into account when defining the total number of tables.

The default value of this parameter is 128. The minimum is 8 and the maximum is 20320. Each table object consumes approximately 20KB per node.

Note

The sum of `MaxNoOfTables`, `MaxNoOfOrderedIndexes`, and `MaxNoOfUniqueHashIndexes` must not exceed $2^{32} - 2$ (4294967294).

- `MaxNoOfOrderedIndexes`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	128	0 - 4294967039 (0xFFFFFEFF)	N

For each ordered index in the cluster, an object is allocated describing what is being indexed and its storage segments. By default, each index so defined also defines an ordered index. Each unique index and primary key has both an ordered index and a hash index. `MaxNoOfOrderedIndexes` sets the total number of ordered indexes that can be in use in the system at any one time.

The default value of this parameter is 128. Each index object consumes approximately 10KB of data per node.

Note

The sum of `MaxNoOfTables`, `MaxNoOfOrderedIndexes`, and `MaxNoOfUniqueHashIndexes` must not exceed $2^{32} - 2$ (4294967294).

- `MaxNoOfUniqueHashIndexes`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	64	0 - 4294967039 (0xFFFFFEFF)	N

For each unique index that is not a primary key, a special table is allocated that maps the unique key to the primary key of the indexed table. By default, an ordered index is also defined for each unique index. To prevent this, you must specify the `USING HASH` option when defining the unique index.

The default value is 64. Each index consumes approximately 15KB per node.

Note

The sum of `MaxNoOfTables`, `MaxNoOfOrderedIndexes`, and `MaxNoOfUniqueHashIndexes` must not exceed $2^{32} - 2$ (4294967294).

- `MaxNoOfTriggers`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	768	0 - 4294967039 (0xFFFFFEFF)	N

Internal update, insert, and delete triggers are allocated for each unique hash index. (This means that three triggers are created for each unique hash index.) However, an *ordered* index requires only a single trigger object. Backups also use three trigger objects for each normal table in the cluster.

Replication between clusters also makes use of internal triggers.

This parameter sets the maximum number of trigger objects in the cluster.

The default value is 768.

- `MaxNoOfIndexes`

  This parameter is deprecated and subject to removal in a future version of MySQL Cluster. You should use `MaxNoOfOrderedIndexes` and `MaxNoOfUniqueHashIndexes` instead.

  This parameter is used only by unique hash indexes. There needs to be one record in this pool for each unique hash index defined in the cluster.

  The default value of this parameter is 128.

- `MaxNoOfSubscriptions`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 4294967039 (0xFFFFFEFF)	N

  Each `NDB` table in a MySQL Cluster requires a subscription in the NDB kernel. For some NDB API applications, it may be necessary or desirable to change this parameter. However, for normal usage with MySQL servers acting as SQL nodes, there is not any need to do so.

  The default value for `MaxNoOfSubscriptions` is 0, which is treated as equal to `MaxNoOfTables`. Each subscription consumes 108 bytes.

- `MaxNoOfSubscribers`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 4294967039 (0xFFFFFEFF)	N

  This parameter is of interest only when using MySQL Cluster Replication. The default value is 0, which is treated as `2 * MaxNoOfTables`; that is, there is one subscription per `NDB` table for each of two MySQL servers (one acting as the replication master and the other as the slave). Each subscriber uses 16 bytes of memory.

  When using circular replication, multi-master replication, and other replication setups involving more than 2 MySQL servers, you should increase this parameter to the number of `mysqld` processes included in replication (this is often, but not always, the same as the number of clusters). For example, if you have a circular replication setup using three MySQL Clusters, with one `mysqld` attached to each cluster, and each of these `mysqld` processes acts as a master and as a slave, you should set `MaxNoOfSubscribers` equal to `3 * MaxNoOfTables`.

  For more information, see Chapter 8, *MySQL Cluster Replication*.

- `MaxNoOfConcurrentSubOperations`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	256	0 - 4294967039 (0xFFFFFEFF)	N

  This parameter sets a ceiling on the number of operations that can be performed by all API nodes in the cluster at one time. The default value (256) is sufficient for normal operations, and might need to be adjusted only in scenarios where there are a great many API nodes each performing a high volume of operations concurrently.

Boolean parameters. The behavior of data nodes is also affected by a set of `[ndbd]` parameters taking on boolean values. These parameters can each be specified as `TRUE` by setting them equal to `1` or `Y`, and as `FALSE` by setting them equal to `0` or `N`.

- `LateAlloc`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	1	0 - 1	N

Allocate memory for this data node after a connection to the management server has been established. Enabled by default.

- `LockPagesInMainMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	0	0 - 2	N

For a number of operating systems, including Solaris and Linux, it is possible to lock a process into memory and so avoid any swapping to disk. This can be used to help guarantee the cluster's real-time characteristics.

This parameter takes one of the integer values `0`, `1`, or `2`, which act as shown in the following list:

- `0`: Disables locking. This is the default value.
- `1`: Performs the lock after allocating memory for the process.
- `2`: Performs the lock before memory for the process is allocated.

If the operating system is not configured to permit unprivileged users to lock pages, then the data node process making use of this parameter may have to be run as system root. (`LockPagesInMainMemory` uses the `mlockall` function. From Linux kernel 2.6.9, unprivileged users can lock memory as limited by `max locked memory`. For more information, see `ulimit -l` and http://linux.die.net/man/2/mlock).

Note

In older MySQL Cluster releases, this parameter was a Boolean. `0` or `false` was the default setting, and disabled locking. `1` or `true` enabled locking of the process after its memory was allocated. MySQL Cluster NDB 7.3 and later treats using `true` or `false` for the value of this parameter as an error.

Important

Beginning with `glibc` 2.10, `glibc` uses per-thread arenas to reduce lock contention on a shared pool, which consumes real memory. In general, a data node process does not need per-thread arenas, since it does not perform any memory allocation after startup. (This difference in allocators does not appear to affect performance significantly.)

The `glibc` behavior is intended to be configurable via the `MALLOC_ARENA_MAX` environment variable, but a bug in this mechanism prior to `glibc` 2.16 meant that this variable could not be set to less than 8, so that the wasted memory could not be reclaimed. (Bug #15907219; see also http://sourceware.org/bugzilla/show_bug.cgi?id=13137 for more information concerning this issue.)

> One possible workaround for this problem is to use the `LD_PRELOAD` environment variable to preload a `jemalloc` memory allocation library to take the place of that supplied with `glibc`.

- `StopOnError`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	1	0, 1	N

This parameter specifies whether a data node process should exit or perform an automatic restart when an error condition is encountered.

This parameter's default value is `1`; this means that, by default, an error causes the data node process to halt.

Users of MySQL Cluster Manager should note that, when `StopOnError` equals 1, this prevents the MySQL Cluster Manager agent from restarting any data nodes after it has performed its own restart and recovery. See Starting and Stopping the Agent on Linux, for more information.

- `CrashOnCorruptedTuple`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	true	true, false	S

When this parameter is enabled, it forces a data node to shut down whenever it encounters a corrupted tuple. In MySQL Cluster NDB 7.3 and later, it is enabled by default.

- `Diskless`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	true|false (1|0)	false	true, false	IS

It is possible to specify MySQL Cluster tables as *diskless*, meaning that tables are not checkpointed to disk and that no logging occurs. Such tables exist only in main memory. A consequence of using diskless tables is that neither the tables nor the records in those tables survive a crash. However, when operating in diskless mode, it is possible to run `ndbd` on a diskless computer.

> **Important**
>
> This feature causes the *entire* cluster to operate in diskless mode.

When this feature is enabled, Cluster online backup is disabled. In addition, a partial start of the cluster is not possible.

`Diskless` is disabled by default.

- `ODirect`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Enabling this parameter causes `NDB` to attempt using `O_DIRECT` writes for LCP, backups, and redo logs, often lowering `kswapd` and CPU usage. When using MySQL Cluster on Linux, enable `ODirect` if you are using a 2.6 or later kernel.

`ODirect` is disabled by default.

- `RestartOnErrorInsert`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	error code	2	0 - 4	N

This feature is accessible only when building the debug version where it is possible to insert errors in the execution of individual blocks of code as part of testing.

This feature is disabled by default.

- `CompressedBackup`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Setting this parameter to `1` causes backup files to be compressed. The compression used is equivalent to `gzip --fast`, and can save 50% or more of the space required on the data node to store uncompressed backup files. Compressed backups can be enabled for individual data nodes, or for all data nodes (by setting this parameter in the `[ndbd default]` section of the `config.ini` file).

> **Important**
>
> You cannot restore a compressed backup to a cluster running a MySQL version that does not support this feature.

The default value is `0` (disabled).

- `CompressedLCP`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Setting this parameter to `1` causes local checkpoint files to be compressed. The compression used is equivalent to `gzip --fast`, and can save 50% or more of the space required on the data node to store uncompressed checkpoint files. Compressed LCPs can be enabled for individual data nodes, or for all data nodes (by setting this parameter in the `[ndbd default]` section of the `config.ini` file).

> **Important**
>
> You cannot restore a compressed local checkpoint to a cluster running a MySQL version that does not support this feature.

The default value is `0` (disabled).

Controlling Timeouts, Intervals, and Disk Paging

There are a number of `[ndbd]` parameters specifying timeouts and intervals between various actions in Cluster data nodes. Most of the timeout values are specified in milliseconds. Any exceptions to this are mentioned where applicable.

- `TimeBetweenWatchDogCheck`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	6000	70 - 4294967039 (0xFFFFFEFF)	N

To prevent the main thread from getting stuck in an endless loop at some point, a “watchdog” thread checks the main thread. This parameter specifies the number of milliseconds between checks. If the process remains in the same state after three checks, the watchdog thread terminates it.

This parameter can easily be changed for purposes of experimentation or to adapt to local conditions. It can be specified on a per-node basis although there seems to be little reason for doing so.

The default timeout is 6000 milliseconds (6 seconds).

- `TimeBetweenWatchDogCheckInitial`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	6000	70 - 4294967039 (0xFFFFFEFF)	N

This is similar to the `TimeBetweenWatchDogCheck` parameter, except that `TimeBetweenWatchDogCheckInitial` controls the amount of time that passes between execution checks inside a database node in the early start phases during which memory is allocated.

The default timeout is 6000 milliseconds (6 seconds).

- `StartPartialTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	30000	0 - 4294967039 (0xFFFFFEFF)	N

This parameter specifies how long the Cluster waits for all data nodes to come up before the cluster initialization routine is invoked. This timeout is used to avoid a partial Cluster startup whenever possible.

This parameter is overridden when performing an initial start or initial restart of the cluster.

The default value is 30000 milliseconds (30 seconds). 0 disables the timeout, in which case the cluster may start only if all nodes are available.

- `StartPartitionedTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	60000	0 - 4294967039 (0xFFFFFEFF)	N

If the cluster is ready to start after waiting for `StartPartialTimeout` milliseconds but is still possibly in a partitioned state, the cluster waits until this timeout has also passed. If `StartPartitionedTimeout` is set to 0, the cluster waits indefinitely.

This parameter is overridden when performing an initial start or initial restart of the cluster.

The default timeout is 60000 milliseconds (60 seconds).

- `StartFailureTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	0	0 - 4294967039 (0xFFFFFEFF)	N

If a data node has not completed its startup sequence within the time specified by this parameter, the node startup fails. Setting this parameter to 0 (the default value) means that no data node timeout is applied.

For nonzero values, this parameter is measured in milliseconds. For data nodes containing extremely large amounts of data, this parameter should be increased. For example, in the case of a data node containing several gigabytes of data, a period as long as 10−15 minutes (that is, 600000 to 1000000 milliseconds) might be required to perform a node restart.

- `StartNoNodeGroupTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	15000	0 - 4294967039 (0xFFFFFEFF)	N

When a data node is configured with `Nodegroup = 65536`, is regarded as not being assigned to any node group. When that is done, the cluster waits `StartNoNodegroupTimeout` milliseconds, then treats such nodes as though they had been added to the list passed to the `--nowait-nodes` option, and starts. The default value is `15000` (that is, the management server waits 15 seconds). Setting this parameter equal to `0` means that the cluster waits indefinitely.

`StartNoNodegroupTimeout` must be the same for all data nodes in the cluster; for this reason, you should always set it in the `[ndbd default]` section of the `config.ini` file, rather than for individual data nodes.

See Section 7.13, “Adding MySQL Cluster Data Nodes Online”, for more information.

- `HeartbeatIntervalDbDb`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	5000	10 - 4294967039 (0xFFFFFEFF)	N

One of the primary methods of discovering failed nodes is by the use of heartbeats. This parameter states how often heartbeat signals are sent and how often to expect to receive them. After missing three heartbeat intervals in a row, the node is declared dead. Thus, the maximum time for discovering a failure through the heartbeat mechanism is four times the heartbeat interval.

In MySQL Cluster NDB 7.3 and later, the default heartbeat interval is 5000 milliseconds (5 seconds). This parameter must not be changed drastically and should not vary widely between nodes. If one node uses 5000 milliseconds and the node watching it uses 1000 milliseconds, obviously the node will be declared dead very quickly. This parameter can be changed during an online software upgrade, but only in small increments.

See also Network communication and latency.

- `HeartbeatIntervalDbApi`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	1500	100 - 4294967039 (0xFFFFFEFF)	N

Each data node sends heartbeat signals to each MySQL server (SQL node) to ensure that it remains in contact. If a MySQL server fails to send a heartbeat in time it is declared “dead,” in which case all ongoing transactions are completed and all resources released. The SQL node cannot reconnect until all activities initiated by the previous MySQL instance have been completed. The three-heartbeat criteria for this determination are the same as described for `HeartbeatIntervalDbDb`.

The default interval is 1500 milliseconds (1.5 seconds). This interval can vary between individual data nodes because each data node watches the MySQL servers connected to it, independently of all other data nodes.

For more information, see Network communication and latency.

- `HeartbeatOrder`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	0	0 - 65535	S

Data nodes send heartbeats to one another in a circular fashion whereby each data node monitors the previous one. If a heartbeat is not detected by a given data node, this node declares the previous data node in the circle “dead” (that is, no longer accessible by the cluster). The determination that a data node is dead is done globally; in other words; once a data node is declared dead, it is regarded as such by all nodes in the cluster.

It is possible for heartbeats between data nodes residing on different hosts to be too slow compared to heartbeats between other pairs of nodes (for example, due to a very low heartbeat interval or temporary connection problem), such that a data node is declared dead, even though the node can still function as part of the cluster. .

In this type of situation, it may be that the order in which heartbeats are transmitted between data nodes makes a difference as to whether or not a particular data node is declared dead. If this declaration occurs unnecessarily, this can in turn lead to the unnecessary loss of a node group and as thus to a failure of the cluster.

Consider a setup where there are 4 data nodes A, B, C, and D running on 2 host computers `host1` and `host2`, and that these data nodes make up 2 node groups, as shown in the following table:

Node Group	Nodes Running on `host1`	Nodes Running on `host2`
Node Group 0:	Node A	Node B
Node Group 1:	Node C	Node D

Suppose the heartbeats are transmitted in the order A->B->C->D->A. In this case, the loss of the heartbeat between the hosts causes node B to declare node A dead and node C to declare node B dead. This results in loss of Node Group 0, and so the cluster fails. On the other hand, if the order of transmission is A->B->D->C->A (and all other conditions remain as previously stated), the loss of the heartbeat causes nodes A and D to be declared dead; in this case, each node group has one surviving node, and the cluster survives.

The `HeartbeatOrder` configuration parameter makes the order of heartbeat transmission user-configurable. The default value for `HeartbeatOrder` is zero; allowing the default value to be used on all data nodes causes the order of heartbeat transmission to be determined by `NDB`. If this parameter is used, it must be set to a nonzero value (maximum 65535) for every data node in the cluster, and this value must be unique for each data node; this causes the heartbeat transmission to proceed from data node to data node in the order of their `HeartbeatOrder` values from lowest to highest (and then directly from the data node having the highest `HeartbeatOrder` to the data node having the lowest value, to complete the circle). The values need not be consecutive; for example, to force the heartbeat transmission order A->B->D->C->A in the scenario outlined previously, you could set the `HeartbeatOrder` values as shown here:

Node	`HeartbeatOrder`
A	10
B	20

Node	HeartbeatOrder
C	30
D	25

To use this parameter to change the heartbeat transmission order in a running MySQL Cluster, you must first set HeartbeatOrder for each data node in the cluster in the global configuration (config.ini) file (or files). To cause the change to take effect, you must perform either of the following:

- A complete shutdown and restart of the entire cluster.
- 2 rolling restarts of the cluster in succession. *All nodes must be restarted in the same order in both rolling restarts*.

You can use DUMP 908 to observe the effect of this parameter in the data node logs.

- ConnectCheckIntervalDelay

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	0	0 - 4294967039 (0xFFFFFEFF)	N

This parameter enables connection checking between data nodes. A data node that fails to respond within an interval of ConnectCheckIntervalDelay milliseconds is considered suspect, and is considered dead after two such intervals.

The default value for this parameter is 0; this is a change from MySQL Cluster NDB 7.1.

- TimeBetweenLocalCheckpoints

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	number of 4-byte words, as a base-2 logarithm	20	0 - 31	N

This parameter is an exception in that it does not specify a time to wait before starting a new local checkpoint; rather, it is used to ensure that local checkpoints are not performed in a cluster where relatively few updates are taking place. In most clusters with high update rates, it is likely that a new local checkpoint is started immediately after the previous one has been completed.

The size of all write operations executed since the start of the previous local checkpoints is added. This parameter is also exceptional in that it is specified as the base-2 logarithm of the number of 4-byte words, so that the default value 20 means 4MB (4×2^{20}) of write operations, 21 would mean 8MB, and so on up to a maximum value of 31, which equates to 8GB of write operations.

All the write operations in the cluster are added together. Setting TimeBetweenLocalCheckpoints to 6 or less means that local checkpoints will be executed continuously without pause, independent of the cluster's workload.

- TimeBetweenGlobalCheckpoints

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	2000	20 - 32000	N

When a transaction is committed, it is committed in main memory in all nodes on which the data is mirrored. However, transaction log records are not flushed to disk as part of the commit. The reasoning behind this behavior is that having the transaction safely committed on at least two autonomous host machines should meet reasonable standards for durability.

It is also important to ensure that even the worst of cases—a complete crash of the cluster—is handled properly. To guarantee that this happens, all transactions taking place within a given interval are put into a global checkpoint, which can be thought of as a set of committed transactions that has been flushed to disk. In other words, as part of the commit process, a transaction is placed in a global checkpoint group. Later, this group's log records are flushed to disk, and then the entire group of transactions is safely committed to disk on all computers in the cluster.

This parameter defines the interval between global checkpoints. The default is 2000 milliseconds.

- `TimeBetweenGlobalCheckpointsTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.5	milliseconds	120000	10 - 4294967039 (0xFFFFFEFF)	N
NDB 7.3.9	milliseconds	120000	10 - 4294967039 (0xFFFFFEFF)	N

This parameter defines the minimum timeout between global checkpoints. The default is 120000 milliseconds.

This parameter was added in MySQL Cluster NDB 7.3.9 and MySQL Cluster 7.4.5. (Bug #20069617)

- `TimeBetweenEpochs`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	100	0 - 32000	N

This parameter defines the interval between synchronization epochs for MySQL Cluster Replication. The default value is 100 milliseconds.

`TimeBetweenEpochs` is part of the implementation of “micro-GCPs”, which can be used to improve the performance of MySQL Cluster Replication.

- `TimeBetweenEpochsTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	0	0 - 256000	N

This parameter defines a timeout for synchronization epochs for MySQL Cluster Replication. If a node fails to participate in a global checkpoint within the time determined by this parameter, the node is shut down. In MySQL Cluster NDB 7.3 and later, the default value is 0; in other words, the timeout is disabled.

`TimeBetweenEpochsTimeout` is part of the implementation of “micro-GCPs”, which can be used to improve the performance of MySQL Cluster Replication.

The current value of this parameter and a warning are written to the cluster log whenever a GCP save takes longer than 1 minute or a GCP save takes longer than 10 seconds.

Setting this parameter to zero has the effect of disabling GCP stops caused by save timeouts, commit timeouts, or both. The maximum possible value for this parameter is 256000 milliseconds.

- `MaxBufferedEpochs`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	epochs	100	0 - 100000	N

The number of unprocessed epochs by which a subscribing node can lag behind. Exceeding this number causes a lagging subscriber to be disconnected.

The default value of 100 is sufficient for most normal operations. If a subscribing node does lag enough to cause disconnections, it is usually due to network or scheduling issues with regard to processes or threads. (In rare circumstances, the problem may be due to a bug in the `NDB` client.) It may be desirable to set the value lower than the default when epochs are longer.

Disconnection prevents client issues from affecting the data node service, running out of memory to buffer data, and eventually shutting down. Instead, only the client is affected as a result of the disconnect (by, for example gap events in the binary log), forcing the client to reconnect or restart the process.

- `MaxBufferedEpochBytes`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	26214400	26214400 (0x01900000) - 4294967039 (0xFFFFFEFF)	N

The total number of bytes allocated for buffering epochs by this node.

- `TimeBetweenInactiveTransactionAbortCheck`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	1000	1000 - 4294967039 (0xFFFFFEFF)	N

Timeout handling is performed by checking a timer on each transaction once for every interval specified by this parameter. Thus, if this parameter is set to 1000 milliseconds, every transaction will be checked for timing out once per second.

The default value is 1000 milliseconds (1 second).

- `TransactionInactiveTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	[see text]	0 - 4294967039 (0xFFFFFEFF)	N

This parameter states the maximum time that is permitted to lapse between operations in the same transaction before the transaction is aborted.

The default for this parameter is `4G` (also the maximum). For a real-time database that needs to ensure that no transaction keeps locks for too long, this parameter should be set to a relatively small value. The unit is milliseconds.

- `TransactionDeadlockDetectionTimeout`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	1200	50 - 4294967039 (0xFFFFFEFF)	N

When a node executes a query involving a transaction, the node waits for the other nodes in the cluster to respond before continuing. A failure to respond can occur for any of the following reasons:

- The node is “dead”
- The operation has entered a lock queue
- The node requested to perform the action could be heavily overloaded.

This timeout parameter states how long the transaction coordinator waits for query execution by another node before aborting the transaction, and is important for both node failure handling and deadlock detection.

The default timeout value is 1200 milliseconds (1.2 seconds).

The minimum for this parameter is 50 milliseconds.

- `DiskSyncSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	4M	32K - 4294967039 (0xFFFFFEFF)	N

This is the maximum number of bytes to store before flushing data to a local checkpoint file. This is done to prevent write buffering, which can impede performance significantly. This parameter is *not* intended to take the place of `TimeBetweenLocalCheckpoints`.

> **Note**
>
> When `ODirect` is enabled, it is not necessary to set `DiskSyncSize`; in fact, in such cases its value is simply ignored.

The default value is 4M (4 megabytes).

- `DiskCheckpointSpeed`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	10M	1M - 4294967039 (0xFFFFFEFF)	N

The amount of data,in bytes per second, that is sent to disk during a local checkpoint. This allocation is shared by DML operations and backups (but not backup logging), which means that backups started during times of intensive DML may be impaired by flooding of the redo log buffer and may fail altogether if the contention is sufficiently severe.

The default value is 10M (10 megabytes per second).

This parameter is deprecated in MySQL Cluster 7.4.1 and later, where you can use instead the configuration parameters `MinDiskWriteSpeed`, `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` to control write speeds for LCPs and backups.

- `DiskCheckpointSpeedInRestart`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	100M	1M - 4294967039 (0xFFFFFEFF)	N

The amount of data,in bytes per second, that is sent to disk during a local checkpoint as part of a restart operation.

The default value is 100M (100 megabytes per second).

This parameter is deprecated in MySQL Cluster 7.4.1 and later, where you can instead use the configuration parameters `MinDiskWriteSpeed`, `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` to control write speeds for LCPs and backups.

- `NoOfDiskPagesToDiskAfterRestartTUP`

  This parameter is deprecated and subject to removal in a future version of MySQL Cluster. Use `DiskCheckpointSpeedInRestart` and `DiskSyncSize` instead. Beginning with MySQL Cluster 7.4.1, you should instead use the configuration parameters `MinDiskWriteSpeed`, `MaxDiskWriteSpeed`, `MaxDiskWriteSpeedOtherNodeRestart`, and `MaxDiskWriteSpeedOwnRestart` introduced in that release.

- `MaxDiskWriteSpeed`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.1	numeric	20M	1M - 1024G	S

Set the maximum rate for writing to disk, in bytes per second, by local checkpoints and backup operations when no restarts (by this data node or any other data node) are taking place in this MySQL Cluster.

For setting the maximum rate of disk writes allowed while this data node is restarting, use `MaxDiskWriteSpeedOwnRestart`. For setting the maximum rate of disk writes allowed while other data nodes are restarting, use `MaxDiskWriteSpeedOtherNodeRestart`. The minimum speed for disk writes by all LCPs and backup operations can be adjusted by setting `MinDiskWriteSpeed`.

`MaxDiskWriteSpeed` was added in MySQL Cluster NDB 7.4.1.

- `MaxDiskWriteSpeedOtherNodeRestart`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.1	numeric	50M	1M - 1024G	S

Set the maximum rate for writing to disk, in bytes per second, by local checkpoints and backup operations when one or more data nodes in this MySQL Cluster are restarting, other than this node.

For setting the maximum rate of disk writes allowed while this data node is restarting, use `MaxDiskWriteSpeedOwnRestart`. For setting the maximum rate of disk writes allowed when no data nodes are restarting anywhere in the cluster, use `MaxDiskWriteSpeed`. The minimum speed for disk writes by all LCPs and backup operations can be adjusted by setting `MinDiskWriteSpeed`.

`MaxDiskWriteSpeedOtherNodeRestart` was added in MySQL Cluster NDB 7.4.1.

- `MaxDiskWriteSpeedOwnRestart`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.1	numeric	200M	1M - 1024G	S

Set the maximum rate for writing to disk, in bytes per second, by local checkpoints and backup operations while this data node is restarting.

For setting the maximum rate of disk writes allowed while other data nodes are restarting, use MaxDiskWriteSpeedOtherNodeRestart. For setting the maximum rate of disk writes allowed when no data nodes are restarting anywhere in the cluster, use MaxDiskWriteSpeed. The minimum speed for disk writes by all LCPs and backup operations can be adjusted by setting MinDiskWriteSpeed.

MaxDiskWriteSpeedOwnRestart was added in MySQL Cluster NDB 7.4.1.

- MinDiskWriteSpeed

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.1	numeric	10M	1M - 1024G	S

Set the minimum rate for writing to disk, in bytes per second, by local checkpoints and backup operations.

The maximum rates of disk writes allowed for LCPs and backups under various conditions are adjustable using the parameters MaxDiskWriteSpeed, MaxDiskWriteSpeedOwnRestart, and MaxDiskWriteSpeedOtherNodeRestart. See the descriptions of these parameters for more information.

MinDiskWriteSpeed was added in MySQL Cluster NDB 7.4.1.

- NoOfDiskPagesToDiskAfterRestartACC

This parameter is deprecated and subject to removal in a future version of MySQL Cluster. In MySQL Cluster NDB 7.3, use DiskCheckpointSpeedInRestart and DiskSyncSize instead. In MySQL Cluster NDB 7.4.1 and later, you should use the parameters MinDiskWriteSpeed, MaxDiskWriteSpeed, MaxDiskWriteSpeedOtherNodeRestart, and MaxDiskWriteSpeedOwnRestart.

- NoOfDiskPagesToDiskDuringRestartTUP (DEPRECATED)

This parameter is deprecated and subject to removal in a future version of MySQL Cluster. In MySQL Cluster NDB 7.3, use DiskCheckpointSpeedInRestart and DiskSyncSize instead. In MySQL Cluster NDB 7.4.1 and later, you should use the parameters MinDiskWriteSpeed, MaxDiskWriteSpeed, MaxDiskWriteSpeedOtherNodeRestart, and MaxDiskWriteSpeedOwnRestart.

- NoOfDiskPagesToDiskDuringRestartACC (DEPRECATED)

This parameter is deprecated and subject to removal in a future version of MySQL Cluster. In MySQL Cluster NDB 7.3, use DiskCheckpointSpeedInRestart and DiskSyncSize instead. In MySQL Cluster NDB 7.4.1 and later, you should use the parameters MinDiskWriteSpeed, MaxDiskWriteSpeed, MaxDiskWriteSpeedOtherNodeRestart, and MaxDiskWriteSpeedOwnRestart.

- ArbitrationTimeout

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	7500	10 - 4294967039 (0xFFFFFEFF)	N

This parameter specifies how long data nodes wait for a response from the arbitrator to an arbitration message. If this is exceeded, the network is assumed to have split.

In MySQL Cluster NDB 7.3 and later, the default value is 7500 milliseconds (7.5 seconds).

- `Arbitration`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	enumeration	Default	Default, Disabled, WaitExternal	N

The `Arbitration` parameter enables a choice of arbitration schemes, corresponding to one of 3 possible values for this parameter:

- **Default.** This enables arbitration to proceed normally, as determined by the `ArbitrationRank` settings for the management and API nodes. This is the default value.
- **Disabled.** Setting `Arbitration = Disabled` in the `[ndbd default]` section of the `config.ini` file to accomplishes the same task as setting `ArbitrationRank` to 0 on all management and API nodes. When `Arbitration` is set in this way, any `ArbitrationRank` settings are ignored.
- **WaitExternal.** The `Arbitration` parameter also makes it possible to configure arbitration in such a way that the cluster waits until after the time determined by `ArbitrationTimeout` has passed for an external cluster manager application to perform arbitration instead of handling arbitration internally. This can be done by setting `Arbitration = WaitExternal` in the `[ndbd default]` section of the `config.ini` file. For best results with the `WaitExternal` setting, it is recommended that `ArbitrationTimeout` be 2 times as long as the interval required by the external cluster manager to perform arbitration.

> **Important**
>
> This parameter should be used only in the `[ndbd default]` section of the cluster configuration file. The behavior of the cluster is unspecified when `Arbitration` is set to different values for individual data nodes.

Buffering and logging. Several `[ndbd]` configuration parameters enable the advanced user to have more control over the resources used by node processes and to adjust various buffer sizes at need.

These buffers are used as front ends to the file system when writing log records to disk. If the node is running in diskless mode, these parameters can be set to their minimum values without penalty due to the fact that disk writes are “faked” by the `NDB` storage engine's file system abstraction layer.

- `UndoIndexBuffer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	2M	1M - 4294967039 (0xFFFFFEFF)	N

The UNDO index buffer, whose size is set by this parameter, is used during local checkpoints. The `NDB` storage engine uses a recovery scheme based on checkpoint consistency in conjunction with an operational REDO log. To produce a consistent checkpoint without blocking the entire system for writes, UNDO logging is done while performing the local checkpoint. UNDO logging is activated on a single table fragment at a time. This optimization is possible because tables are stored entirely in main memory.

The UNDO index buffer is used for the updates on the primary key hash index. Inserts and deletes rearrange the hash index; the NDB storage engine writes UNDO log records that map all physical changes to an index page so that they can be undone at system restart. It also logs all active insert operations for each fragment at the start of a local checkpoint.

Reads and updates set lock bits and update a header in the hash index entry. These changes are handled by the page-writing algorithm to ensure that these operations need no UNDO logging.

This buffer is 2MB by default. The minimum value is 1MB, which is sufficient for most applications. For applications doing extremely large or numerous inserts and deletes together with large transactions and large primary keys, it may be necessary to increase the size of this buffer. If this buffer is too small, the NDB storage engine issues internal error code 677 (`Index UNDO buffers overloaded`).

> **Important**
>
> It is not safe to decrease the value of this parameter during a rolling restart.

- `UndoDataBuffer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	16M	1M - 4294967039 (0xFFFFFEFF)	N

This parameter sets the size of the UNDO data buffer, which performs a function similar to that of the UNDO index buffer, except the UNDO data buffer is used with regard to data memory rather than index memory. This buffer is used during the local checkpoint phase of a fragment for inserts, deletes, and updates.

Because UNDO log entries tend to grow larger as more operations are logged, this buffer is also larger than its index memory counterpart, with a default value of 16MB.

This amount of memory may be unnecessarily large for some applications. In such cases, it is possible to decrease this size to a minimum of 1MB.

It is rarely necessary to increase the size of this buffer. If there is such a need, it is a good idea to check whether the disks can actually handle the load caused by database update activity. A lack of sufficient disk space cannot be overcome by increasing the size of this buffer.

If this buffer is too small and gets congested, the NDB storage engine issues internal error code 891 (`Data UNDO buffers overloaded`).

> **Important**
>
> It is not safe to decrease the value of this parameter during a rolling restart.

- `RedoBuffer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	32M	1M - 4294967039 (0xFFFFFEFF)	N

All update activities also need to be logged. The REDO log makes it possible to replay these updates whenever the system is restarted. The NDB recovery algorithm uses a “fuzzy” checkpoint of the data together with the UNDO log, and then applies the REDO log to play back all changes up to the restoration point.

`RedoBuffer` sets the size of the buffer in which the REDO log is written. The default value is 32MB; the minimum value is 1MB.

If this buffer is too small, the `NDB` storage engine issues error code 1221 (`REDO log buffers overloaded`). For this reason, you should exercise care if you attempt to decrease the value of `RedoBuffer` as part of an online change in the cluster's configuration.

`ndbmtd` allocates a separate buffer for each LDM thread (see `ThreadConfig`). For example, with 4 LDM threads, an `ndbmtd` data node actually has 4 buffers and allocates `RedoBuffer` bytes to each one, for a total of `4 * RedoBuffer` bytes.

- `EventLogBufferSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	8192	0 - 64K	S

Controls the size of the circular buffer used for NDB log events within data nodes.

Controlling log messages. In managing the cluster, it is very important to be able to control the number of log messages sent for various event types to `stdout`. For each event category, there are 16 possible event levels (numbered 0 through 15). Setting event reporting for a given event category to level 15 means all event reports in that category are sent to `stdout`; setting it to 0 means that there will be no event reports made in that category.

By default, only the startup message is sent to `stdout`, with the remaining event reporting level defaults being set to 0. The reason for this is that these messages are also sent to the management server's cluster log.

An analogous set of levels can be set for the management client to determine which event levels to record in the cluster log.

- `LogLevelStartup`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	1	0 - 15	N

The reporting level for events generated during startup of the process.

The default level is 1.

- `LogLevelShutdown`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	0	0 - 15	N

The reporting level for events generated as part of graceful shutdown of a node.

The default level is 0.

- `LogLevelStatistic`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	0	0 - 15	N

The reporting level for statistical events such as number of primary key reads, number of updates, number of inserts, information relating to buffer usage, and so on.

The default level is 0.

- `LogLevelCheckpoint`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	log level	0	0 - 15	N

The reporting level for events generated by local and global checkpoints.

The default level is 0.

- `LogLevelNodeRestart`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	0	0 - 15	N

The reporting level for events generated during node restart.

The default level is 0.

- `LogLevelConnection`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	0	0 - 15	N

The reporting level for events generated by connections between cluster nodes.

The default level is 0.

- `LogLevelError`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	0	0 - 15	N

The reporting level for events generated by errors and warnings by the cluster as a whole. These errors do not cause any node failure but are still considered worth reporting.

The default level is 0.

- `LogLevelCongestion`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	levelr	0	0 - 15	N

The reporting level for events generated by congestion. These errors do not cause node failure but are still considered worth reporting.

The default level is 0.

- `LogLevelInfo`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	0	0 - 15	N

The reporting level for events generated for information about the general state of the cluster.

The default level is 0.

- `MemReportFrequency`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 4294967039 (0xFFFFFEFF)	N

This parameter controls how often data node memory usage reports are recorded in the cluster log; it is an integer value representing the number of seconds between reports.

Each data node's data memory and index memory usage is logged as both a percentage and a number of 32 KB pages of the DataMemory and IndexMemory, respectively, set in the config.ini file. For example, if DataMemory is equal to 100 MB, and a given data node is using 50 MB for data memory storage, the corresponding line in the cluster log might look like this:

```
2006-12-24 01:18:16 [MgmSrvr] INFO -- Node 2: Data usage is 50%(1280 32K pages of total 2560)
```

MemReportFrequency is not a required parameter. If used, it can be set for all cluster data nodes in the [ndbd default] section of config.ini, and can also be set or overridden for individual data nodes in the corresponding [ndbd] sections of the configuration file. The minimum value—which is also the default value—is 0, in which case memory reports are logged only when memory usage reaches certain percentages (80%, 90%, and 100%), as mentioned in the discussion of statistics events in Section 7.6.2, “MySQL Cluster Log Events”.

- StartupStatusReportFrequency

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	seconds	0	0 - 4294967039 (0xFFFFFEFF)	N

When a data node is started with the --initial, it initializes the redo log file during Start Phase 4 (see Section 7.1, “Summary of MySQL Cluster Start Phases”). When very large values are set for NoOfFragmentLogFiles, FragmentLogFileSize, or both, this initialization can take a long time.You can force reports on the progress of this process to be logged periodically, by means of the StartupStatusReportFrequency configuration parameter. In this case, progress is reported in the cluster log, in terms of both the number of files and the amount of space that have been initialized, as shown here:

```
2009-06-20 16:39:23 [MgmSrvr] INFO -- Node 1: Local redo log file initialization status:
#Total files: 80, Completed: 60
#Total MBytes: 20480, Completed: 15557
2009-06-20 16:39:23 [MgmSrvr] INFO -- Node 2: Local redo log file initialization status:
#Total files: 80, Completed: 60
#Total MBytes: 20480, Completed: 15570
```

These reports are logged each StartupStatusReportFrequency seconds during Start Phase 4. If StartupStatusReportFrequency is 0 (the default), then reports are written to the cluster log only when at the beginning and at the completion of the redo log file initialization process.

Debugging Parameters. In MySQL Cluster NDB 7.3 and later, it is possible to cause logging of traces for events generated by creating and dropping tables using DictTrace. This parameter is useful only in debugging NDB kernel code. DictTrace takes an integer value; currently, 0 (default - no logging) and 1 (logging enabled) are the only supported values.

Backup parameters. The [ndbd] parameters discussed in this section define memory buffers set aside for execution of online backups.

- BackupDataBufferSize

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	16M	0 - 4294967039 (0xFFFFFEFF)	N

In creating a backup, there are two buffers used for sending data to the disk. The backup data buffer is used to fill in data recorded by scanning a node's tables. Once this buffer has been filled to the level specified as BackupWriteSize, the pages are sent to disk. While flushing data to disk, the backup process can continue filling this buffer until it runs out of space. When this happens, the backup process pauses the scan and waits until some disk writes have completed freeing up memory so that scanning may continue.

The default value for this parameter is 16MB.

- `BackupDiskWriteSpeedPct`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.4.8	percent	50	0 - 90	N

During normal operation, data nodes attempt to maximize the disk write speed used for local checkpoints and backups while remaining within the bounds set by `MinDiskWriteSpeed` and `MaxDiskWriteSpeed`. In MySQL Cluster NDB 7.4, the implementation of disk write throttling has been changed to give each LDM thread an equal share of the total budget. This allows parallel LCPs to take place without exceeding the disk I/O budget. Because a backup is executed by only one LDM thread, this effectively caused a budget cut, resulting in longer backup completion times, and—if the rate of change is sufficiently high—in failure to complete the backup when the backup log buffer fill rate is higher than the achievable write rate.

This problem is addressed in MySQL Cluster NDB 7.4.8 and later by the addition of the `BackupDiskWriteSpeedPct` configuration parameter (Bug #20204854). This parameter takes a value in the range 0-90 (inclusive) which is interpreted as the percentage of the node's maximum write rate budget that is reserved prior to sharing out the remainder of the budget among LDM threads for LCPs. The LDM thread running the backup receives the whole write rate budget for the backup, plus its (reduced) share of the write rate budget for local checkpoints. This makes the disk write rate budget in NDB 7.4.8 and later behave similarly to how it is handled in MySQL Cluster NDB 7.3 and previous MySQL Cluster release series.

The default value for this parameter is 50 (interpreted as 50%).

- `BackupLogBufferSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	16M	0 - 4294967039 (0xFFFFFEFF)	N

The backup log buffer fulfills a role similar to that played by the backup data buffer, except that it is used for generating a log of all table writes made during execution of the backup. The same principles apply for writing these pages as with the backup data buffer, except that when there is no more space in the backup log buffer, the backup fails. For that reason, the size of the backup log buffer must be large enough to handle the load caused by write activities while the backup is being made. See Section 7.3.3, “Configuration for MySQL Cluster Backups”.

The default value for this parameter should be sufficient for most applications. In fact, it is more likely for a backup failure to be caused by insufficient disk write speed than it is for the backup log buffer to become full. If the disk subsystem is not configured for the write load caused by applications, the cluster is unlikely to be able to perform the desired operations.

It is preferable to configure cluster nodes in such a manner that the processor becomes the bottleneck rather than the disks or the network connections.

The default value for this parameter is 16MB.

- `BackupMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	32M	0 - 4294967039 (0xFFFFFEFF)	N

This parameter is simply the sum of `BackupDataBufferSize` and `BackupLogBufferSize`.

The default value of this parameter in MySQL Cluster NDB 7.3 and later is 16MB + 16MB = 32MB.

> **Important**
>
> If `BackupDataBufferSize` and `BackupLogBufferSize` taken together exceed the default value for `BackupMemory`, then this parameter must be set explicitly in the `config.ini` file to their sum.

- `BackupReportFrequency`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	seconds	0	0 - 4294967039 (0xFFFFFEFF)	N

This parameter controls how often backup status reports are issued in the management client during a backup, as well as how often such reports are written to the cluster log (provided cluster event logging is configured to permit it—see Logging and checkpointing). `BackupReportFrequency` represents the time in seconds between backup status reports.

The default value is 0.

- `BackupWriteSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	256K	2K - 4294967039 (0xFFFFFEFF)	N

This parameter specifies the default size of messages written to disk by the backup log and backup data buffers.

The default value for this parameter is 256KB.

- `BackupMaxWriteSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	1M	2K - 4294967039 (0xFFFFFEFF)	N

This parameter specifies the maximum size of messages written to disk by the backup log and backup data buffers.

The default value for this parameter is 1MB.

> **Important**
>
> When specifying these parameters, the following relationships must hold true. Otherwise, the data node will be unable to start.

- `BackupDataBufferSize >= BackupWriteSize + 188KB`
- `BackupLogBufferSize >= BackupWriteSize + 16KB`
- `BackupMaxWriteSize >= BackupWriteSize`

MySQL Cluster Realtime Performance Parameters

The `[ndbd]` parameters discussed in this section are used in scheduling and locking of threads to specific CPUs on multiprocessor data node hosts.

> **Note**
>
> To make use of these parameters, the data node process must be run as system root.

- `LockExecuteThreadToCPU`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	CPU ID	64K	0 - 64K	N

When used with `ndbd`, this parameter (now a string) specifies the ID of the CPU assigned to handle the `NDBCLUSTER` execution thread. When used with `ndbmtd`, the value of this parameter is a comma-separated list of CPU IDs assigned to handle execution threads. Each CPU ID in the list should be an integer in the range 0 to 65535 (inclusive).

The number of IDs specified should match the number of execution threads determined by `MaxNoOfExecutionThreads`. However, there is no guarantee that threads are assigned to CPUs in any given order when using this parameter. You can obtain more finely-grained control of this type using `ThreadConfig`.

`LockExecuteThreadToCPU` has no default value.

- `LockMaintThreadsToCPU`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	CPU ID	[none]	0 - 64K	N

This parameter specifies the ID of the CPU assigned to handle `NDBCLUSTER` maintenance threads.

The value of this parameter is an integer in the range 0 to 65535 (inclusive). In MySQL Cluster NDB 7.3 and later, there is no default value.

- `RealtimeScheduler`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Setting this parameter to 1 enables real-time scheduling of data node threads.

Prior to MySQL Cluster NDB 7.3.3, this parameter did not work correctly with data nodes running `ndbmtd`. (Bug #16961971)

The default is 0 (scheduling disabled).

- `SchedulerExecutionTimer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	µsec	50	0 - 11000	N

This parameter specifies the time in microseconds for threads to be executed in the scheduler before being sent. Setting it to 0 minimizes the response time; to achieve higher throughput, you can increase the value at the expense of longer response times.

The default is 50 µsec, which our testing shows to increase throughput slightly in high-load cases without materially delaying requests.

- `SchedulerSpinTimer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	µsec	0	0 - 500	N

This parameter specifies the time in microseconds for threads to be executed in the scheduler before sleeping.

The default value is 0.

- BuildIndexThreads

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	0	0 - 128	S

This parameter determines the number of threads to create when rebuilding ordered indexes during a system or node start, as well as when running `ndb_restore --rebuild-indexes` [313]. It is supported only when there is more than one fragment for the table per data node (for example, when the `MAX_ROWS` option has been used with `CREATE TABLE`).

Setting this parameter to 0 (the default) disables multi-threaded building of ordered indexes.

This parameter is supported when using `ndbd` or `ndbmtd`.

You can enable multi-threaded builds during data node initial restarts by setting the `TwoPassInitialNodeRestartCopy` data node configuration parameter to `TRUE`.

- TwoPassInitialNodeRestartCopy

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Multi-threaded building of ordered indexes can be enabled for initial restarts of data nodes by setting this configuration parameter to `TRUE`, which enables two-pass copying of data during initial node restarts.

You must also set `BuildIndexThreads` to a nonzero value.

- Numa

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	1	...	N

This parameter determines whether Non-Uniform Memory Access (NUMA) is controlled by the operating or by the data node process, whether the data node uses `ndbd` or `ndbmtd`. By default, `NDB` attempts to use an interleaved NUMA memory allocation policy on any data node where the host operating system provides NUMA support.

Setting `Numa = 0` means that the datanode process does not itself attempt to set a policy for memory allocation, and permits this behavior to be determined by the operating system, which may be further guided by the separate `numactl` tool. That is, `Numa = 0` yields the system default behavior, which can be customised by `numactl`. For many Linux systems, the system default behavior is to allocate socket-local memory to any given process at allocation time. This can be problematic when using `ndbmtd`; this is because `nbdmtd` allocates all memory at startup, leading to an imbalance, giving different access speeds for different sockets, especially when locking pages in main memory.

Setting `Numa = 1` means that the data node process uses `libnuma` to request interleaved memory allocation. (This can also be accomplished manually, on the operating system level, using `numactl`.)

Using interleaved allocation in effect tells the data node process to ignore non-uniform memory access but does not attempt to take any advantage of fast local memory; instead, the data node process tries to avoid imbalances due to slow remote memory. If interleaved allocation is not desired, set `Numa` to 0 so that the desired behavior can be determined on the operating system level.

The `Numa` configuration parameter is supported only on Linux systems where `libnuma.so` is available.

Multi-Threading Configuration Parameters (ndbmtd). `ndbmtd` runs by default as a single-threaded process and must be configured to use multiple threads, using either of two methods, both of which require setting configuration parameters in the `config.ini` file. The first method is simply to set an appropriate value for the `MaxNoOfExecutionThreads` configuration parameter. MySQL Cluster NDB 7.3 and later also support a second method, whereby it is possible to set up more complex rules for `ndbmtd` multi-threading using `ThreadConfig`. The next few paragraphs provide information about these parameters and their use with multi-threaded data nodes.

- `MaxNoOfExecutionThreads`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	integer	2	2 - 36	IS
NDB 7.3.3	integer	2	2 - 72	IS

This parameter directly controls the number of execution threads used by `ndbmtd`, up to a maximum of 72 (previous to MySQL Cluster NDB 7.3.3, this was 36). Although this parameter is set in `[ndbd]` or `[ndbd default]` sections of the `config.ini` file, it is exclusive to `ndbmtd` and does not apply to `ndbd`.

Setting `MaxNoOfExecutionThreads` sets the number of threads for each type as determined by a matrix in the file `storage/ndb/src/kernel/vm/mt_thr_config.cpp`. This table shows these numbers of threads for possible values of `MaxNoOfExecutionThreads` in MySQL Cluster NDB 7.4.3 and later (Bug #75220, Bug #20215689). (A table with information about the matrix applicable in previous versions of MySQL Cluster follows this one.) Rows containing values which changed in MySQL Cluster NDB 7.4.3 are shown in emphasized text.

`MaxNoOfExecutionThreads` Value	LDM Threads	TC Threads	Send Threads	Receive Threads
0 .. 3	1	1	0	1
4 .. 6	2	1	0	1
7 .. 8	4	1	0	1
9	4	2	0	1
10	4	2	1	1
11	4	3	1	1
12	*6*	*3*	*1*	*2*
13	*6*	*2*	*1*	*2*
14	*6*	*3*	*1*	*2*
15	*6*	*3*	*2*	*2*
16	8	3	1	2
17	8	4	1	2
18	8	4	2	2
19	8	5	2	2
20	*10*	*4*	*2*	*2*
21	*10*	*5*	*2*	*2*

MaxNoOfExecutionThreads Value	LDM Threads	TC Threads	Send Threads	Receive Threads
22	*10*	*5*	*2*	*3*
23	*10*	*6*	*2*	*3*
24	12	5	2	3
25	12	6	2	3
26	12	6	3	3
27	12	7	3	3
28	12	7	3	4
29	12	8	3	4
30	12	8	4	4
31	12	9	4	4
32	16	8	3	3
33	16	8	3	4
34	16	8	4	4
35	16	9	4	4
36	16	10	4	4
37	16	10	4	5
38	16	11	4	5
39	16	11	5	5
40	*20*	*10*	*4*	*4*
41	*20*	*10*	*4*	*5*
42	*20*	*11*	*4*	*5*
43	*20*	*11*	*5*	*5*
44	*20*	*12*	*5*	*5*
45	*20*	*12*	*5*	*6*
46	*20*	*13*	*5*	*6*
47	*20*	*13*	*6*	*6*
48	24	12	5	5
49	24	12	5	6
50	24	13	5	6
51	24	13	6	6
52	24	14	6	6
53	24	14	6	7
54	24	15	6	7
55	24	15	7	7
56	24	16	7	7
57	24	16	7	8
58	24	17	7	8
59	24	17	8	8
60	24	18	8	8
61	24	18	8	9

`MaxNoOfExecutionThreads` Value	LDM Threads	TC Threads	Send Threads	Receive Threads
62	24	19	8	9
63	24	19	9	9
64	32	16	7	7
65	32	16	7	8
66	32	17	7	8
67	32	17	8	8
68	32	18	8	8
69	32	18	8	9
70	32	19	8	9
71	32	20	8	9
72	32	20	8	10

The following table shows how the number of threads for each type is obtained for values of `MaxNoOfExecutionThreads` in MySQL Cluster NDB 7.4.2 and earlier. This table can be also used with MySQL Cluster NDB 7.3.2 and earlier, except that in these versions the maximum value for `MaxNoOfExecutionThreads` is 36, and thus rows from this table which correspond to values greater than 36 do not apply in versions prior to MySQL Cluster NDB 7.3.3.

`MaxNoOfExecutionThreads` Value	LDM Threads	TC Threads	Send Threads	Receive Threads
0 .. 3	1	1	0	1
4 .. 6	2	1	0	1
7 .. 8	4	1	0	1
9	4	2	0	1
10	4	2	1	1
11	4	3	1	1
12	4	3	1	2
13	4	3	2	2
14	4	4	2	2
15	4	5	2	2
16	8	3	1	2
17	8	4	1	2
18	8	4	2	2
19	8	5	2	2
20	8	5	2	3
21	8	5	3	3
22	8	6	3	3
23	8	7	3	3
24	12	5	2	3
25	12	6	2	3
26	12	6	3	3
27	12	7	3	3

`MaxNoOfExecutionThreads` Value	LDM Threads	TC Threads	Send Threads	Receive Threads
28	12	7	3	4
29	12	8	3	4
30	12	8	4	4
31	12	9	4	4
32	16	8	3	3
33	16	8	3	4
34	16	8	4	4
35	16	9	4	4
36	16	10	4	4
37	16	10	4	5
38	16	11	4	5
39	16	11	5	5
40	16	12	5	5
41	16	12	5	6
42	16	13	5	6
43	16	13	6	6
44	16	14	6	6
45	16	14	6	7
46	16	15	6	7
47	16	15	7	7
48	24	12	5	5
49	24	12	5	6
50	24	13	5	6
51	24	13	6	6
52	24	14	6	6
53	24	14	6	7
54	24	15	6	7
55	24	15	7	7
56	24	16	7	7
57	24	16	7	8
58	24	17	7	8
59	24	17	8	8
60	24	18	8	8
61	24	18	8	9
62	24	19	8	9
63	24	19	9	9
64	32	16	7	7
65	32	16	7	8
66	32	17	7	8
67	32	17	8	8

MaxNoOfExecutionThreads Value	LDM Threads	TC Threads	Send Threads	Receive Threads
68	32	18	8	8
69	32	18	8	9
70	32	19	8	9
71	32	20	8	9
72	32	20	8	10

In MySQL Cluster NDB 7.3 and later, there is always one SUMA (replication) thread.

The number of LDM threads must not exceed `NoOfFragmentLogParts`. If this parameter's value is the default (4), this means that you must increase it as well, when setting `MaxNoOfExecutionThreads` to 16 or greater; that is, you should set `NoOfFragmentLogParts` to the corresponding number of LDM threads value shown for that value of `MaxNoOfExecutionThreads` in the preceding table.

The thread types are described later in this section (see `ThreadConfig`).

Setting this parameter outside the permitted range of values causes the management server to abort on startup with the error `Error line *number*: Illegal value *value* for parameter MaxNoOfExecutionThreads`.

For `MaxNoOfExecutionThreads`, a value of 0 or 1 is rounded up internally by `NDB` to 2, so that 2 is considered this parameter's default and minimum value.

`MaxNoOfExecutionThreads` is generally intended to be set equal to the number of CPU threads available, and to allocate a number of threads of each type suitable to typical workloads. It does not assign particular threads to specified CPUs. For cases where it is desirable to vary from the settings provided, or to bind threads to CPUs, you should use `ThreadConfig` instead, which allows you to allocate each thread directly to a desired type, CPU, or both.

The multi-threaded data node process always spawns at least 5 threads, listed here:

- 1 local query handler (LDM) thread
- 1 transaction coordinator (TC) thread
- 1 send thread

  (It is possible to keep any separate send threads from being employed, as explained elsewhere in this section.)
- 1 receive thread
- 1 subscription manager (SUMA or replication) thread

Changing the number of LDM threads always requires a system restart, whether it is changed using this parameter or `ThreadConfig`. If the cluster's `IndexMemory` usage is greater than 50%, changing this requires an initial restart of the cluster. (A maximum of 30-35% `IndexMemory` usage is recommended in such cases.) Otherwise, resource usage and LDM thread allocation cannot be balanced between nodes, which can result in underutilized and overutilized LDM threads, and ultimately data node failures.

- `NoOfFragmentLogParts`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	4	4, 8, 12, 16	IN

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.3	numeric	4	4, 8, 12, 16, 24, 32	IN

Set the number of log file groups for redo logs belonging to this `ndbmtd`. Prior to MySQL Cluster NDB 7.3.3, this value must be an even multiple of 4 between 4 and 16, inclusive. In MySQL Cluster NDB 7.3.3 and later, the maximum is 32; the value must, as before, be an even multiple of 4.

The number of LQH threads used by `ndbmtd` must not exceed `NoOfFragmentLogParts`, and this number may increase when increasing `MaxNoOfExecutionThreads`; see the description of this parameter for more information.

- `ThreadConfig`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	''	...	IS

This parameter is used with `ndbmtd` to assign threads of different types to different CPUs. Its value is a string whose format has the following syntax:

```
ThreadConfig := entry[,entry[,...]]
entry := type={param[,param[,...]]}
type := ldm | main | recv | send | rep | io
param := count=number | cpubind=cpu_list
```

The curly braces (`{`...`}`) surrounding the list of parameters is required, even if there is only one parameter in the list.

A *`param`* (parameter) specifies the number of threads of the given type (`count`), the CPUs to which the threads of the given type are to be bound (`cpubind`), or both.

The *`type`* attribute represents an NDB thread type. The thread types supported in MySQL Cluster NDB 7.3 and later, and the range of permitted `count` values for each, are provided in the following list:

- `ldm`: Local query handler (`DBLQH` kernel block) that handles data. The more LDM threads that are used, the more highly partitioned the data becomes. Each LDM thread maintains its own sets of data and index partitions, as well as its own redo log. In MySQL Cluster NDB 7.3.3 and later, the maximum is 32 such threads; in MySQL Cluster NDB 7.3.2 and earlier, the maximum is 16.

  > **Important**
  >
  > Changing the number of LDM threads requires a system restart to be effective and safe for cluster operations. (This is also true when this is done using `MaxNoOfExecutionThreads`.) If `IndexMemory` usage is in excess of 50%, an initial restart of the cluster is required; a maximum of 30-35% `IndexMemory` usage is recommended in such cases. Otherwise, `IndexMemory` and `DataMemory` usage as well as the allocation of LDM threads cannot be balanced between nodes, which can ultimately lead to data node failures.

- `tc`: Transaction coordinator thread (`DBTC` kernel block) containing the state of an ongoing transaction. In MySQL Cluster NDB 7.3, the number of TC threads is configurable; a total of 32 is possible in MySQL Cluster NDB 7.3.3 and later; previously this was 16.

  Optimally, every new transaction can be assigned to a new TC thread. In most cases 1 TC thread per 2 LDM threads is sufficient to guarantee that this can happen. In cases where the number of writes is relatively small when compared to the number of reads, it is possible that only 1 TC thread per 4 LQH threads is required to maintain transaction states. Conversely, in applications that perform a great many updates, it may be necessary for the ratio of TC threads to LDM threads to approach 1 (for example, 3 TC threads to 4 LDM threads).

Range: (*NDB 7.3.3 and later*) 1 - 32; (*NDB 7.3.2 and earlier*) 1 - 16.

- `main`: Data dictionary and transaction coordinator (`DBDIH` and `DBTC` kernel blocks), providing schema management. This is always handled by a single dedicated thread.

  Range: 1 only.

- `recv`: Receive thread (`CMVMI` kernel block). Each receive thread handles one or more sockets for communicating with other nodes in a MySQL Cluster, with one socket per node. MySQL Cluster NDB 7.3 and later support multiple receive threads. In MySQL Cluster NDB 7.3.2 and earlier, the maximum is 8 such threads; in MySQL Cluster NDB 7.3.3 and later, the maximum is 16.

  Range: (*NDB 7.3.3 and later*) 1 - 16; (*NDB 7.3.2 and earlier*) 1 - 8.

- `send`: Send thread (`CMVMI` kernel block). To increase throughput, it is possible to perform sends from one or more separate, dedicated threads (maximum 8).

  Previously, all threads handled their own sending directly; this can still be made to happen by setting the number of send threads to 0 (this also happens when `MaxNoOfExecutionThreads` is set less than 10). While doing so can have an adeverse impact on throughput, it can also in some cases provide decreased latency.

  Range: (*NDB 7.3.3 and later*) 0 - 16; (*NDB 7.3.2 and earlier*) 0 - 8.

- `rep`: Replication thread (`SUMA` kernel block). Asynchronous replication operations are always handled by a single, dedicated thread.

  Range: 1 only.

- `io`: File system and other miscellaneous operations. These are not demanding tasks, and are always handled as a group by a single, dedicated I/O thread.

  Range: 1 only.

Simple examples:

```
# Example 1.
ThreadConfig=ldm={count=2,cpubind=1,2},main={cpubind=12},rep={cpubind=11}
# Example 2.
Threadconfig=main={cpubind=0},ldm={count=4,cpubind=1,2,5,6},io={cpubind=3}
```

It is usually desirable when configuring thread usage for a data node host to reserve one or more number of CPUs for operating system and other tasks. Thus, for a host machine with 24 CPUs, you might want to use 20 CPU threads (leaving 4 for other uses), with 8 LDM threads, 4 TC threads (half the number of LDM threads), 3 send threads, 3 receive threads, and 1 thread each for schema management, asynchronous replication, and I/O operations. (This is almost the same distribution of threads used when `MaxNoOfExecutionThreads` is set equal to 20.) The following `ThreadConfig` setting performs these assignments, additionally binding all of these threads to specific CPUs:

```
ThreadConfig=ldm{count=8,cpubind=1,2,3,4,5,6,7,8},main={cpubind=9},io={cpubind=9}, \
rep={cpubind=10},tc{count=4,cpubind=11,12,13,14},recv={count=3,cpubind=15,16,17}, \
send{count=3,cpubind=18,19,20}
```

It should be possible in most cases to bind the main (schema management) thread and the I/O thread to the same CPU, as we have done in the example just shown.

In order to take advantage of the enhanced stability that the use of `ThreadConfig` offers, it is necessary to insure that CPUs are isolated, and that they not subject to interrupts, or to being scheduled for other tasks by the operating system. On many Linux systems, you can do this by setting `IRQBALANCE_BANNED_CPUS` in `/etc/sysconfig/irqbalance` to `0xFFFFF0`, and by using the `isolcpus` boot option in `grub.conf`. For specific information, see your operating system or platform documentation.

Disk Data Configuration Parameters. Configuration parameters affecting Disk Data behavior include the following:

- `DiskPageBufferEntries`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.8	32K pages	10	1 - 1000	N
NDB 7.4.3	32K pages	10	1 - 1000	N

This is the number of page entries (page references) to allocate. It is specified as a number of 32K pages in `DiskPageBufferMemory`. The default is sufficient for most cases but you may need to increase the value of this parameter if you encounter problems with very large transactions on Disk Data tables. Each page entry requires approximately 100 bytes.

- `DiskPageBufferMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	64M	4M - 1T	N

This determines the amount of space used for caching pages on disk, and is set in the `[ndbd]` or `[ndbd default]` section of the `config.ini` file. It is measured in bytes. Each page takes up 32 KB. This means that MySQL Cluster Disk Data storage always uses *`N`* * 32 KB memory where *`N`* is some nonnegative integer.

The default value for this parameter is `64M` (2000 pages of 32 KB each).

You can query the `ndbinfo.diskpagebuffer` table to help determine whether the value for this parameter should be increased to minimize unnecessary disk seeks. See Section 7.10.12, “The ndbinfo diskpagebuffer Table”, for more information.

- `SharedGlobalMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	128M	0 - 64T	N

This parameter determines the amount of memory that is used for log buffers, disk operations (such as page requests and wait queues), and metadata for tablespaces, log file groups, `UNDO` files, and data files. The shared global memory pool also provides memory used for satisfying the memory requirements of the `UNDO_BUFFER_SIZE` option used with `CREATE LOGFILE GROUP` and `ALTER LOGFILE GROUP` statements, including any default value implied for this options by the setting of the `InitialLogFileGroup` data node configuration parameter. `SharedGlobalMemory` can be set in the `[ndbd]` or `[ndbd default]` section of the `config.ini` configuration file, and is measured in bytes.

The default value is `128M`.

- `DiskIOThreadPool`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	threads	2	0 - 4294967039 (0xFFFFFEFF)	N

This parameter determines the number of unbound threads used for Disk Data file access. Before `DiskIOThreadPool` was introduced, exactly one thread was spawned for each Disk Data file, which could lead to performance issues, particularly when using very large data files. With `DiskIOThreadPool`, you can—for example—access a single large data file using several threads working in parallel.

This parameter applies to Disk Data I/O threads only.

The optimum value for this parameter depends on your hardware and configuration, and includes these factors:

- **Physical distribution of Disk Data files.** You can obtain better performance by placing data files, undo log files, and the data node file system on separate physical disks. If you do this with some or all of these sets of files, then you can set `DiskIOThreadPool` higher to enable separate threads to handle the files on each disk.

- **Disk performance and types.** The number of threads that can be accommodated for Disk Data file handling is also dependent on the speed and throughput of the disks. Faster disks and higher throughput allow for more disk I/O threads. Our test results indicate that solid-state disk drives can handle many more disk I/O threads than conventional disks, and thus higher values for `DiskIOThreadPool`.

The default value for this parameter is 2.

- **Disk Data file system parameters.** The parameters in the following list make it possible to place MySQL Cluster Disk Data files in specific directories without the need for using symbolic links.

  - `FileSystemPathDD`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	filename	[see text]	...	IN

If this parameter is specified, then MySQL Cluster Disk Data data files and undo log files are placed in the indicated directory. This can be overridden for data files, undo log files, or both, by specifying values for `FileSystemPathDataFiles`, `FileSystemPathUndoFiles`, or both, as explained for these parameters. It can also be overridden for data files by specifying a path in the `ADD DATAFILE` clause of a `CREATE TABLESPACE` or `ALTER TABLESPACE` statement, and for undo log files by specifying a path in the `ADD UNDOFILE` clause of a `CREATE LOGFILE GROUP` or `ALTER LOGFILE GROUP` statement. If `FileSystemPathDD` is not specified, then `FileSystemPath` is used.

If a `FileSystemPathDD` directory is specified for a given data node (including the case where the parameter is specified in the `[ndbd default]` section of the `config.ini` file), then starting that data node with `--initial` causes all files in the directory to be deleted.

  - `FileSystemPathDataFiles`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	filename	[see text]	...	IN

If this parameter is specified, then MySQL Cluster Disk Data data files are placed in the indicated directory. This overrides any value set for `FileSystemPathDD`. This parameter can be overridden for a given data file by specifying a path in the `ADD DATAFILE` clause of a `CREATE TABLESPACE` or `ALTER TABLESPACE` statement used to create that data file. If `FileSystemPathDataFiles` is not specified, then `FileSystemPathDD` is used (or `FileSystemPath`, if `FileSystemPathDD` has also not been set).

If a FileSystemPathDataFiles directory is specified for a given data node (including the case where the parameter is specified in the [ndbd default] section of the config.ini file), then starting that data node with --initial causes all files in the directory to be deleted.

- FileSystemPathUndoFiles

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	filename	[see text]	...	IN

If this parameter is specified, then MySQL Cluster Disk Data undo log files are placed in the indicated directory. This overrides any value set for FileSystemPathDD. This parameter can be overridden for a given data file by specifying a path in the ADD UNDO clause of a CREATE LOGFILE GROUP or ALTER LOGFILE GROUP statement used to create that data file. If FileSystemPathUndoFiles is not specified, then FileSystemPathDD is used (or FileSystemPath, if FileSystemPathDD has also not been set).

If a FileSystemPathUndoFiles directory is specified for a given data node (including the case where the parameter is specified in the [ndbd default] section of the config.ini file), then starting that data node with --initial causes all files in the directory to be deleted.

For more information, see Section 7.12.1, “MySQL Cluster Disk Data Objects”.

- **Disk Data object creation parameters.** The next two parameters enable you—when starting the cluster for the first time—to cause a Disk Data log file group, tablespace, or both, to be created without the use of SQL statements.

- InitialLogFileGroup

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	[see text]	...	S

This parameter can be used to specify a log file group that is created when performing an initial start of the cluster. InitialLogFileGroup is specified as shown here:

```
InitialLogFileGroup = [name=name;] [undo_buffer_size=size;] file-specification-list
file-specification-list:
    file-specification[; file-specification[; ...]]
file-specification:
    filename:size
```

The name of the log file group is optional and defaults to DEFAULT-LG. The undo_buffer_size is also optional; if omitted, it defaults to 64M. Each *file-specification* corresponds to an undo log file, and at least one must be specified in the *file-specification-list*. Undo log files are placed according to any values that have been set for FileSystemPath, FileSystemPathDD, and FileSystemPathUndoFiles, just as if they had been created as the result of a CREATE LOGFILE GROUP or ALTER LOGFILE GROUP statement.

Consider the following:

```
InitialLogFileGroup = name=LG1; undo_buffer_size=128M; undo1.log:250M; undo2.log:150M
```

This is equivalent to the following SQL statements:

```
CREATE LOGFILE GROUP LG1
    ADD UNDOFILE 'undo1.log'
    INITIAL_SIZE 250M
    UNDO_BUFFER_SIZE 128M
    ENGINE NDBCLUSTER;
```

```
ALTER LOGFILE GROUP LG1
    ADD UNDOFILE 'undo2.log'
    INITIAL_SIZE 150M
    ENGINE NDBCLUSTER;
```

This logfile group is created when the data nodes are started with `--initial`.

Prior to MySQL Cluster NDB 7.3.6, resources for the initial log file group are taken from the global memory pool whose size is determined by the value of the `SharedGlobalMemory` data node configuration parameter; in these versions, if this parameter is set too low and the values set in `InitialLogFileGroup` for the logfile group's initial size or undo buffer size are too high, the cluster may fail to create the default log file group when starting, or fail to start altogether. In MySQL Cluster NDB 7.3.6 and later, resources for the initial log file group are added to the global memory pool along with those indicated by the value of `SharedGlobalMemory` (Bug #11762867).

This parameter, if used, should always be set in the `[ndbd default]` section of the `config.ini` file. The behavior of a MySQL Cluster when different values are set on different data nodes is not defined.

- `InitialTablespace`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	[see text]	...	S

This parameter can be used to specify a MySQL Cluster Disk Data tablespace that is created when performing an initial start of the cluster. `InitialTablespace` is specified as shown here:

```
InitialTablespace = [name=name;] [extent_size=size;] file-specification-list
```

The `name` of the tablespace is optional and defaults to `DEFAULT-TS`. The `extent_size` is also optional; it defaults to `1M`. The *`file-specification-list`* uses the same syntax as shown with the `InitialLogfileGroup` parameter, the only difference being that each *`file-specification`* used with `InitialTablespace` corresponds to a data file. At least one must be specified in the *`file-specification-list`*. Data files are placed according to any values that have been set for `FileSystemPath`, `FileSystemPathDD`, and `FileSystemPathDataFiles`, just as if they had been created as the result of a `CREATE TABLESPACE` or `ALTER TABLESPACE` statement.

For example, consider the following line specifying `InitialTablespace` in the `[ndbd default]` section of the `config.ini` file (as with `InitialLogfileGroup`, this parameter should always be set in the `[ndbd default]` section, as the behavior of a MySQL Cluster when different values are set on different data nodes is not defined):

```
InitialTablespace = name=TS1; extent_size=8M; data1.dat:2G; data2.dat:4G
```

This is equivalent to the following SQL statements:

```
CREATE TABLESPACE TS1
    ADD DATAFILE 'data1.dat'
    EXTENT_SIZE 8M
    INITIAL_SIZE 2G
    ENGINE NDBCLUSTER;
ALTER TABLESPACE TS1
    ADD DATAFILE 'data2.dat'
    INITIAL_SIZE 4G
    ENGINE NDBCLUSTER;
```

This tablespace is created when the data nodes are started with `--initial`, and can be used whenever creating MySQL Cluster Disk Data tables thereafter.

Disk Data and GCP Stop errors. Errors encountered when using Disk Data tables such as `Node nodeid killed this node because GCP stop was detected` (error 2303) are often referred to as “GCP stop errors”. Such errors occur when the redo log is not flushed to disk quickly enough; this is usually due to slow disks and insufficient disk throughput.

You can help prevent these errors from occurring by using faster disks, and by placing Disk Data files on a separate disk from the data node file system. Reducing the value of `TimeBetweenGlobalCheckpoints` tends to decrease the amount of data to be written for each global checkpoint, and so may provide some protection against redo log buffer overflows when trying to write a global checkpoint; however, reducing this value also permits less time in which to write the GCP, so this must be done with caution.

In addition to the considerations given for `DiskPageBufferMemory` as explained previously, it is also very important that the `DiskIOThreadPool` configuration parameter be set correctly; having `DiskIOThreadPool` set too high is very likely to cause GCP stop errors (Bug #37227).

GCP stops can be caused by save or commit timeouts; the `TimeBetweenEpochsTimeout` data node configuration parameter determines the timeout for commits. However, it is possible to disable both types of timeouts by setting this parameter to 0.

Parameters for configuring send buffer memory allocation. Send buffer memory is allocated dynamically from a memory pool shared between all transporters, which means that the size of the send buffer can be adjusted as necessary. (Previously, the NDB kernel used a fixed-size send buffer for every node in the cluster, which was allocated when the node started and could not be changed while the node was running.) The `TotalSendBufferMemory` and `OverLoadLimit` data node configuration parameters permit the setting of limits on this memory allocation. For more information about the use of these parameters (as well as `SendBufferMemory`), see Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”.

- `ExtraSendBufferMemory`

  This parameter specifies the amount of transporter send buffer memory to allocate in addition to any set using `TotalSendBufferMemory`, `SendBufferMemory`, or both.

- `TotalSendBufferMemory`

  This parameter is used to determine the total amount of memory to allocate on this node for shared send buffer memory among all configured transporters.

  If this parameter is set, its minimum permitted value is 256KB; 0 indicates that the parameter has not been set. For more detailed information, see Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”.

- `ReservedSendBufferMemory`

  This parameter is present in `NDBCLUSTER` source code beginning with MySQL Cluster NDB 6.4.0. However, it is not currently enabled.

  This parameter was deprecated in MySQL Cluster NDB 7.2, and is subject to removal in a future release of MySQL Cluster (Bug #11760629, Bug #53053).

For more detailed information about the behavior and use of `TotalSendBufferMemory` and `ReservedSendBufferMemory`, and about configuring send buffer memory parameters in MySQL Cluster, see Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”.

See also Section 7.13, “Adding MySQL Cluster Data Nodes Online”.

Redo log over-commit handling. It is possible to control a data node's handling of operations when too much time is taken flushing redo logs to disk. This occurs when a given redo log flush takes longer than `RedoOverCommitLimit` seconds, more than `RedoOverCommitCounter` times, causing any pending transactions to be aborted. When this happens, the API node that sent the transaction can handle the operations that should have been committed either by queuing the operations and re-trying them, or by aborting them, as determined by `DefaultOperationRedoProblemAction`. The data node configuration parameters for setting the timeout and number of times it may be exceeded before the API node takes this action are described in the following list:

- `RedoOverCommitCounter`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	3	0 - 4294967039 (0xFFFFFEFF)	N

When `RedoOverCommitLimit` is exceeded when trying to write a given redo log to disk this many times or more, any transactions that were not committed as a result are aborted, and an API node where any of these transactions originated handles the operations making up those transactions according to its value for `DefaultOperationRedoProblemAction` (by either queuing the operations to be re-tried, or aborting them).

`RedoOverCommitCounter` defaults to 3. Set it to 0 to disable the limit.

- `RedoOverCommitLimit`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	seconds	20	0 - 4294967039 (0xFFFFFEFF)	N

This parameter sets an upper limit in seconds for trying to write a given redo log to disk before timing out. The number of times the data node tries to flush this redo log, but takes longer than `RedoOverCommitLimit`, is kept and compared with `RedoOverCommitCounter`, and when flushing takes too long more times than the value of that parameter, any transactions that were not committed as a result of the flush timeout are aborted. When this occurs, the API node where any of these transactions originated handles the operations making up those transactions according to its `DefaultOperationRedoProblemAction` setting (it either queues the operations to be re-tried, or aborts them).

By default, `RedoOverCommitLimit` is 20 seconds. Set to 0 to disable checking for redo log flush timeouts. This parameter was added in MySQL Cluster NDB 7.1.10.

Controlling restart attempts. It is possible to exercise finely-grained control over restart attempts by data nodes when they fail to start using the `MaxStartFailRetries` and `StartFailRetryDelay` data node configuration parameters.

`MaxStartFailRetries` limits the total number of retries made before giving up on starting the data node, `StartFailRetryDelay` sets the number of seconds between retry attempts. These parameters are listed here:

- `StartFailRetryDelay`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 4294967039 (0xFFFFFEFF)	N

Use this parameter to set the number of seconds between restart attempts by the data node in the event on failure on startup. The default is 0 (no delay).

Both this parameter and `MaxStartFailRetries` are ignored unless `StopOnError` is equal to 0.

- `MaxStartFailRetries`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	3	0 - 4294967039 (0xFFFFFEFF)	N

Use this parameter to limit the number restart attempts made by the data node in the event that it fails on startup. The default is 3 attempts.

Both this parameter and `StartFailRetryDelay` are ignored unless `StopOnError` is equal to 0.

NDB index statistics parameters. The parameters in the following list relate to NDB index statistics generation, which was introduced in MySQL Cluster NDB 7.2.1.

- `IndexStatAutoCreate`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	false, true	S

Enable or disable automatic statistics collection when indexes are created. Disabled by default.

This parameter was added in MySQL Cluster NDB 7.2.1.

- `IndexStatAutoUpdate`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	false, true	S

Enable or disable monitoring of indexes for changes and trigger automatic statistics updates these are detected. The amount and degree of change needed to trigger the updates are determined by the settings for the `IndexStatTriggerPct` and `IndexStatTriggerScale` options.

This parameter was added in MySQL Cluster NDB 7.2.1.

- `IndexStatSaveSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	32768	0 - 4294967039 (0xFFFFFEFF)	IN

Maximum space in bytes allowed for the saved statistics of any given index in the `NDB` system tables and in the `mysqld` memory cache. This consumes `IndexMemory`.

At least one sample is always produced, regardless of any size limit. This size is scaled by `IndexStatSaveScale`.

This parameter was added in MySQL Cluster NDB 7.2.1.

The size specified by `IndexStatSaveSize` is scaled by the value of `IndexStatTriggerPct` for a large index, times 0.01. This is further multiplied by the logarithm to the base 2 of the index size. Setting `IndexStatTriggerPct` equal to 0 disables the scaling effect.

- `IndexStatSaveScale`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	percentage	100	0 - 4294967039 (0xFFFFFEFF)	IN

The size specified by IndexStatSaveSize is scaled by the value of IndexStatTriggerPct for a large index, times 0.01. This is further multiplied by the logarithm to the base 2 of the index size. Setting IndexStatTriggerPct equal to 0 disables the scaling effect.

- IndexStatTriggerPct

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	percentage	100	0 - 4294967039 (0xFFFFFEFF)	IN

Percentage change in updates that triggers an index statistics update. The value is scaled by IndexStatTriggerScale. You can disable this trigger altogether by setting IndexStatTriggerPct to 0.

This parameter was added in MySQL Cluster NDB 7.2.1.

- IndexStatTriggerScale

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	percentage	100	0 - 4294967039 (0xFFFFFEFF)	IN

Scale IndexStatTriggerPct by this amount times 0.01 for a large index. A value of 0 disables scaling.

This parameter was added in MySQL Cluster NDB 7.2.1.

- IndexStatUpdateDelay

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	seconds	60	0 - 4294967039 (0xFFFFFEFF)	IN

Minimum delay in seconds between automatic index statistics updates for a given index. Setting this variable to 0 disables any delay. The default is 60 seconds.

This parameter was added in MySQL Cluster NDB 7.2.1.

5.3.7 Defining SQL and Other API Nodes in a MySQL Cluster

The [mysqld] and [api] sections in the config.ini file define the behavior of the MySQL servers (SQL nodes) and other applications (API nodes) used to access cluster data. None of the parameters shown is required. If no computer or host name is provided, any host can use this SQL or API node.

Generally speaking, a [mysqld] section is used to indicate a MySQL server providing an SQL interface to the cluster, and an [api] section is used for applications other than mysqld processes accessing cluster data, but the two designations are actually synonymous; you can, for instance, list parameters for a MySQL server acting as an SQL node in an [api] section.

> **Note**
>
> For a discussion of MySQL server options for MySQL Cluster, see Section 5.3.8.1, “MySQL Server Options for MySQL Cluster”; for information about MySQL server system variables relating to MySQL Cluster, see Section 5.3.8.2, “MySQL Cluster System Variables”.

- Id

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 255	IS

The `Id` is an integer value used to identify the node in all cluster internal messages. The permitted range of values is 1 to 255 inclusive. This value must be unique for each node in the cluster, regardless of the type of node.

> **Note**
>
> Data node IDs must be less than 49, regardless of the MySQL Cluster version used. If you plan to deploy a large number of data nodes, it is a good idea to limit the node IDs for API nodes (and management nodes) to values greater than 48.

`NodeId` is the preferred parameter name to use when identifying API nodes. (`Id` continues to be supported for backward compatibility, but is now deprecated and generates a warning when used. It is also subject to future removal.)

- `ConnectionMap`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	[none]	...	N

Specifies which data nodes to connect.

- `NodeId`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	1 - 255	IS

The `NodeId` is an integer value used to identify the node in all cluster internal messages. The permitted range of values is 1 to 255 inclusive. This value must be unique for each node in the cluster, regardless of the type of node.

> **Note**
>
> Data node IDs must be less than 49, regardless of the MySQL Cluster version used. If you plan to deploy a large number of data nodes, it is a good idea to limit the node IDs for API nodes (and management nodes) to values greater than 48.

`NodeId` is the preferred parameter name to use when identifying management nodes. An alias, `Id`, was used for this purpose in very old versions of MySQL Cluster, and continues to be supported for backward compatibility; it is now deprecated and generates a warning when used, and is subject to removal in a future release of MySQL Cluster.

- `ExecuteOnComputer`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name	[none]	...	S

This refers to the `Id` set for one of the computers (hosts) defined in a `[computer]` section of the configuration file.

- `HostName`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

Specifying this parameter defines the hostname of the computer on which the SQL node (API node) is to reside. To specify a hostname, either this parameter or `ExecuteOnComputer` is required.

If no `HostName` or `ExecuteOnComputer` is specified in a given `[mysql]` or `[api]` section of the `config.ini` file, then an SQL or API node may connect using the corresponding "slot" from any host which can establish a network connection to the management server host machine. *This differs from the default behavior for data nodes, where `localhost` is assumed for `HostName` unless otherwise specified*.

- `ArbitrationRank`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	0-2	0	0 - 2	N

This parameter defines which nodes can act as arbitrators. Both management nodes and SQL nodes can be arbitrators. A value of 0 means that the given node is never used as an arbitrator, a value of 1 gives the node high priority as an arbitrator, and a value of 2 gives it low priority. A normal configuration uses the management server as arbitrator, setting its `ArbitrationRank` to 1 (the default for management nodes) and those for all SQL nodes to 0 (the default for SQL nodes).

By setting `ArbitrationRank` to 0 on all management and SQL nodes, you can disable arbitration completely. You can also control arbitration by overriding this parameter; to do so, set the `Arbitration` parameter in the `[ndbd default]` section of the `config.ini` global configuration file.

- `ArbitrationDelay`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	milliseconds	0	0 - 4294967039 (0xFFFFFEFF)	N

Setting this parameter to any other value than 0 (the default) means that responses by the arbitrator to arbitration requests will be delayed by the stated number of milliseconds. It is usually not necessary to change this value.

- `BatchByteSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	16K	1024 - 1M	N

For queries that are translated into full table scans or range scans on indexes, it is important for best performance to fetch records in properly sized batches. It is possible to set the proper size both in terms of number of records (`BatchSize`) and in terms of bytes (`BatchByteSize`). The actual batch size is limited by both parameters.

The speed at which queries are performed can vary by more than 40% depending upon how this parameter is set.

This parameter is measured in bytes. The default value in MySQL Cluster NDB 7.3 and later is 16K.

- `BatchSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	records	256	1 - 992	N

This parameter is measured in number of records and is by default set to 256. The maximum size is 992.

- `ExtraSendBufferMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	0	0 - 4294967039 (0xFFFFFEFF)	N

This parameter specifies the amount of transporter send buffer memory to allocate in addition to any that has been set using `TotalSendBufferMemory`, `SendBufferMemory`, or both.

- `HeartbeatThreadPriority`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	string	[none]	...	S

Use this parameter to set the scheduling policy and priority of heartbeat threads for management and API nodes. The syntax for setting this parameter is shown here:

```
HeartbeatThreadPriority = policy[, priority]
policy:
  {FIFO | RR}
```

When setting this parameter, you must specify a policy. This is one of `FIFO` (first in, first in) or `RR` (round robin). This followed optionally by the priority (an integer).

- `MaxScanBatchSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	256K	32K - 16M	N

The batch size is the size of each batch sent from each data node. Most scans are performed in parallel to protect the MySQL Server from receiving too much data from many nodes in parallel; this parameter sets a limit to the total batch size over all nodes.

The default value of this parameter is set to 256KB. Its maximum size is 16MB.

- `TotalSendBufferMemory`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	0	256K - 4294967039 (0xFFFFFEFF)	N

This parameter is used to determine the total amount of memory to allocate on this node for shared send buffer memory among all configured transporters.

If this parameter is set, its minimum permitted value is 256KB; 0 indicates that the parameter has not been set. For more detailed information, see Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”.

- `AutoReconnect`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

This parameter is `false` by default. This forces disconnected API nodes (including MySQL Servers acting as SQL nodes) to use a new connection to the cluster rather than attempting to re-use an existing one, as re-use of connections can cause problems when using dynamically-allocated node IDs. (Bug #45921)

> **Note**
>
> This parameter can be overridden using the NDB API. For more information, see Ndb_cluster_connection::set_auto_reconnect(), and Ndb_cluster_connection::get_auto_reconnect().

- `DefaultOperationRedoProblemAction`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	enumeration	QUEUE	ABORT, QUEUE	S

This parameter (along with `RedoOverCommitLimit` and `RedoOverCommitCounter`) controls the data node's handling of operations when too much time is taken flushing redo logs to disk. This occurs when a given redo log flush takes longer than `RedoOverCommitLimit` seconds, more than `RedoOverCommitCounter` times, causing any pending transactions to be aborted.

When this happens, the node can respond in either of two ways, according to the value of `DefaultOperationRedoProblemAction`, listed here:

- `ABORT`: Any pending operations from aborted transactions are also aborted.
- `QUEUE`: Pending operations from transactions that were aborted are queued up to be re-tried. This the default. In MySQL Cluster NDB 7.3.10 and later as well as MySQL Cluster NDB 7.4.7 and later, pending operations are still aborted when the redo log runs out of space—that is, when **`P_TAIL_PROBLEM`** errors occur. (Bug #20782580)

- `DefaultHashMapSize`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	buckets	3840	0 - 3840	N

MySQL Cluster NDB 7.2.7 and later use a larger default table hash map size (3840) than in previous releases (240). Beginning with MySQL Cluster NDB 7.2.11, the size of the table hash maps used by `NDB` is configurable using this parameter; previously this value was hard-coded. `DefaultHashMapSize` can take any of three possible values (0, 240, 3840). These values and their effects are described in the following table.

Value	Description / Effect
`0`	Use the lowest value set, if any, for this parameter among all data nodes and API nodes in the cluster; if it is not set on any data or API node, use the default value.
`240`	Original hash map size, used by default in all MySQL Cluster releases prior to MySQL Cluster NDB 7.2.7.
`3840`	Larger hash map size as used by default in MySQL Cluster NDB 7.2.7 and later

The primary intended use for this parameter is to facilitate upgrades and especially downgrades between MySQL Cluster NDB 7.2.7 and later MySQL Cluster versions, in which the larger hash map size (3840) is the default, and earlier releases (in which the default was 240), due to the fact that this change is not otherwise backward compatible (Bug #14800539). By setting this parameter to 240 prior to performing an upgrade from an older version where this value is in use, you can cause the cluster to continue using the smaller size for table hash maps, in which case the tables remain compatible with earlier versions following the upgrade. `DefaultHashMapSize` can be set for individual data nodes, API nodes, or both, but setting it once only, in the `[ndbd default]` section of the `config.ini` file, is the recommended practice.

After increasing this parameter, to have existing tables to take advantage of the new size, you can run `ALTER TABLE ... REORGANIZE PARTITION` on them, after which they can use the larger hash map size. This is in addition to performing a rolling restart, which makes the larger hash maps available to new tables, but does not enable existing tables to use them.

Decreasing this parameter online after any tables have been created or modified with `DefaultHashMapSize` equal to 3840 is not currently supported.

- `Wan`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

Use WAN TCP setting as default.

- `ConnectBackoffMaxTime`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.7	integer	0	0 - 4294967039 (0xFFFFFEFF)	N
NDB 7.4.2	integer	0	0 - 4294967039 (0xFFFFFEFF)	N

Starting with MySQL Cluster NDB 7.3.7 and MySQL Cluster NDB 7.4.2, in a MySQL Cluster with many unstarted data nodes, the value of this parameter can be raised to circumvent connection attempts to data nodes which have not yet begun to function in the cluster, as well as moderate high traffic to management nodes. As long as the API node is not connected to any new data nodes, the value of the `StartConnectBackoffMaxTime` parameter is applied; otherwise, `ConnectBackoffMaxTime` is used to determine the length of time in milliseconds to wait between connection attempts.

Time elapsed *during* node connection attempts is not taken into account when calculating elapsed time for this parameter. The timeout is applied with approximately 100 ms resolution, starting with a 100 ms delay; for each subsequent attempt, the length of this period is doubled until it reaches `ConnectBackoffMaxTime` milliseconds, up to a maximum of 100000 ms (100s).

Once the API node is connected to a data node and that node reports (in a heartbeat message) that it has connected to other data nodes, connection attempts to those data nodes are no longer affected by this parameter, and are made every 100 ms thereafter until connected. Once a data node has started, it can take up `HeartbeatIntervalDbApi` for the API node to be notified that this has occurred.

- `StartConnectBackoffMaxTime`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.7	integer	0	0 - 4294967039 (0xFFFFFEFF)	N
NDB 7.4.2	integer	0	0 - 4294967039 (0xFFFFFEFF)	N

Starting with MySQL Cluster NDB 7.3.7 and MySQL Cluster NDB 7.4.2, in a MySQL Cluster with many unstarted data nodes, the value of this parameter can be raised to circumvent connection attempts to data nodes which have not yet begun to function in the cluster, as well as moderate high traffic to management nodes. As long as the API node is not connected to any new data nodes, the value of the `StartConnectBackoffMaxTime` parameter is applied; otherwise, `ConnectBackoffMaxTime` is used to determine the length of time in milliseconds to wait between connection attempts.

Time elapsed *during* node connection attempts is not taken into account when calculating elapsed time for this parameter. The timeout is applied with approximately 100 ms resolution, starting with a 100 ms delay; for each subsequent attempt, the length of this period is doubled until it reaches `StartConnectBackoffMaxTime` milliseconds, up to a maximum of 100000 ms (100s).

Once the API node is connected to a data node and that node reports (in a heartbeat message) that it has connected to other data nodes, connection attempts to those data nodes are no longer affected by this parameter, and are made every 100 ms thereafter until connected. Once a data node has started, it can take up `HeartbeatIntervalDbApi` for the API node to be notified that this has occurred.

You can also obtain information from a MySQL server running as a MySQL Cluster SQL node using `SHOW STATUS` in the `mysql` client, as shown here:

```
mysql> SHOW STATUS LIKE 'ndb%';
+-----------------------------+---------------+
| Variable_name               | Value         |
+-----------------------------+---------------+
| Ndb_cluster_node_id         | 5             |
| Ndb_config_from_host        | 192.168.0.112 |
| Ndb_config_from_port        | 1186          |
| Ndb_number_of_storage_nodes | 4             |
+-----------------------------+---------------+
4 rows in set (0.02 sec)
```

For information about the status variables appearing in the output from this statement, see Section 5.3.8.3, “MySQL Cluster Status Variables”.

> **Note**
>
> To add new SQL or API nodes to the configuration of a running MySQL Cluster, it is necessary to perform a rolling restart of all cluster nodes after adding new `[mysqld]` or `[api]` sections to the `config.ini` file (or files, if you are using more than one management server). This must be done before the new SQL or API nodes can connect to the cluster.
>
> It is *not* necessary to perform any restart of the cluster if new SQL or API nodes can employ previously unused API slots in the cluster configuration to connect to the cluster.

5.3.8 MySQL Server Options and Variables for MySQL Cluster

This section provides information about MySQL server options, server and status variables that are specific to MySQL Cluster. For general information on using these, and for other options and variables not specific to MySQL Cluster, see The MySQL Server.

For MySQL Cluster configuration parameters used in the cluster configuration file (usually named `config.ini`), see Chapter 5, *Configuration of MySQL Cluster*.

5.3.8.1 MySQL Server Options for MySQL Cluster

This section provides descriptions of mysqld server options relating to MySQL Cluster. For information about mysqld options not specific to MySQL Cluster, and for general information about the use of options with mysqld, see Server Command Options.

For information about command-line options used with other MySQL Cluster processes (ndbd, ndb_mgmd, and ndb_mgm), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”. For information about command-line options used with NDB utility programs (such as ndb_desc, ndb_size.pl, and ndb_show_tables), see Chapter 6, *MySQL Cluster Programs*.

- --ndb-batch-size=#

Table 5.10 Type and value information for ndb-batch-size

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb-batch-size		
Yes	Yes	No
Yes	Global	No
NDB 7.3	integer	32768 / 0 - 31536000
DESCRIPTION: Size (in bytes) to use for NDB transaction batches		

This sets the size in bytes that is used for NDB transaction batches.

- --ndb-cluster-connection-pool=#

Table 5.11 Type and value information for ndb-cluster-connection-pool

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb-cluster-connection-pool		
Yes	Yes	Yes
Yes	Global	No
NDB 7.3	integer	1 / 1 - 63
DESCRIPTION: Number of connections to the cluster used by MySQL		

By setting this option to a value greater than 1 (the default), a mysqld process can use multiple connections to the cluster, effectively mimicking several SQL nodes. Each connection requires its own [api] or [mysqld] section in the cluster configuration (config.ini) file, and counts against the maximum number of API connections supported by the cluster.

Suppose that you have 2 cluster host computers, each running an SQL node whose mysqld process was started with --ndb-cluster-connection-pool=4; this means that the cluster must have 8 API slots available for these connections (instead of 2). All of these connections are set up when the SQL node connects to the cluster, and are allocated to threads in a round-robin fashion.

This option is useful only when running mysqld on host machines having multiple CPUs, multiple cores, or both. For best results, the value should be smaller than the total number of cores available on the host machine. Setting it to a value greater than this is likely to degrade performance severely.

> **Important**
>
> Because each SQL node using connection pooling occupies multiple API node slots—each slot having its own node ID in the cluster—you must *not* use a node ID as part of the cluster connection string when starting any `mysqld` process that employs connection pooling.
>
> Setting a node ID in the connection string when using the `--ndb-cluster-connection-pool` option causes node ID allocation errors when the SQL node attempts to connect to the cluster.

- `--ndb-blob-read-batch-bytes=*bytes*`

Table 5.12 Type and value information for ndb-blob-read-batch-bytes

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-blob-read-batch-bytes`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	65536 / 0 - 4294967295
DESCRIPTION: Specifies size in bytes that large BLOB reads should be batched into. 0 = no limit.		

This option can be used to set the size (in bytes) for batching of `BLOB` data reads in MySQL Cluster applications. When this batch size is exceeded by the amount of `BLOB` data to be read within the current transaction, any pending `BLOB` read operations are immediately executed.

The maximum value for this option is 4294967295; the default is 65536. Setting it to 0 has the effect of disabling `BLOB` read batching.

> **Note**
>
> In NDB API applications, you can control `BLOB` write batching with the `setMaxPendingBlobReadBytes()` and `getMaxPendingBlobReadBytes()` methods.

- `--ndb-blob-write-batch-bytes=*bytes*`

Table 5.13 Type and value information for ndb-blob-write-batch-bytes

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-blob-write-batch-bytes`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	65536 / 0 - 4294967295
DESCRIPTION: Specifies size in bytes that large BLOB writes should be batched into. 0 = no limit.		

This option can be used to set the size (in bytes) for batching of BLOB data writes in MySQL Cluster applications. When this batch size is exceeded by the amount of BLOB data to be written within the current transaction, any pending BLOB write operations are immediately executed.

The maximum value for this option is 4294967295; the default is 65536. Setting it to 0 has the effect of disabling BLOB write batching.

> **Note**
>
> In NDB API applications, you can control BLOB write batching with the `setMaxPendingBlobWriteBytes()` and `getMaxPendingBlobWriteBytes()` methods.

- `--ndb-connectstring=*connection_string*`

Table 5.14 Type and value information for ndb-connectstring

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-connectstring`		
Yes	No	No
Yes		No
NDB 7.3	string	
DESCRIPTION: Point to the management server that distributes the cluster configuration		

When using the NDBCLUSTER storage engine, this option specifies the management server that distributes cluster configuration data. See Section 5.3.3, “MySQL Cluster Connection Strings”, for syntax.

- `--ndb-deferred-constraints=[0|1]`

Table 5.15 Type and value information for ndb-deferred-constraints

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-deferred-constraints`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	0 / 0 - 1
DESCRIPTION: Specifies that constraint checks on unique indexes (where these are supported) should be deferred until commit time. Not normally needed or used; for testing purposes only.		

Controls whether or not constraint checks on unique indexes are deferred until commit time, where such checks are supported. `0` is the default.

This option is not normally needed for operation of MySQL Cluster or MySQL Cluster Replication, and is intended primarily for use in testing.

- `--ndb-distribution=[KEYHASH|LINHASH]`

Table 5.16 Type and value information for ndb-distribution

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-distribution`		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	enumeration	KEYHASH / LINHASH, KEYHASH
DESCRIPTION: Default distribution for new tables in NDBCLUSTER (KEYHASH or LINHASH, default is KEYHASH)		

Controls the default distribution method for `NDB` tables. Can be set to either of `KEYHASH` (key hashing) or `LINHASH` (linear hashing). `KEYHASH` is the default.

- `--ndb-mgmd-host=host[:port]`

Table 5.17 Type and value information for ndb-mgmd-host

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-mgmd-host`		
Yes	No	No
Yes		No
NDB 7.3	string	localhost:1186
DESCRIPTION: Set the host (and port, if desired) for connecting to management server		

Can be used to set the host and port number of a single management server for the program to connect to. If the program requires node IDs or references to multiple management servers (or both) in its connection information, use the `--ndb-connectstring` option instead.

- `--ndbcluster`

Table 5.18 Type and value information for ndbcluster

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndbcluster`		
Yes	No	No
Yes		No
NDB 7.3	boolean	FALSE
DESCRIPTION: Enable NDB Cluster (if this version of MySQL supports it)		

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
Disabled by `--skip-ndbcluster`		

The `NDBCLUSTER` storage engine is necessary for using MySQL Cluster. If a `mysqld` binary includes support for the `NDBCLUSTER` storage engine, the engine is disabled by default. Use the `--ndbcluster` option to enable it. Use `--skip-ndbcluster` to explicitly disable the engine.

- `--ndb-log-apply-status`

Table 5.19 Type and value information for ndb-log-apply-status

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-log-apply-status`		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	OFF
DESCRIPTION: Cause a MySQL server acting as a slave to log mysql.ndb_apply_status updates received from its immediate master in its own binary log, using its own server ID. Effective only if the server is started with the --ndbcluster option.		

Causes a slave `mysqld` to log any updates received from its immediate master to the `mysql.ndb_apply_status` table in its own binary log using its own server ID rather than the server ID of the master. In a circular or chain replication setting, this allows such updates to propagate to the `mysql.ndb_apply_status` tables of any MySQL servers configured as slaves of the current `mysqld`.

In a chain replication setup, using this option allows downstream (slave) clusters to be aware of their positions relative to all of their upstream contributors (masters).

In a circular replication setup, this option causes changes to `ndb_apply_status` tables to complete the entire circuit, eventually propagating back to the originating MySQL Cluster. This also allows a cluster acting as a master to see when its changes (epochs) have been applied to the other clusters in the circle.

This option has no effect unless the MySQL server is started with the `--ndbcluster` option.

- `--ndb-log-empty-epochs=[0|1]`

Table 5.20 Type and value information for ndb-log-empty-epochs

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-log-empty-epochs`		

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	boolean	OFF
DESCRIPTION: When enabled, causes epochs in which there were no changes to be written to the ndb_apply_status and ndb_binlog_index tables, even when --log-slave-updates is enabled.		

Causes epochs during which there were no changes to be written to the `ndb_apply_status` and `ndb_binlog_index` tables, even when `--log-slave-updates` is enabled.

By default this option is disabled. Disabling `--ndb-log-empty-epochs` causes epoch transactions with no changes not to be written to the binary log, although a row is still written even for an empty epoch in `ndb_binlog_index`.

Because `--ndb-log-empty-epochs=1` causes the size of `ndb_binlog_index` table to increase independently of the size of the binary log, users should be prepared to manage the growth of this table, even if they expect the cluster to be idle a large part of the time.

- `--ndb-log-exclusive-reads=[0|1]`

Table 5.21 Type and value information for ndb-log-exclusive-reads

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-log-exclusive-reads`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	boolean	0
DESCRIPTION: Log primary key reads with exclusive locks; allow conflict resolution based on read conflicts.		

In MySQL Cluster NDB 7.4.1 and later, starting the server with this option causes primary key reads to be logged with exclusive locks, which allows for MySQL Cluster Replication conflict detection and resolution based on read conflicts. You can also enable and disable these locks at runtime by setting the value of the `ndb_log_exclusive_reads` system variable to 1 or 0, respectively. 0 (disable locking) is the default.

For more information, see Read conflict detection and resolution.

- `--ndb-log-orig`

Table 5.22 Type and value information for ndb-log-orig

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb-log-orig		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	OFF
DESCRIPTION: Log originating server id and epoch in mysql.ndb_binlog_index table.		

Log the originating server ID and epoch in the `ndb_binlog_index` table.

> **Note**
>
> This makes it possible for a given epoch to have multiple rows in `ndb_binlog_index`, one for each originating epoch.

For more information, see Section 8.4, “MySQL Cluster Replication Schema and Tables”.

- `--ndb-log-transaction-id`

Table 5.23 Type and value information for ndb-log-transaction-id

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb-log-transaction-id		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	OFF
DESCRIPTION: Write NDB transaction IDs in the binary log. Requires --log-bin-v1-events=OFF.		

Causes a slave `mysqld` to write the NDB transaction ID in each row of the binary log. Such logging requires the use of the Version 2 event format for the binary log; thus, `--log-bin-use-v1-row-events` must be set to `FALSE` in order to use this option.

This option is not supported in mainline MySQL Server 5.6. It is required to enable MySQL Cluster Replication conflict detection and resolution using the `NDB$EPOCH_TRANS()` function (see NDB $EPOCH_TRANS()).

The default value is `FALSE`.

For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `--ndb-nodeid=#`

Table 5.24 Type and value information for ndb-nodeid

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb-nodeid		
Yes	No	Yes
Yes	Global	No
5.0.45	integer	/ 1 - 63
5.1.5	integer	/ 1 - 255
DESCRIPTION: MySQL Cluster node ID for this MySQL server		

Set this MySQL server's node ID in a MySQL Cluster.

The `--ndb-nodeid` option overrides any node ID set with `--ndb-connectstring`, regardless of the order in which the two options are used.

In addition, if `--ndb-nodeid` is used, then either a matching node ID must be found in a `[mysqld]` or `[api]` section of `config.ini`, or there must be an "open" `[mysqld]` or `[api]` section in the file (that is, a section without a `NodeId` or `Id` parameter specified). This is also true if the node ID is specified as part of the connection string.

Regardless of how the node ID is determined, its is shown as the value of the global status variable `Ndb_cluster_node_id` in the output of `SHOW STATUS`, and as `cluster_node_id` in the `connection` row of the output of `SHOW ENGINE NDBCLUSTER STATUS`.

For more information about node IDs for MySQL Cluster SQL nodes, see Section 5.3.7, "Defining SQL and Other API Nodes in a MySQL Cluster".

- `--ndb_optimization_delay=`*`milliseconds`*

Table 5.25 Type and value information for ndb_optimization_delay

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_optimization_delay		
No	Yes	No
No	Global	Yes
NDB 7.3	integer	10 / 0 - 100000
DESCRIPTION: Sets the number of milliseconds to wait between processing sets of rows by OPTIMIZE TABLE on NDB tables.		

Set the number of milliseconds to wait between sets of rows by `OPTIMIZE TABLE` statements on `NDB` tables. The default is 10.

- `--ndb-recv-thread-activation-threshold=`*`threshold`*

Table 5.26 Type and value information for ndb-recv-thread-activation-threshold

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-recv-thread-activation-threshold`		
Yes	No	No
Yes		No
5.6.10-ndb-7.3.1	integer	8 / 0 (MIN_ACTIVATION_THRESHOLD) - 16 (MAX_ACTIVATION_THRESHOLD)
DESCRIPTION: Activation threshold when receive thread takes over the polling of the cluster connection (measured in concurrently active threads)		

When this number of concurrently active threads is reached, the receive thread takes over polling of the cluster connection.

- `--ndb-recv-thread-cpu-mask=bitmask`

Table 5.27 Type and value information for ndb-recv-thread-cpu-mask

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-recv-thread-cpu-mask`		
Yes	No	No
Yes		No
NDB 7.3	bitmap	[empty]
DESCRIPTION: CPU mask for locking receiver threads to specific CPUs; specified as hexadecimal. See documentation for details.		

Set a CPU mask for locking receiver threads to specific CPUs. This is specified as a hexadecimal bitmask; for example, `0x33` means that one CPU is used per receiver thread. An empty string (no locking of receiver threads) is the default.

- `ndb-transid-mysql-connection-map=state`

Table 5.28 Type and value information for ndb-transid-mysql-connection-map

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-transid-mysql-connection-map`		
Yes	No	No
No		No

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
NDB 7.3	enumeration	ON / ON, OFF, FORCE
DESCRIPTION: Enable or disable the ndb_transid_mysql_connection_map plugin; that is, enable or disable the INFORMATION_SCHEMA table having that name.		

Enables or disables the plugin that handles the `ndb_transid_mysql_connection_map` table in the `INFORMATION_SCHEMA` database. Takes one of the values `ON`, `OFF`, or `FORCE`. `ON` (the default) enables the plugin. `OFF` disables the plugin, which makes `ndb_transid_mysql_connection_map` inaccessible. `FORCE` keeps the MySQL Server from starting if the plugin fails to load and start.

You can see whether the `ndb_transid_mysql_connection_map` table plugin is running by checking the output of `SHOW PLUGINS`.

- `--ndb-wait-connected=`*`seconds`*

Table 5.29 Type and value information for ndb-wait-connected

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-wait-connected`		
Yes	Yes	No
Yes	Global	No
NDB 7.3	integer	0 / 0 - 31536000
5.1.56-ndb-7.1.16, 5.1.56-ndb-7.0.27	integer	30 / 0 - 31536000
DESCRIPTION: Time (in seconds) for the MySQL server to wait for connection to cluster management and data nodes before accepting MySQL client connections.		

This option sets the period of time that the MySQL server waits for connections to MySQL Cluster management and data nodes to be established before accepting MySQL client connections. The time is specified in seconds. The default value is `30`.

- `--ndb-wait-setup=`*`seconds`*

Table 5.30 Type and value information for ndb-wait-setup

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb-wait-setup`		
Yes	Yes	No
Yes	Global	No
NDB 7.3	integer	15 / 0 - 31536000

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
5.1.56-ndb-7.0.27, 5.1.56-ndb-7.1.16	integer	30 / 0 - 31536000
DESCRIPTION: Time (in seconds) for the MySQL server to wait for NDB engine setup to complete.		

This variable shows the period of time that the MySQL server waits for the NDB storage engine to complete setup before timing out and treating NDB as unavailable. The time is specified in seconds. The default value is 30.

- --server-id-bits=#

Table 5.31 Type and value information for server-id-bits

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
server-id-bits		
Yes	Yes	No
Yes	Global	No
NDB 7.3	integer	32 / 7 - 32
DESCRIPTION: Sets the number of least significant bits in the server_id actually used for identifying the server, permitting NDB API applications to store application data in the most significant bits. server_id must be less than 2 to the power of this value.		

This option indicates the number of least significant bits within the 32-bit server_id which actually identify the server. Indicating that the server is actually identified by fewer than 32 bits makes it possible for some of the remaining bits to be used for other purposes, such as storing user data generated by applications using the NDB API's Event API within the AnyValue of an OperationOptions structure (MySQL Cluster uses the AnyValue to store the server ID).

When extracting the effective server ID from server_id for purposes such as detection of replication loops, the server ignores the remaining bits. The --server-id-bits option is used to mask out any irrelevant bits of server_id in the IO and SQL threads when deciding whether an event should be ignored based on the server ID.

This data can be read from the binary log by mysqlbinlog, provided that it is run with its own --server-id-bits option set to 32 (the default).

The value of server_id must be less than 2 ^ server_id_bits; otherwise, mysqld refuses to start.

This system variable is supported only by MySQL Cluster. It is not supported in the standard MySQL 5.6 Server.

- --skip-ndbcluster

Table 5.32 Type and value information for skip-ndbcluster

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
skip-ndbcluster		
Yes	No	No
Yes		No
DESCRIPTION: Disable the NDB Cluster storage engine		

Disable the `NDBCLUSTER` storage engine. This is the default for binaries that were built with `NDBCLUSTER` storage engine support; the server allocates memory and other resources for this storage engine only if the `--ndbcluster` option is given explicitly. See Section 5.1, "Quick Test Setup of MySQL Cluster", for an example.

5.3.8.2 MySQL Cluster System Variables

This section provides detailed information about MySQL server system variables that are specific to MySQL Cluster and the `NDB` storage engine. For system variables not specific to MySQL Cluster, see Server System Variables. For general information on using system variables, see Using System Variables.

- `create_old_temporals`

Table 5.33 Type and value information for create_old_temporals

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
create_old_temporals		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	false
DESCRIPTION: Use pre-5.6.4 storage format for temporal types when creating tables. Intended for use in replication and upgrades/downgrades between NDB 7.2 and NDB 7.3/7.4.		

Causes `mysqld` to use the storage formats for temporal data types that were used in the MySQL server prior to MySQL 5.6.4; that is, `TIME`, `DATETIME`, and `TIMESTAMP` columns are created without support for fractional seconds. This affects all `CREATE TABLE` and `ALTER TABLE` statements.

The `create_old_temporals` system variable is read-only, with a default value of `false`; to enable it, use the `--create-old-temporals` option on the command line or in the server configuration file.

> **Important**
>
> `avoid_temporal_upgrade` must also be enabled for this feature to work properly. It is also strongly recommended that you enable `show_old_temporals` as well. See the descriptions of these variables for more information, as well as Storage Requirements for Date and Time Types.

This variable was added in MySQL Cluster NDB 7.3.10 and MySQL Cluster NDB 7.4.7; it is specific to MySQL Cluster and is not available in standard MySQL Server releases. It is intended to facilitate upgrades from MySQL Cluster NDB 7.2 to MySQL Cluster NDB 7.3 and 7.4; following this, table columns of the affected types can be upgraded to the new storage format. `create_old_temporals` is deprecated and scheduled for removal in a future version of MySQL Cluster.

- `have_ndbcluster`

Table 5.34 Type and value information for have_ndbcluster

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`have_ndbcluster`		
No	Yes	No
No	Global	No
NDB 7.3	boolean	
DESCRIPTION: Whether mysqld supports NDB Cluster tables (set by --ndbcluster option)		

`YES` if `mysqld` supports `NDBCLUSTER` tables. `DISABLED` if `--skip-ndbcluster` is used.

This variable is deprecated and is removed in MySQL 5.6. Use `SHOW ENGINES` instead.

- `ndb_autoincrement_prefetch_sz`

Table 5.35 Type and value information for ndb_autoincrement_prefetch_sz

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_autoincrement_prefetch_sz`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	32 / 1 - 256
5.0.56	integer	1 / 1 - 256
5.1.1	integer	32 / 1 - 256
5.1.23	integer	1 / 1 - 256
5.1.16-ndb-6.2.0	integer	32 / 1 - 256
5.1.23-ndb-6.2.10	integer	1 / 1 - 256
5.1.19-ndb-6.3.0	integer	32 / 1 - 256
5.1.23-ndb-6.3.7	integer	1 / 1 - 256
5.1.41-ndb-6.3.31	integer	1 / 1 - 65536
5.1.30-ndb-6.4.0	integer	32 / 1 - 256
5.1.41-ndb-7.0.11	integer	1 / 1 - 65536
5.5.15-ndb-7.2.1	integer	1 / 1 - 65536

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
DESCRIPTION: NDB auto-increment prefetch size		

Determines the probability of gaps in an autoincremented column. Set it to `1` to minimize this. Setting it to a high value for optimization makes inserts faster, but decreases the likelihood that consecutive autoincrement numbers will be used in a batch of inserts. The mininum and default value is 1. The maximum value for `ndb_autoincrement_prefetch_sz` is 65536.

This variable affects only the number of `AUTO_INCREMENT` IDs that are fetched between statements; within a given statement, at least 32 IDs are obtained at a time. The default value is 1.

> **Important**
>
> This variable does not affect inserts performed using `INSERT ... SELECT`.

- `ndb_cache_check_time`

Table 5.36 Type and value information for ndb_cache_check_time

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_cache_check_time`		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	integer	0 / -
DESCRIPTION: Number of milliseconds between checks of cluster SQL nodes made by the MySQL query cache		

The number of milliseconds that elapse between checks of MySQL Cluster SQL nodes by the MySQL query cache. Setting this to 0 (the default and minimum value) means that the query cache checks for validation on every query.

The recommended maximum value for this variable is 1000, which means that the check is performed once per second. A larger value means that the check is performed and possibly invalidated due to updates on different SQL nodes less often. It is generally not desirable to set this to a value greater than 2000.

- `ndb_clear_apply_status`

Table 5.37 Type and value information for ndb_clear_apply_status

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_clear_apply_status`		
Yes	Yes	No

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
No	Global	Yes
NDB 7.3	boolean	ON
DESCRIPTION: Causes RESET SLAVE to clear all rows from the ndb_apply_status table. ON by default.		

By the default, executing `RESET SLAVE` causes a MySQL Cluster replication slave to purge all rows from its `ndb_apply_status` table. In MySQL Cluster NDB 7.4.9 and later you can disable this by setting `ndb_clear_apply_status=OFF`.

- `ndb_deferred_constraints`

Table 5.38 Type and value information for ndb_deferred_constraints

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_deferred_constraints`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	0 / 0 - 1
DESCRIPTION: Specifies that constraint checks should be deferred (where these are supported). Not normally needed or used; for testing purposes only.		

Controls whether or not constraint checks are deferred, where these are supported. `0` is the default.

This variable is not normally needed for operation of MySQL Cluster or MySQL Cluster Replication, and is intended primarily for use in testing.

- `ndb_distribution`

Table 5.39 Type and value information for ndb_distribution

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_distribution`		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	enumeration	KEYHASH / LINHASH, KEYHASH
DESCRIPTION: Default distribution for new tables in NDBCLUSTER (KEYHASH or LINHASH, default is KEYHASH)		

Controls the default distribution method for NDB tables. Can be set to either of KEYHASH (key hashing) or LINHASH (linear hashing). KEYHASH is the default.

- ndb_eventbuffer_free_percent

Table 5.40 Type and value information for ndb_eventbuffer_free_percent

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_eventbuffer_free_percent		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	integer	20 / 1 - 99
DESCRIPTION: Percentage of free memory that should be available in event buffer before resumption of buffering, after reaching limit set by ndb_eventbuffer_max_alloc.		

Sets the percentage of the maximum memory allocated to the event buffer (ndb_eventbuffer_max_alloc) that should be available in event buffer after reaching the maximum, before starting to buffer again.

ndb_eventbuffer_free_percent was added in MySQL Cluster NDB 7.4.3.

- ndb_eventbuffer_max_alloc

Table 5.41 Type and value information for ndb_eventbuffer_max_alloc

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_eventbuffer_max_alloc		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	integer	0 / 0 - 4294967295
DESCRIPTION: Maximum memory that can be allocated for buffering events by the NDB API. Defaults to 0 (no limit).		

Sets the maximum amount memory (in bytes) that can be allocated for buffering events by the NDB API. 0 means that no limit is imposed, and is the default.

This variable was added in MySQL Cluster NDB 7.3.3.

- ndb_extra_logging

Table 5.42 Type and value information for ndb_extra_logging

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_extra_logging		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	integer	0 / -
5.1.19-ndb-6.3.0	integer	1 / -
DESCRIPTION: Controls logging of MySQL Cluster schema, connection, and data distribution events in the MySQL error log		

This variable enables recording in the MySQL error log of information specific to the NDB storage engine.

When this variable is set to 0, the only information specific to NDB that is written to the MySQL error log relates to transaction handling. If it set to a value greater than 0 but less than 10, NDB table schema and connection events are also logged, as well as whether or not conflict resolution is in use, and other NDB errors and information. If the value is set to 10 or more, information about NDB internals, such as the progress of data distribution among cluster nodes, is also written to the MySQL error log. The default is 1.

- `ndb_force_send`

Table 5.43 Type and value information for ndb_force_send

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_force_send		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	boolean	TRUE
DESCRIPTION: Forces sending of buffers to NDB immediately, without waiting for other threads		

Forces sending of buffers to NDB immediately, without waiting for other threads. Defaults to ON.

- `ndb_index_stat_cache_entries`

Table 5.44 Type and value information for ndb_index_stat_cache_entries

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_index_stat_cache_entries		

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	32 / 0 - 4294967295
DESCRIPTION: Sets the granularity of the statistics by determining the number of starting and ending keys		

Sets the granularity of the statistics by determining the number of starting and ending keys to store in the statistics memory cache. Zero means no caching takes place; in this case, the data nodes are always queried directly. Default value: `32`.

> **Note**
>
> If `ndb_index_stat_enable` is `OFF`, then setting this variable has no effect.

This variable was deprecated in MySQL 5.1, and is removed from MySQL Cluster NDB 7.3.5 and later.

- `ndb_index_stat_enable`

Table 5.45 Type and value information for ndb_index_stat_enable

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_index_stat_enable`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	boolean	OFF
5.5.15-ndb-7.2.1	boolean	ON
DESCRIPTION: Use NDB index statistics in query optimization		

Use `NDB` index statistics in query optimization. The default is `ON`.

- `ndb_index_stat_option`

Table 5.46 Type and value information for ndb_index_stat_option

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_index_stat_option`		
Yes	Yes	No
Yes	Both	Yes

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
NDB 7.3	string	loop_enable=1000ms,loop_idle=1000ms,l update_batch=1,read_batch=4,idle_batch= check_delay=10m,delete_batch=8, clean_delay=1m,error_batch=4, error_delay=1m,evict_batch=8,evict_delay cache_lowpct=90,zero_total=0
5.1.56-ndb-7.1.17	string	loop_checkon=1000ms,loop_idle=1000ms update_batch=1,read_batch=4,idle_batch= check_delay=1m,delete_batch=8,clean_de error_delay=1m,evict_batch=8,evict_delay cache_lowpct=90
DESCRIPTION: Comma-separated list of tunable options for NDB index statistics; the list should contain no spaces		

This variable is used for providing tuning options for NDB index statistics generation. The list consist of comma-separated name-value pairs of option names and values, and this list must not contain any space characters.

Options not used when setting `ndb_index_stat_option` are not changed from their default values. For example, you can set `ndb_index_stat_option = 'loop_idle=1000ms,cache_limit=32M'`.

Time values can be optionally suffixed with `h` (hours), `m` (minutes), or `s` (seconds). Millisecond values can optionally be specified using `ms`; millisecond values cannot be specified using `h`, `m`, or `s`.) Integer values can be suffixed with `K`, `M`, or `G`.

The names of the options that can be set using this variable are shown in the table that follows. The table also provides brief descriptions of the options, their default values, and (where applicable) their minimum and maximum values.

Name	Description	Default/Units	Minimum/Maximum
`loop_enable`		1000 ms	0/4G
`loop_idle`	Time to sleep when idle	1000 ms	0/4G
`loop_busy`	Time to sleep when more work is waiting	100 ms	0/4G
`update_batch`		1	0/4G
`read_batch`		4	1/4G
`idle_batch`		32	1/4G
`check_batch`		8	1/4G
`check_delay`	How often to check for new statistics	10 m	1/4G
`delete_batch`		8	0/4G
`clean_delay`		1 m	0/4G
`error_batch`		4	1/4G
`error_delay`		1 m	1/4G
`evict_batch`		8	1/4G

Name	Description	Default/Units	Minimum/Maximum
evict_delay	Clean LRU cache, from read time	1 m	0/4G
cache_limit	Maximum amount of memory in bytes used for cached index statistics by this mysqld; clean up the cache when this is exceeded.	32 M	0/4G
cache_lowpct		90	0/100
zero_total	Setting this to 1 resets all accumulating counters in ndb_index_stat_status to 0. This option value is also reset to 0 when this is done.	0	0/1

- ndb_index_stat_update_freq

Table 5.47 Type and value information for ndb_index_stat_update_freq

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_index_stat_update_freq		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	integer	20 / 0 - 4294967295
DESCRIPTION: How often to query data nodes instead of the statistics cache		

How often to query data nodes instead of the statistics cache. For example, a value of 20 (the default) means to direct every 20th query to the data nodes.

> **Note**
>
> If ndb_index_stat_cache_entries is 0, then setting this variable has no effect; in this case, every query is sent directly to the data nodes.

This variable was deprecated in MySQL 5.1, and is removed from MySQL Cluster NDB 7.3.5 and later.

- ndb_join_pushdown

Table 5.48 Type and value information for ndb_join_pushdown

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_join_pushdown		
No	Yes	No
No	Both	Yes
5.1.51-ndb-7.2.0	boolean	TRUE
DESCRIPTION: Enables pushing down of joins to data nodes		

This variable controls whether joins on `NDB` tables are pushed down to the NDB kernel (data nodes). Previously, a join was handled using multiple accesses of `NDB` by the SQL node; however, when `ndb_join_pushdown` is enabled, a pushable join is sent in its entirety to the data nodes, where it can be distributed among the data nodes and executed in parallel on multiple copies of the data, with a single, merged result being returned to `mysqld`. This can reduce greatly the number of round trips between an SQL node and the data nodes required to handle such a join.

By default, `ndb_join_pushdown` is enabled.

Conditions for NDB pushdown joins. In order for a join to be pushable, it must meet the following conditions:

1. Only columns can be compared, and all columns to be joined must use *exactly* the same data type.

   This means that expressions such as `t1.a = t2.a + constant` cannot be pushed down, and that (for example) a join on an `INT` column and a `BIGINT` column also cannot be pushed down.

2. Queries referencing `BLOB` or `TEXT` columns are not supported.

3. Explicit locking is not supported; however, the `NDB` storage engine's characteristic implicit row-based locking is enforced.

   This means that a join using `FOR UPDATE` cannot be pushed down.

4. In order for a join to be pushed down, child tables in the join must be accessed using one of the `ref`, `eq_ref`, or `const` access methods, or some combination of these methods.

   Outer joined child tables can only be pushed using `eq_ref`.

   If the root of the pushed join is an `eq_ref` or `const`, only child tables joined by `eq_ref` can be appended. (A table joined by `ref` is likely to become the root of another pushed join.)

   If the query optimizer decides on `Using join cache` for a candidate child table, that table cannot be pushed as a child. However, it may be the root of another set of pushed tables.

5. Joins referencing tables explicitly partitioned by `[LINEAR] HASH`, `LIST`, or `RANGE` currently cannot be pushed down.

You can see whether a given join can be pushed down by checking it with `EXPLAIN`; when the join can be pushed down, you can see references to the `pushed join` in the `Extra` column of the output, as shown in this example:

```
mysql> EXPLAIN
```

```
    ->     SELECT e.first_name, e.last_name, t.title, d.dept_name
    ->         FROM employees e
    ->         JOIN dept_emp de ON e.emp_no=de.emp_no
    ->         JOIN departments d ON d.dept_no=de.dept_no
    ->         JOIN titles t ON e.emp_no=t.emp_no\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: d
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 9
        Extra: Parent of 4 pushed join@1
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: de
         type: ref
possible_keys: PRIMARY,emp_no,dept_no
          key: dept_no
      key_len: 4
          ref: employees.d.dept_no
         rows: 5305
        Extra: Child of 'd' in pushed join@1
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: e
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: employees.de.emp_no
         rows: 1
        Extra: Child of 'de' in pushed join@1
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
         type: ref
possible_keys: PRIMARY,emp_no
          key: emp_no
      key_len: 4
          ref: employees.de.emp_no
         rows: 19
        Extra: Child of 'e' in pushed join@1
4 rows in set (0.00 sec)
```

> **Note**
>
> If inner joined child tables are joined by `ref`, *and* the result is ordered or grouped by a sorted index, this index cannot provide sorted rows, which forces writing to a sorted tempfile.

Two additional sources of information about pushed join performance are available:

1. The status variables `Ndb_pushed_queries_defined`, `Ndb_pushed_queries_dropped`, `Ndb_pushed_queries_executed`, and `Ndb_pushed_reads`.

2. The counters in the `ndbinfo.counters` table that belong to the `DBSPJ` kernel block. See Section 7.10.7, "The ndbinfo counters Table", for information about these counters. See also The DBSPJ Block, in the *MySQL Cluster API Developer Guide*.

- `ndb_log_apply_status`

Table 5.49 Type and value information for ndb_log_apply_status

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_log_apply_status		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	OFF
DESCRIPTION: Whether or not a MySQL server acting as a slave logs mysql.ndb_apply_status updates received from its immediate master in its own binary log, using its own server ID.		

A read-only variable which shows whether the server was started with the `--ndb-log-apply-status` option.

- `ndb_log_bin`

Table 5.50 Type and value information for ndb_log_bin

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_log_bin		
Yes	Yes	No
No	Both	Yes
NDB 7.3	boolean	ON
DESCRIPTION: Write updates to NDB tables in the binary log. Effective only if binary logging is enabled with --log-bin.		

Causes updates to `NDB` tables to be written to the binary log. Setting this variable has no effect if binary logging is not already enabled for the server using `log_bin`. `ndb_log_bin` defaults to 1 (ON); normally, there is never any need to change this value in a production environment.

- `ndb_log_binlog_index`

Table 5.51 Type and value information for ndb_log_binlog_index

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_log_binlog_index		
Yes	Yes	No
No	Global	Yes
NDB 7.3	boolean	ON

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
DESCRIPTION: Insert mapping between epochs and binary log positions into the ndb_binlog_index table. Defaults to ON. Effective only if binary logging is enabled on the server.		

Causes a mapping of epochs to positions in the binary log to be inserted into the `ndb_binlog_index` table. Setting this variable has no effect if binary logging is not already enabled for the server using `log_bin`. (In addition, `ndb_log_bin` must not be disabled.) `ndb_log_binlog_index` defaults to `1` (`ON`); normally, there is never any need to change this value in a production environment.

- `ndb_log_empty_epochs`

Table 5.52 Type and value information for ndb_log_empty_epochs

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_log_empty_epochs`		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	boolean	OFF
DESCRIPTION: When enabled, epochs in which there were no changes are written to the ndb_apply_status and ndb_binlog_index tables, even when log_slave_updates is enabled.		

When this variable is set to 0, epoch transactions with no changes are not written to the binary log, although a row is still written even for an empty epoch in `ndb_binlog_index`.

- `ndb_log_exclusive_reads`

Table 5.53 Type and value information for ndb_log_exclusive_reads

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_log_exclusive_reads`		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	boolean	0
DESCRIPTION: Log primary key reads with exclusive locks; allow conflict resolution based on read conflicts.		

In MySQL Cluster NDB 7.4.1 and later, this variable determines whether primary key reads are logged with exclusive locks, which allows for MySQL Cluster Replication conflict detection and resolution based on read conflicts. To enable these locks, set the value of `ndb_log_exclusive_reads` to 1. 0, which disables such locking, is the default.

For more information, see Read conflict detection and resolution.

- ndb_log_orig

Table 5.54 Type and value information for ndb_log_orig

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_log_orig		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	OFF
DESCRIPTION: Whether the id and epoch of the originating server are recorded in the mysql.ndb_binlog_index table. Set using the --ndb-log-orig option when starting mysqld.		

Shows whether the originating server ID and epoch are logged in the `ndb_binlog_index` table. Set using the `--ndb-log-orig` server option.

- ndb_log_transaction_id

Table 5.55 Type and value information for ndb_log_transaction_id

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_log_transaction_id		
No	Yes	No
No	Global	No
NDB 7.3	boolean	OFF
DESCRIPTION: Whether NDB transaction IDs are written into the binary log. (Read-only.)		

This read-only, Boolean system variable shows whether a slave `mysqld` writes NDB transaction IDs in the binary log (required to use "active-active" MySQL Cluster Replication with `NDB$EPOCH_TRANS()` conflict detection). To change the setting, use the `--ndb-log-transaction-id` option.

`ndb_log_transaction_id` is not supported in mainline MySQL Server 5.6.

For more information, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- ndb_optimized_node_selection

Table 5.56 Type and value information for ndb_optimized_node_selection

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_optimized_node_selection		
Yes	Yes	No
Yes	Global	No
NDB 7.3	boolean	ON
5.1.22-ndb-6.3.4	integer	3 / 0 - 3
DESCRIPTION: Determines how an SQL node chooses a cluster data node to use as transaction coordinator		

There are two forms of optimized node selection, described here:

1. The SQL node uses *promixity* to determine the transaction coordinator; that is, the “closest” data node to the SQL node is chosen as the transaction coordinator. For this purpose, a data node having a shared memory connection with the SQL node is considered to be “closest” to the SQL node; the next closest (in order of decreasing proximity) are: TCP connection to `localhost`; SCI connection; TCP connection from a host other than `localhost`.

2. The SQL thread uses *distribution awareness* to select the data node. That is, the data node housing the cluster partition accessed by the first statement of a given transaction is used as the transaction coordinator for the entire transaction. (This is effective only if the first statement of the transaction accesses no more than one cluster partition.)

This option takes one of the integer values `0`, `1`, `2`, or `3`. `3` is the default. These values affect node selection as follows:

- `0`: Node selection is not optimized. Each data node is employed as the transaction coordinator 8 times before the SQL thread proceeds to the next data node.

- `1`: Proximity to the SQL node is used to determine the transaction coordinator.

- `2`: Distribution awareness is used to select the transaction coordinator. However, if the first statement of the transaction accesses more than one cluster partition, the SQL node reverts to the round-robin behavior seen when this option is set to `0`.

- `3`: If distribution awareness can be employed to determine the transaction coordinator, then it is used; otherwise proximity is used to select the transaction coordinator. (This is the default behavior.)

- `ndb_recv_thread_activation_threshold`

Table 5.57 Type and value information for ndb_recv_thread_activation_threshold

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_recv_thread_activation_threshold		
No	No	No

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
No		No
5.6.10-ndb-7.3.1	integer	8 / 0 (MIN_ACTIVATION_THRESHOLD) - 16 (MAX_ACTIVATION_THRESHOLD)
DESCRIPTION: Activation threshold when receive thread takes over the polling of the cluster connection (measured in concurrently active threads)		

When this number of concurrently active threads is reached, the receive thread takes over polling of the cluster connection.

This variable is global in scope. It can also be set on startup using the `--ndb-recv-thread-activation-threshold` option.

- `ndb_recv_thread_cpu_mask`

Table 5.58 Type and value information for ndb_recv_thread_cpu_mask

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_recv_thread_cpu_mask`		
No	Yes	No
No	Global	Yes
NDB 7.3	bitmap	[empty]
DESCRIPTION: CPU mask for locking receiver threads to specific CPUs; specified as hexadecimal. See documentation for details.		

CPU mask for locking receiver threads to specific CPUs. This is specified as a hexadecimal bitmask; for example, `0x33` means that one CPU is used per receiver thread. An empty string is the default; setting `ndb_recv_thread_cpu_mask` to this value removes any receiver thread locks previously set.

This variable is global in scope. It can also be set on startup using the `--ndb-recv-thread-cpu-mask` option.

- `ndb_report_thresh_binlog_epoch_slip`

Table 5.59 Type and value information for ndb_report_thresh_binlog_epoch_slip

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_report_thresh_binlog_epoch_slip`		
Yes	No	No

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
Yes		No
NDB 7.3	integer	3 / 0 - 256
DESCRIPTION: This is a threshold on the number of epochs to be behind before reporting binary log status		

This is a threshold on the number of epochs to be behind before reporting binary log status. For example, a value of `3` (the default) means that if the difference between which epoch has been received from the storage nodes and which epoch has been applied to the binary log is 3 or more, a status message will be sent to the cluster log.

- `ndb_report_thresh_binlog_mem_usage`

Table 5.60 Type and value information for ndb_report_thresh_binlog_mem_usage

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndb_report_thresh_binlog_mem_usage`		
Yes	No	No
Yes		No
NDB 7.3	integer	10 / 0 - 10
DESCRIPTION: This is a threshold on the percentage of free memory remaining before reporting binary log status		

This is a threshold on the percentage of free memory remaining before reporting binary log status. For example, a value of `10` (the default) means that if the amount of available memory for receiving binary log data from the data nodes falls below 10%, a status message will be sent to the cluster log.

- `slave_allow_batching`

Table 5.61 Type and value information for slave_allow_batching

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`slave_allow_batching`		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	boolean	off
DESCRIPTION: Turns update batching on and off for a replication slave		

Whether or not batched updates are enabled on MySQL Cluster replication slaves.

This variable is available for mysqld only as supplied with MySQL Cluster or built from the MySQL Cluster sources. For more information, see Section 8.6, "Starting MySQL Cluster Replication (Single Replication Channel)".

- ndb_show_foreign_key_mock_tables

Table 5.62 Type and value information for ndb_show_foreign_key_mock_tables

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_show_foreign_key_mock_tables		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	boolean	OFF
DESCRIPTION: Show the mock tables used to support foreign_key_checks=0.		

Show the mock tables used by NDB to support foreign_key_checks=0. When this is enabled, extra warnings are shown when creating and dropping the tables. The real (internal) name of the table can be seen in the output of SHOW CREATE TABLE.

- ndb_slave_conflict_role

Table 5.63 Type and value information for ndb_slave_conflict_role

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_slave_conflict_role		
Yes	Yes	No
Yes	Global	Yes
NDB 7.3	enumeration	NONE / NONE, PRIMARY, SECONDARY, PASS
DESCRIPTION: Role for slave to play in conflict detection and resolution. Value is one of PRIMARY, SECONDARY, PASS, or NONE (default). Can be changed only when slave SQL thread is stopped. See documentation for further information.		

Determine the role of this SQL node (and MySQL Cluster) in a circular ("active-active") replication setup. ndb_slave_conflict_role can take any one of the values PRIMARY, SECONDARY, PASS, or NULL (the default). The slave SQL thread must be stopped before you can change ndb_slave_conflict_role. In addition, it is not possible to change directly between PASS and either of PRIMARY or SECONDARY directly; in such cases, you must ensure that the SQL thread is stopped, then execute SET @@GLOBAL.ndb_slave_conflict_role = 'NONE' first.

This variable was added in MySQL Cluster NDB 7.4.1. For more information, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- ndb_table_no_logging

Table 5.64 Type and value information for ndb_table_no_logging

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_table_no_logging		
No	Yes	No
No	Session	Yes
NDB 7.3	boolean	FALSE
DESCRIPTION: NDB tables created when this setting is enabled are not checkpointed to disk (although table schema files are created). The setting in effect when the table is created with or altered to use NDBCLUSTER persists for the lifetime of the table.		

When this variable is set to `ON` or `1`, it causes `NDB` tables not to be checkpointed to disk. More specifically, this setting applies to tables which are created or altered using `ENGINE NDB` when `ndb_table_no_logging` is enabled, and continues to apply for the lifetime of the table, even if `ndb_table_no_logging` is later changed. Suppose that `A`, `B`, `C`, and `D` are tables that we create (and perhaps also alter), and that we also change the setting for `ndb_table_no_logging` as shown here:

```
SET @@ndb_table_no_logging = 1;
CREATE TABLE A ... ENGINE NDB;
CREATE TABLE B ... ENGINE MYISAM;
CREATE TABLE C ... ENGINE MYISAM;
ALTER TABLE B ENGINE NDB;
SET @@ndb_table_no_logging = 0;
CREATE TABLE D ... ENGINE NDB;
ALTER TABLE C ENGINE NDB;
SET @@ndb_table_no_logging = 1;
```

After the previous sequence of events, tables `A` and `B` are not checkpointed; `A` was created with `ENGINE NDB` and B was altered to use `NDB`, both while `ndb_table_no_logging` was enabled. However, tables `C` and `D` are logged; `C` was altered to use `NDB` and `D` was created using `ENGINE NDB`, both while `ndb_table_no_logging` was disabled. Setting `ndb_table_no_logging` back to `1` or `ON` does *not* cause table `C` or `D` to be checkpointed.

> **Note**
>
> `ndb_table_no_logging` has no effect on the creation of `NDB` table schema files; to suppress these, use `ndb_table_temporary` instead.

- `ndb_table_temporary`

Table 5.65 Type and value information for ndb_table_temporary

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_table_temporary		
No	Yes	No
No	Session	Yes

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
NDB 7.3	boolean	FALSE
DESCRIPTION: NDB tables are not persistent on disk: no schema files are created and the tables are not logged		

When set to ON or 1, this variable causes NDB tables not to be written to disk: This means that no table schema files are created, and that the tables are not logged.

> **Note**
>
> Setting this variable currently has no effect in MySQL Cluster NDB 7.0 and later. This is a known issue; see Bug #34036.

- ndb_use_copying_alter_table

Table 5.66 Type and value information for ndb_use_copying_alter_table

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_use_copying_alter_table		
No	Yes	No
No	Both	No
DESCRIPTION: Use copying ALTER TABLE operations in MySQL Cluster		

Forces NDB to use copying of tables in the event of problems with online ALTER TABLE operations. The default value is OFF.

- ndb_use_exact_count

Table 5.67 Type and value information for ndb_use_exact_count

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_use_exact_count		
No	Yes	No
No	Both	Yes
NDB 7.3	boolean	ON
5.1.47-ndb-7.1.8	boolean	OFF
DESCRIPTION: Use exact row count when planning queries		

Forces NDB to use a count of records during SELECT COUNT(*) query planning to speed up this type of query. The default value is OFF, which allows for faster queries overall.

- ndb_use_transactions

Table 5.68 Type and value information for ndb_use_transactions

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_use_transactions		
Yes	Yes	No
Yes	Both	Yes
NDB 7.3	boolean	ON
DESCRIPTION: Forces NDB to use a count of records during SELECT COUNT(*) query planning to speed up this type of query		

You can disable `NDB` transaction support by setting this variable's values to `OFF` (not recommended). The default is `ON`.

- `ndb_version`

Table 5.69 Type and value information for ndb_version

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_version		
No	Yes	No
No	Global	No
NDB 7.3	string	
DESCRIPTION: Shows build and NDB engine version as an integer.		

`NDB` engine version, as a composite integer.

- `ndb_version_string`

Table 5.70 Type and value information for ndb_version_string

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndb_version_string		
No	Yes	No
No	Global	No
NDB 7.3	string	
DESCRIPTION: Shows build information including NDB engine version in ndb-x.y.z format.		

`NDB` engine version in `ndb-x.y.z` format.

- `server_id_bits`

Table 5.71 Type and value information for server_id_bits

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
server_id_bits		
Yes	Yes	No
Yes	Global	No
NDB 7.3	integer	32 / 7 - 32
DESCRIPTION: The effective value of server_id if the server was started with the --server-id-bits option set to a nondefault value.		

The effective value of `server_id` if the server was started with the `--server-id-bits` option set to a nondefault value.

If the value of `server_id` greater than or equal to 2 to the power of `server_id_bits`, `mysqld` refuses to start.

This system variable is supported only by MySQL Cluster. `server_id_bits` is not supported by the standard MySQL Server.

- `transaction_allow_batching`

Table 5.72 Type and value information for transaction_allow_batching

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
transaction_allow_batching		
No	Yes	No
No	Session	Yes
NDB 7.3	boolean	FALSE
DESCRIPTION: Allows batching of statements within a transaction. Disable AUTOCOMMIT to use.		

When set to `1` or `ON`, this variable enables batching of statements within the same transaction. To use this variable, `autocommit` must first be disabled by setting it to `0` or `OFF`; otherwise, setting `transaction_allow_batching` has no effect.

It is safe to use this variable with transactions that performs writes only, as having it enabled can lead to reads from the "before" image. You should ensure that any pending transactions are committed (using an explicit `COMMIT` if desired) before issuing a `SELECT`.

> **Important**
>
> `transaction_allow_batching` should not be used whenever there is the possibility that the effects of a given statement depend on the outcome of a previous statement within the same transaction.

This variable is currently supported for MySQL Cluster only.

The system variables in the following list all relate to the ndbinfo information database.

- ndbinfo_database

Table 5.73 Type and value information for ndbinfo_database

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndbinfo_database		
No	Yes	No
No	Global	No
NDB 7.3	string	ndbinfo
DESCRIPTION: The name used for the NDB information database; read only.		

Shows the name used for the NDB information database; the default is ndbinfo. This is a read-only variable whose value is determined at compile time; you can set it by starting the server using --ndbinfo-database=*name*, which sets the value shown for this variable but does not actually change the name used for the NDB information database.

- ndbinfo_max_bytes

Table 5.74 Type and value information for ndbinfo_max_bytes

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndbinfo_max_bytes		
Yes	Yes	No
No	Both	Yes
NDB 7.3	integer	0 / -
DESCRIPTION: Used for debugging only.		

Used in testing and debugging only.

- ndbinfo_max_rows

Table 5.75 Type and value information for ndbinfo_max_rows

Command Line	System Variable	Status Variable
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
ndbinfo_max_rows		
Yes	Yes	No
No	Both	Yes
NDB 7.3	integer	10 / -
DESCRIPTION: Used for debugging only.		

Used in testing and debugging only.

- `ndbinfo_offline`

Table 5.76 Type and value information for ndbinfo_offline

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndbinfo_offline`		
No	Yes	No
No	Global	Yes
NDB 7.3	boolean	OFF
DESCRIPTION: Put the ndbinfo database into offline mode, in which no rows are returned from tables or views.		

Place the `ndbinfo` database into offline mode, in which tables and views can be opened even when they do not actually exist, or when they exist but have different definitions in `NDB`. No rows are returned from such tables (or views).

- `ndbinfo_show_hidden`

Table 5.77 Type and value information for ndbinfo_show_hidden

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndbinfo_show_hidden`		
Yes	Yes	No
No	Both	Yes
NDB 7.3	boolean	OFF
DESCRIPTION: Whether to show ndbinfo internal base tables in the mysql client. The default is OFF.		

Whether or not the `ndbinfo` database's underlying internal tables are shown in the `mysql` client. The default is `OFF`.

- `ndbinfo_table_prefix`

Table 5.78 Type and value information for ndbinfo_table_prefix

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndbinfo_table_prefix`		
Yes	Yes	No
No	Both	Yes

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
NDB 7.3	string	ndb$
DESCRIPTION: The prefix to use for naming ndbinfo internal base tables		

The prefix used in naming the ndbinfo database's base tables (normally hidden, unless exposed by setting `ndbinfo_show_hidden`). This is a read-only variable whose default value is "`ndb$`". You can start the server with the `--ndbinfo-table-prefix` option, but this merely sets the variable and does not change the actual prefix used to name the hidden base tables; the prefix itself is determined at compile time.

- `ndbinfo_version`

Table 5.79 Type and value information for ndbinfo_version

Command Line	**System Variable**	**Status Variable**
Option File	**Scope**	**Dynamic**
From Version	**Type**	**Default, Range**
Notes		
`ndbinfo_version`		
No	Yes	No
No	Global	No
NDB 7.3	string	
DESCRIPTION: The version of the ndbinfo engine; read only.		

Shows the version of the `ndbinfo` engine in use; read-only.

5.3.8.3 MySQL Cluster Status Variables

This section provides detailed information about MySQL server status variables that relate to MySQL Cluster and the `NDB` storage engine. For status variables not specific to MySQL Cluster, and for general information on using status variables, see Server Status Variables.

- `Handler_discover`

  The MySQL server can ask the `NDBCLUSTER` storage engine if it knows about a table with a given name. This is called discovery. `Handler_discover` indicates the number of times that tables have been discovered using this mechanism.

- `Ndb_api_bytes_sent_count_session`

  Amount of data (in bytes) sent to the data nodes in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_bytes_sent_count_slave`

  Amount of data (in bytes) sent to the data nodes by this slave.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_sent_count`

  Amount of data (in bytes) sent to the data nodes by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count_session`

  Amount of data (in bytes) received from the data nodes in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count_slave`

  Amount of data (in bytes) received from the data nodes by this slave.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count`

  Amount of data (in bytes) received from the data nodes by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_data_count_injector`

  The number of row change events received by the NDB binlog injector thread.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_data_count`

  The number of row change events received by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_nondata_count_injector`

The number of events received, other than row change events, by the NDB binary log injector thread.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_nondata_count`

  The number of events received, other than row change events, by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_bytes_count_injector`

  The number of bytes of events received by the NDB binlog injector thread.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_bytes_count`

  The number of bytes of events received by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count_session`

  The number of operations in this client session based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count_slave`

  The number of operations by this slave based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count`

  The number of operations by this MySQL Server (SQL node) based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it is effectively global in scope.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_pruned_scan_count_session

  The number of scans in this client session that have been pruned to a single partition.

  Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it relates to the current session only, and is not affected by any other clients of this mysqld.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_pruned_scan_count_slave

  The number of scans by this slave that have been pruned to a single partition.

  Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_pruned_scan_count

  The number of scans by this MySQL Server (SQL node) that have been pruned to a single partition.

  Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_range_scan_count_session

  The number of range scans that have been started in this client session.

  Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it relates to the current session only, and is not affected by any other clients of this mysqld.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_range_scan_count_slave

  The number of range scans that have been started by this slave.

  Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_range_scan_count

  The number of range scans that have been started by this MySQL Server (SQL node).

  Although this variable can be read using either SHOW GLOBAL STATUS or SHOW SESSION STATUS, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_read_row_count_session

The total number of rows that have been read in this client session. This includes all rows read by any primary key, unique key, or scan operation made in this client session.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_read_row_count_slave`

  The total number of rows that have been read by this slave. This includes all rows read by any primary key, unique key, or scan operation made by this slave.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_read_row_count`

  The total number of rows that have been read by this MySQL Server (SQL node). This includes all rows read by any primary key, unique key, or scan operation made by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count_session`

  The number of batches of rows received in this client session. 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count_slave`

  The number of batches of rows received by this slave. 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count`

  The number of batches of rows received by this MySQL Server (SQL node). 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count_session`

The number of table scans that have been started in this client session, including scans of internal tables,.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count_slave`

  The number of table scans that have been started by this slave, including scans of internal tables,.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count`

  The number of table scans that have been started by this MySQL Server (SQL node), including scans of internal tables,.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count_session`

  The number of transactions aborted in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count_slave`

  The number of transactions aborted by this slave.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count`

  The number of transactions aborted by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count_session`

  The number of transactions closed in this client session. This value may be greater than the sum of `Ndb_api_trans_commit_count_session` and `Ndb_api_trans_abort_count_session`, since some transactions may have been rolled back.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count_slave`

  The number of transactions closed by this slave. This value may be greater than the sum of `Ndb_api_trans_commit_count_slave` and `Ndb_api_trans_abort_count_slave`, since some transactions may have been rolled back.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count`

  The number of transactions closed by this MySQL Server (SQL node). This value may be greater than the sum of `Ndb_api_trans_commit_count` and `Ndb_api_trans_abort_count`, since some transactions may have been rolled back.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count_session`

  The number of transactions committed in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count_slave`

  The number of transactions committed by this slave.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count`

  The number of transactions committed by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count_session`

  The total number of rows that have been read in this client session. This includes all rows read by any primary key, unique key, or scan operation made in this client session.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count_slave`

  The total number of rows that have been read by this slave. This includes all rows read by any primary key, unique key, or scan operation made by this slave.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count`

  The total number of rows that have been read by this MySQL Server (SQL node). This includes all rows read by any primary key, unique key, or scan operation made by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count_session`

  The number of transactions started in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count_slave`

  The number of transactions started by this slave.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count`

  The number of transactions started by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_uk_op_count_session`

  The number of operations in this client session based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_uk_op_count_slave`

  The number of operations by this slave based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_uk_op_count`

  The number of operations by this MySQL Server (SQL node) based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count_session`

  The number of times a thread has been blocked in this client session while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit implicit executes for blob and auto-increment operations not visible to clients.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count_slave`

  The number of times a thread has been blocked by this slave while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit implicit executes for blob and auto-increment operations not visible to clients.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count`

  The number of times a thread has been blocked by this MySQL Server (SQL node) while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit implicit executes for blob and auto-increment operations not visible to clients.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_meta_request_count_session`

  The number of times a thread has been blocked in this client session waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_wait_meta_request_count_slave`

  The number of times a thread has been blocked by this slave waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_wait_meta_request_count`

  The number of times a thread has been blocked by this MySQL Server (SQL node) waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_wait_nanos_count_session`

  Total time (in nanoseconds) spent in this client session waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_wait_nanos_count_slave`

  Total time (in nanoseconds) spent by this slave waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_wait_nanos_count`

  Total time (in nanoseconds) spent by this MySQL Server (SQL node) waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, "NDB API Statistics Counters and Variables".

- `Ndb_api_wait_scan_result_count_session`

  The number of times a thread has been blocked in this client session while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_scan_result_count_slave`

  The number of times a thread has been blocked by this slave while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replication slave, or does not use NDB tables, this value is always 0.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_scan_result_count`

  The number of times a thread has been blocked by this MySQL Server (SQL node) while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 7.15, “NDB API Statistics Counters and Variables”.

- `Ndb_cluster_node_id`

  If the server is acting as a MySQL Cluster node, then the value of this variable its node ID in the cluster.

  If the server is not part of a MySQL Cluster, then the value of this variable is 0.

- `Ndb_config_from_host`

  If the server is part of a MySQL Cluster, the value of this variable is the host name or IP address of the Cluster management server from which it gets its configuration data.

  If the server is not part of a MySQL Cluster, then the value of this variable is an empty string.

- `Ndb_config_from_port`

  If the server is part of a MySQL Cluster, the value of this variable is the number of the port through which it is connected to the Cluster management server from which it gets its configuration data.

  If the server is not part of a MySQL Cluster, then the value of this variable is 0.

- `Ndb_conflict_fn_max_del_win`

  Shows the number of times that a row was rejected on the current SQL node due to MySQL Cluster Replication conflict resolution using `NDB$MAX_DELETE_WIN()`, since the last time that this `mysqld` was started.

  For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_max`

  Used in MySQL Cluster Replication conflict resolution, this variable shows the number of times that a row was not applied on the current SQL node due to “greatest timestamp wins” conflict resolution since the last time that this `mysqld` was started.

  For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_old`

Used in MySQL Cluster Replication conflict resolution, this variable shows the number of times that a row was not applied as the result of “same timestamp wins” conflict resolution on a given `mysqld` since the last time it was restarted.

For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_epoch`

  Used in MySQL Cluster Replication conflict resolution, this variable shows the number of rows found to be in conflict using `NDB$EPOCH()` conflict resolution on a given `mysqld` since the last time it was restarted.

  For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_epoch2`

  Shows the number of rows found to be in conflict in MySQL Cluster Replication conflict resolution, when using `NDB$EPOCH2()`, on the master designated as the primary since the last time it was restarted.

  Added in MySQL Cluster NDB 7.4.2.

  For more information, see NDB$EPOCH2().

- `Ndb_conflict_fn_epoch_trans`

  Used in MySQL Cluster Replication conflict resolution, this variable shows the number of rows found to be in conflict using `NDB$EPOCH_TRANS()` conflict resolution on a given `mysqld` since the last time it was restarted.

  For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_epoch2_trans`

  Used in MySQL Cluster Replication conflict resolution, this variable shows the number of rows found to be in conflict using `NDB$EPOCH_TRANS2()` conflict resolution on a given `mysqld` since the last time it was restarted.

  Added in MySQL Cluster NDB 7.4.2.

  For more information, see NDB$EPOCH2_TRANS().

- `Ndb_conflict_last_conflict_epoch`

  The most recent epoch in which a conflict was detected on this slave. You can compare this value with `Ndb_slave_max_replicated_epoch`; if `Ndb_slave_max_replicated_epoch` is greater than `Ndb_conflict_last_conflict_epoch`, no conflicts have yet been detected.

  This variable was added in MySQL Cluster NDB 7.4.2.

  See Section 8.11, “MySQL Cluster Replication Conflict Resolution”, for more information.

- `Ndb_conflict_reflected_op_discard_count`

  When using MySQL Cluster Replication conflict resolution, this is the number of reflected operations that were not applied on the secondary, due to encountering an error during execution.

  This variable was added in MySQL Cluster NDB 7.4.2.

  See Section 8.11, “MySQL Cluster Replication Conflict Resolution”, for more information.

- `Ndb_conflict_reflected_op_prepare_count`

When using conflict resolution with MySQL Cluster Replication, this status variable contains the number of reflected operations that have been defined (that is, prepared for execution on the secondary).

This variable was added in MySQL Cluster NDB 7.4.2.

See Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- `Ndb_conflict_refresh_op_count`

  When using conflict resolution with MySQL Cluster Replication, this gives the number of refresh operations that have been prepared for execution on the secondary.

  This variable was added in MySQL Cluster NDB 7.4.2.

  See Section 8.11, "MySQL Cluster Replication Conflict Resolution", for more information.

- `Ndb_conflict_last_stable_epoch`

  Number of rows found to be in conflict by a transactional conflict function

  This variable was added in MySQL Cluster NDB 7.4.2.

  See Section 8.11, "MySQL Cluster Replication Conflict Resolution", for more information.

- `Ndb_conflict_trans_row_conflict_count`

  Used in MySQL Cluster Replication conflict resolution, this status variable shows the number of rows found to be directly in-conflict by a transactional conflict function on a given `mysqld` since the last time it was restarted.

  Currently, the only transactional conflict detection function supported by MySQL Cluster is NDB$EPOCH_TRANS(), so this status variable is effectively the same as `Ndb_conflict_fn_epoch_trans`.

  For more information, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- `Ndb_conflict_trans_row_reject_count`

  Used in MySQL Cluster Replication conflict resolution, this status variable shows the total number of rows realigned due to being determined as conflicting by a transactional conflict detection function. This includes not only `Ndb_conflict_trans_row_conflict_count`, but any rows in or dependent on conflicting transactions.

  For more information, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- `Ndb_conflict_trans_reject_count`

  Used in MySQL Cluster Replication conflict resolution, this status variable shows the number of transactions found to be in conflict by a transactional conflict detection function.

  For more information, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- `Ndb_conflict_trans_detect_iter_count`

  Used in MySQL Cluster Replication conflict resolution, this shows the number of internal iterations required to commit an epoch transaction. Should be (slightly) greater than or equal to `Ndb_conflict_trans_conflict_commit_count`.

  For more information, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

- `Ndb_conflict_trans_conflict_commit_count`

Used in MySQL Cluster Replication conflict resolution, this shows the number of epoch transactions committed after they required transactional conflict handling.

For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

- `Ndb_epoch_delete_delete_count`

  When using delete-delete conflict detection, this is the number of delete-delete conflicts detected, where a delete operation is applied, but the indicated row does not exist.

  Added in MySQL Cluster NDB 7.4.2.

- `Ndb_execute_count`

  Provides the number of round trips to the `NDB` kernel made by operations.

- `Ndb_last_commit_epoch_server`

  The epoch most recently committed by `NDB`.

  This variable was added in MySQL Cluster NDB 7.3.8 and MySQL Cluster NDB 7.4.1.

- `Ndb_last_commit_epoch_session`

  The epoch most recently committed by this `NDB` client.

  This variable was added in MySQL Cluster NDB 7.3.8 and MySQL Cluster NDB 7.4.1.

- `Ndb_number_of_data_nodes`

  If the server is part of a MySQL Cluster, the value of this variable is the number of data nodes in the cluster.

  If the server is not part of a MySQL Cluster, then the value of this variable is 0.

- `Ndb_pushed_queries_defined`

  The total number of joins pushed down to the NDB kernel for distributed handling on the data nodes.

  > **Note**
  >
  > Joins tested using `EXPLAIN` that can be pushed down contribute to this number.

- `Ndb_pushed_queries_dropped`

  The number of joins that were pushed down to the NDB kernel but that could not be handled there.

- `Ndb_pushed_queries_executed`

  The number of joins successfully pushed down to `NDB` and executed there.

- `Ndb_pushed_reads`

  The number of rows returned to `mysqld` from the NDB kernel by joins that were pushed down.

  > **Note**
  >
  > Executing `EXPLAIN` on joins that can be pushed down to `NDB` does not add to this number.

- `Ndb_pruned_scan_count`

This variable holds a count of the number of scans executed by NDBCLUSTER since the MySQL Cluster was last started where NDBCLUSTER was able to use partition pruning.

Using this variable together with Ndb_scan_count can be helpful in schema design to maximize the ability of the server to prune scans to a single table partition, thereby involving only a single data node.

- Ndb_scan_count

  This variable holds a count of the total number of scans executed by NDBCLUSTER since the MySQL Cluster was last started.

- Ndb_slave_max_replicated_epoch

  The most recently committed epoch on this slave. In MySQL Cluster NDB 7.4.1 and later, you can compare this value with Ndb_conflict_last_conflict_epoch; if Ndb_slave_max_replicated_epoch is the greater of the two, no conflicts have yet been detected.

  This variable was added in MySQL Cluster NDB 7.3.8 and MySQL Cluster NDB 7.4.1.

  For more information, see Section 8.11, “MySQL Cluster Replication Conflict Resolution”.

5.3.9 MySQL Cluster TCP/IP Connections

TCP/IP is the default transport mechanism for all connections between nodes in a MySQL Cluster. Normally it is not necessary to define TCP/IP connections; MySQL Cluster automatically sets up such connections for all data nodes, management nodes, and SQL or API nodes.

> **Note**
>
> For an exception to this rule, see Section 5.3.10, “MySQL Cluster TCP/IP Connections Using Direct Connections”.

To override the default connection parameters, it is necessary to define a connection using one or more [tcp] sections in the config.ini file. Each [tcp] section explicitly defines a TCP/IP connection between two MySQL Cluster nodes, and must contain at a minimum the parameters NodeId1 and NodeId2, as well as any connection parameters to override.

It is also possible to change the default values for these parameters by setting them in the [tcp default] section.

> **Important**
>
> Any [tcp] sections in the config.ini file should be listed *last*, following all other sections in the file. However, this is not required for a [tcp default] section. This requirement is a known issue with the way in which the config.ini file is read by the MySQL Cluster management server.

Connection parameters which can be set in [tcp] and [tcp default] sections of the config.ini file are listed here:

- NodeId1

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	[none]	...	N

  To identify a connection between two nodes it is necessary to provide their node IDs in the [tcp] section of the configuration file as the values of NodeId1 and NodeId2. These are the same unique Id values for each of these nodes as described in Section 5.3.7, “Defining SQL and Other API Nodes in a MySQL Cluster”.

- NodeId2

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	[none]	...	N

To identify a connection between two nodes it is necessary to provide their node IDs in the [tcp] section of the configuration file as the values of NodeId1 and NodeId2. These are the same unique Id values for each of these nodes as described in Section 5.3.7, “Defining SQL and Other API Nodes in a MySQL Cluster”.

- HostName1

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

The HostName1 and HostName2 parameters can be used to specify specific network interfaces to be used for a given TCP connection between two nodes. The values used for these parameters can be host names or IP addresses.

- HostName2

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

The HostName1 and HostName2 parameters can be used to specify specific network interfaces to be used for a given TCP connection between two nodes. The values used for these parameters can be host names or IP addresses.

- OverloadLimit

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	0	0 - 4294967039 (0xFFFFFEFF)	N

When more than this many unsent bytes are in the send buffer, the connection is considered overloaded.

This parameter can be used to determine the amount of unsent data that must be present in the send buffer before the connection is considered overloaded. See Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”, for more information.

- SendBufferMemory

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	2M	256K - 4294967039 (0xFFFFFEFF)	N

TCP transporters use a buffer to store all messages before performing the send call to the operating system. When this buffer reaches 64KB its contents are sent; these are also sent when a round of messages have been executed. To handle temporary overload situations it is also possible to define a bigger send buffer.

If this parameter is set explicitly, then the memory is not dedicated to each transporter; instead, the value used denotes the hard limit for how much memory (out of the total available memory—that is, TotalSendBufferMemory) that may be used by a single transporter. For more information about configuring dynamic transporter send buffer memory allocation in MySQL Cluster, see Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”.

The default size of the send buffer is 2MB, which is the size recommended in most situations. The minimum size is 64 KB; the theoretical maximum is 4 GB.

- SendSignalId

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	[see text]	true, false	N

To be able to retrace a distributed message datagram, it is necessary to identify each message. When this parameter is set to Y, message IDs are transported over the network. This feature is disabled by default in production builds, and enabled in -debug builds.

- Checksum

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

This parameter is a boolean parameter (enabled by setting it to Y or 1, disabled by setting it to N or 0). It is disabled by default. When it is enabled, checksums for all messages are calculated before they placed in the send buffer. This feature ensures that messages are not corrupted while waiting in the send buffer, or by the transport mechanism.

- PortNumber (*OBSOLETE*)

This formerly specified the port number to be used for listening for connections from other nodes. This parameter should no longer be used; use the ServerPort data node configuration parameter for this purpose instead.

- ReceiveBufferMemory

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	2M	16K - 4294967039 (0xFFFFFEFF)	N

Specifies the size of the buffer used when receiving data from the TCP/IP socket.

The default value of this parameter is 2MB. The minimum possible value is 16KB; the theoretical maximum is 4GB.

- TCP_RCV_BUF_SIZE

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	70080	1 - 2G	N
NDB 7.3.1	unsigned	0	0 - 2G	N

Determines the size of the receive buffer set during TCP transporter initialization. Prior to MySQL Cluster NDB 7.3.1, the default was 70080 and the minimum was 1. In MySQL Cluster NDB 7.3.1 and later, the default and minimum value is 0, which allows the operating system or platform to set this value. The default is recommended for most common usage cases.

- TCP_SND_BUF_SIZE

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	71540	1 - 2G	N

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.1	unsigned	0	0 - 2G	N

Determines the size of the send buffer set during TCP transporter initialization. Prior to MySQL Cluster NDB 7.3.1, the default was 71540 and the minimum was 1. In MySQL Cluster NDB 7.3.1 and later, the default and minimum value is 0, which allows the operating system or platform to set this value. The default is recommended for most common usage cases.

- `TCP_MAXSEG_SIZE`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 2G	N

Determines the size of the memory set during TCP transporter initialization. The default is recommended for most common usage cases.

- `TcpBind_INADDR_ANY`

Setting this parameter to `TRUE` or `1` binds `IP_ADDR_ANY` so that connections can be made from anywhere (for autogenerated connections). The default is `FALSE` (`0`).

5.3.10 MySQL Cluster TCP/IP Connections Using Direct Connections

Setting up a cluster using direct connections between data nodes requires specifying explicitly the crossover IP addresses of the data nodes so connected in the `[tcp]` section of the cluster `config.ini` file.

In the following example, we envision a cluster with at least four hosts, one each for a management server, an SQL node, and two data nodes. The cluster as a whole resides on the `172.23.72.*` subnet of a LAN. In addition to the usual network connections, the two data nodes are connected directly using a standard crossover cable, and communicate with one another directly using IP addresses in the `1.1.0.*` address range as shown:

```
# Management Server
[ndb_mgmd]
Id=1
HostName=172.23.72.20
# SQL Node
[mysqld]
Id=2
HostName=172.23.72.21
# Data Nodes
[ndbd]
Id=3
HostName=172.23.72.22
[ndbd]
Id=4
HostName=172.23.72.23
# TCP/IP Connections
[tcp]
NodeId1=3
NodeId2=4
HostName1=1.1.0.1
HostName2=1.1.0.2
```

The `HostName1` and `HostName2` parameters are used only when specifying direct connections.

The use of direct TCP connections between data nodes can improve the cluster's overall efficiency by enabling the data nodes to bypass an Ethernet device such as a switch, hub, or router, thus cutting down on the cluster's latency.

> **Note**
>
> To take the best advantage of direct connections in this fashion with more than two data nodes, you must have a direct connection between each data node and every other data node in the same node group.

5.3.11 MySQL Cluster Shared-Memory Connections

MySQL Cluster attempts to use the shared memory transporter and configure it automatically where possible. `[shm]` sections in the `config.ini` file explicitly define shared-memory connections between nodes in the cluster. When explicitly defining shared memory as the connection method, it is necessary to define at least `NodeId1`, `NodeId2`, and `ShmKey`. All other parameters have default values that should work well in most cases.

> **Important**
>
> *SHM functionality is considered experimental only*. It is not officially supported in any current MySQL Cluster release, and testing results indicate that SHM performance is not appreciably greater than when using TCP/IP for the transporter.
>
> For these reasons, you must determine for yourself or by using our free resources (forums, mailing lists) whether SHM can be made to work correctly in your specific case.

- `NodeId1`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	[none]	...	N

To identify a connection between two nodes it is necessary to provide node identifiers for each of them, as `NodeId1` and `NodeId2`.

- `NodeId2`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	[none]	...	N

To identify a connection between two nodes it is necessary to provide node identifiers for each of them, as `NodeId1` and `NodeId2`.

- `HostName1`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given SHM connection between two nodes. The values used for these parameters can be host names or IP addresses.

- `HostName2`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

The HostName1 and HostName2 parameters can be used to specify specific network interfaces to be used for a given SHM connection between two nodes. The values used for these parameters can be host names or IP addresses.

- OverloadLimit

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	0	0 - 4294967039 (0xFFFFFEFF)	N

When more than this many unsent bytes are in the send buffer, the connection is considered overloaded.

This parameter can be used to determine the amount of unsent data that must be present in the send buffer before the connection is considered overloaded. See Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”, for more information.

- ShmKey

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	0 - 4294967039 (0xFFFFFEFF)	N

When setting up shared memory segments, a node ID, expressed as an integer, is used to identify uniquely the shared memory segment to use for the communication. There is no default value.

- ShmSize

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	1M	64K - 4294967039 (0xFFFFFEFF)	N

Each SHM connection has a shared memory segment where messages between nodes are placed by the sender and read by the reader. The size of this segment is defined by ShmSize. The default value is 1MB.

- SendSignalId

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

To retrace the path of a distributed message, it is necessary to provide each message with a unique identifier. Setting this parameter to Y causes these message IDs to be transported over the network as well. This feature is disabled by default in production builds, and enabled in -debug builds.

- Checksum

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	true	true, false	N

This parameter is a boolean (Y/N) parameter which is disabled by default. When it is enabled, checksums for all messages are calculated before being placed in the send buffer.

This feature prevents messages from being corrupted while waiting in the send buffer. It also serves as a check against data being corrupted during transport.

- SigNum

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	0 - 4294967039 (0xFFFFFEFF)	N

When using the shared memory transporter, a process sends an operating system signal to the other process when there is new data available in the shared memory. Should that signal conflict with an existing signal, this parameter can be used to change it. This is a possibility when using SHM due to the fact that different operating systems use different signal numbers.

The default value of `SigNum` is 0; therefore, it must be set to avoid errors in the cluster log when using the shared memory transporter. Typically, this parameter is set to 10 in the `[shm default]` section of the `config.ini` file.

5.3.12 SCI Transport Connections in MySQL Cluster

`[sci]` sections in the `config.ini` file explicitly define SCI (Scalable Coherent Interface) connections between cluster nodes. Using SCI transporters in MySQL Cluster is supported only when the MySQL binaries are built using `--with-ndb-sci=/your/path/to/SCI`. The *`path`* should point to a directory that contains at a minimum `lib` and `include` directories containing SISCI libraries and header files. (See Section 5.4, “Using High-Speed Interconnects with MySQL Cluster” for more information about SCI.)

In addition, SCI requires specialized hardware.

It is strongly recommended to use SCI Transporters only for communication between `ndbd` processes. Using SCI Transporters means that the `ndbd` processes never sleep. For this reason, SCI Transporters should be used only on machines having at least two CPUs dedicated for use by `ndbd` processes. There should be at least one CPU per `ndbd` process, with at least one CPU left in reserve to handle operating system activities.

- `NodeId1`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	[none]	...	N

To identify a connection between two nodes it is necessary to provide node identifiers for each of them, as `NodeId1` and `NodeId2`.

- `NodeId2`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	numeric	[none]	...	N

To identify a connection between two nodes it is necessary to provide node identifiers for each of them, as `NodeId1` and `NodeId2`.

- `Host1SciId0`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	0 - 4294967039 (0xFFFFFEFF)	N

This identifies the SCI node ID on the first Cluster node (identified by `NodeId1`).

- `Host1SciId1`

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 4294967039 (0xFFFFFEFF)	N

It is possible to set up SCI Transporters for failover between two SCI cards which then should use separate networks between the nodes. This identifies the node ID and the second SCI card to be used on the first node.

- Host2SciId0

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	[none]	0 - 4294967039 (0xFFFFFEFF)	N

This identifies the SCI node ID on the second Cluster node (identified by `NodeId2`).

- Host2SciId1

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	0	0 - 4294967039 (0xFFFFFEFF)	N

When using two SCI cards to provide failover, this parameter identifies the second SCI card to be used on the second node.

- HostName1

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given SCI connection between two nodes. The values used for these parameters can be host names or IP addresses.

- HostName2

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	name or IP address	[none]	...	N

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given SCI connection between two nodes. The values used for these parameters can be host names or IP addresses.

- SharedBufferSize

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	10M	64K - 4294967039 (0xFFFFFEFF)	N

Each SCI transporter has a shared memory segment used for communication between the two nodes. Setting the size of this segment to the default value of 1MB should be sufficient for most applications. Using a smaller value can lead to problems when performing many parallel inserts; if the shared buffer is too small, this can also result in a crash of the `ndbd` process.

- SendLimit

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	unsigned	8K	128 - 32K	N

A small buffer in front of the SCI media stores messages before transmitting them over the SCI network. By default, this is set to 8KB. Our benchmarks show that performance is best at 64KB but 16KB reaches within a few percent of this, and there was little if any advantage to increasing it beyond 8KB.

- SendSignalId

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	true	true, false	N

To trace a distributed message it is necessary to identify each message uniquely. When this parameter is set to `Y`, message IDs are transported over the network. This feature is disabled by default in production builds, and enabled in `-debug` builds.

- Checksum

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	boolean	false	true, false	N

This parameter is a boolean value, and is disabled by default. When `Checksum` is enabled, checksums are calculated for all messages before they are placed in the send buffer. This feature prevents messages from being corrupted while waiting in the send buffer. It also serves as a check against data being corrupted during transport.

- OverloadLimit

Effective Version	Type/Units	Default	Range/Values	Restart Type
NDB 7.3.0	bytes	0	0 - 4294967039 (0xFFFFFEFF)	N

When more than this many unsent bytes are in the send buffer, the connection is considered overloaded. See Section 5.3.13, “Configuring MySQL Cluster Send Buffer Parameters”, for more information.

5.3.13 Configuring MySQL Cluster Send Buffer Parameters

Formerly, the NDB kernel employed a send buffer whose size was fixed at 2MB for each node in the cluster, this buffer being allocated when the node started. Because the size of this buffer could not be changed after the cluster was started, it was necessary to make it large enough in advance to accommodate the maximum possible load on any transporter socket. However, this was an inefficient use of memory, since much of it often went unused, and could result in large amounts of resources being wasted when scaling up to many API nodes.

This problem was eventually solved (in MySQL Cluster NDB 7.0) by employing a unified send buffer whose memory is allocated dynamically from a pool shared by all transporters. This means that the size of the send buffer can be adjusted as necessary. Configuration of the unified send buffer can accomplished by setting the following parameters:

- **TotalSendBufferMemory.** This parameter can be set for all types of MySQL Cluster nodes—that is, it can be set in the `[ndbd]`, `[mgm]`, and `[api]` (or `[mysql]`) sections of the `config.ini` file. It represents the total amount of memory (in bytes) to be allocated by each node for which it is set for use among all configured transporters. If set, its minimum is 256KB; the maximum is 4294967039.

To be backward-compatible with existing configurations, this parameter takes as its default value the sum of the maximum send buffer sizes of all configured transporters, plus an additional 32KB (one page) per transporter. The maximum depends on the type of transporter, as shown in the following table:

Transporter	Maximum Send Buffer Size (bytes)
TCP	`SendBufferMemory` (default = 2M)
SCI	`SendLimit` (default = 8K) plus 16K
SHM	20K

This enables existing configurations to function in close to the same way as they did with MySQL Cluster NDB 6.3 and earlier, with the same amount of memory and send buffer space available to each transporter. However, memory that is unused by one transporter is not available to other transporters.

- **OverloadLimit.** This parameter is used in the `config.ini` file `[tcp]` section, and denotes the amount of unsent data (in bytes) that must be present in the send buffer before the connection is considered overloaded. When such an overload condition occurs, transactions that affect the overloaded connection fail with NDB API Error 1218 (`Send Buffers overloaded in NDB kernel`) until the overload status passes. The default value is 0, in which case the effective overload limit is calculated as `SendBufferMemory * 0.8` for a given connection. The maximum value for this parameter is 4G.

- **SendBufferMemory.** This value denotes a hard limit for the amount of memory that may be used by a single transporter out of the entire pool specified by `TotalSendBufferMemory`. However, the sum of `SendBufferMemory` for all configured transporters may be greater than the `TotalSendBufferMemory` that is set for a given node. This is a way to save memory when many nodes are in use, as long as the maximum amount of memory is never required by all transporters at the same time.

- **ReservedSendBufferMemory.** This optional data node parameter, if set, gives an amount of memory (in bytes) that is reserved for connections between data nodes; this memory is not allocated to send buffers used for communications with management servers or API nodes. This provides a way to protect the cluster against misbehaving API nodes that use excess send memory and thus cause failures in communications internally in the NDB kernel. If set, its the minimum permitted value for this parameters is 256KB; the maximum is 4294967039.

5.4 Using High-Speed Interconnects with MySQL Cluster

Even before design of `NDBCLUSTER` began in 1996, it was evident that one of the major problems to be encountered in building parallel databases would be communication between the nodes in the network. For this reason, `NDBCLUSTER` was designed from the very beginning to permit the use of a number of different data transport mechanisms. In this Manual, we use the term *transporter* for these.

The MySQL Cluster codebase provides for four different transporters:

- *TCP/IP using 100 Mbps or gigabit Ethernet*, as discussed in Section 5.3.9, “MySQL Cluster TCP/IP Connections”.

- *Direct (machine-to-machine) TCP/IP*; although this transporter uses the same TCP/IP protocol as mentioned in the previous item, it requires setting up the hardware differently and is configured differently as well. For this reason, it is considered a separate transport mechanism for MySQL Cluster. See Section 5.3.10, “MySQL Cluster TCP/IP Connections Using Direct Connections”, for details.

- *Shared memory (SHM)*. For more information about SHM, see Section 5.3.11, “MySQL Cluster Shared-Memory Connections”.

> **Note**
>
> SHM is considered experimental only, and is not officially supported.

- *Scalable Coherent Interface (SCI)*, as described in the next section of this chapter, Section 5.3.12, "SCI Transport Connections in MySQL Cluster".

Most users today employ TCP/IP over Ethernet because it is ubiquitous. TCP/IP is also by far the best-tested transporter for use with MySQL Cluster.

We are working to make sure that communication with the ndbd process is made in "chunks" that are as large as possible because this benefits all types of data transmission.

For users who desire it, it is also possible to use cluster interconnects to enhance performance even further. There are two ways to achieve this: Either a custom transporter can be designed to handle this case, or you can use socket implementations that bypass the TCP/IP stack to one extent or another. We have experimented with both of these techniques using the SCI (Scalable Coherent Interface) technology developed by Dolphin Interconnect Solutions.

5.4.1 Configuring MySQL Cluster to use SCI Sockets

It is possible employing Scalable Coherent Interface (SCI) technology to achieve a significant increase in connection speeds and throughput between MySQL Cluster data and SQL nodes. To use SCI, it is necessary to obtain and install Dolphin SCI network cards and to use the drivers and other software supplied by Dolphin. You can get information on obtaining these, from Dolphin Interconnect Solutions. SCI SuperSocket or SCI Transporter support is available for 32-bit and 64-bit Linux, Solaris, Windows, and other platforms. See the Dolphin documentation referenced later in this section for more detailed information regarding platforms supported for SCI.

Once you have acquired the required Dolphin hardware and software, you can obtain detailed information on how to adapt a MySQL Cluster configured for normal TCP/IP communication to use SCI from the from the Dolphin SCI online documentation.

5.4.2 MySQL Cluster Interconnects and Performance

The ndbd process has a number of simple constructs which are used to access the data in a MySQL Cluster. We have created a very simple benchmark to check the performance of each of these and the effects which various interconnects have on their performance.

There are four access methods:

- **Primary key access.** This is access of a record through its primary key. In the simplest case, only one record is accessed at a time, which means that the full cost of setting up a number of TCP/IP messages and a number of costs for context switching are borne by this single request. In the case where multiple primary key accesses are sent in one batch, those accesses share the cost of setting up the necessary TCP/IP messages and context switches. If the TCP/IP messages are for different destinations, additional TCP/IP messages need to be set up.

- **Unique key access.** Unique key accesses are similar to primary key accesses, except that a unique key access is executed as a read on an index table followed by a primary key access on the table. However, only one request is sent from the MySQL Server, and the read of the index table is handled by ndbd. Such requests also benefit from batching.

- **Full table scan.** When no indexes exist for a lookup on a table, a full table scan is performed. This is sent as a single request to the ndbd process, which then divides the table scan into a set of parallel scans on all cluster ndbd processes. In future versions of MySQL Cluster, an SQL node will be able to filter some of these scans.

- **Range scan using ordered index.** When an ordered index is used, it performs a scan in the same manner as the full table scan, except that it scans only those records which are in the range used by the query transmitted by the MySQL server (SQL node). All partitions are scanned in parallel when all bound index attributes include all attributes in the partitioning key.

With benchmarks developed internally by MySQL for testing simple and batched primary and unique key accesses, we have found that using SCI sockets improves performance by approximately 100% over TCP/IP, except in rare instances when communication performance is not an issue. This can occur when scan filters make up most of processing time or when very large batches of primary key accesses are achieved. In that case, the CPU processing in the ndbd processes becomes a fairly large part of the overhead.

Using the SCI transporter instead of SCI Sockets is only of interest in communicating between ndbd processes. Using the SCI transporter is also only of interest if a CPU can be dedicated to the ndbd process because the SCI transporter ensures that this process will never go to sleep. It is also important to ensure that the ndbd process priority is set in such a way that the process does not lose priority due to running for an extended period of time, as can be done by locking processes to CPUs in Linux 2.6. If such a configuration is possible, the ndbd process will benefit by 10−70% as compared with using SCI sockets. (The larger figures will be seen when performing updates and probably on parallel scan operations as well.)

There are several other optimized socket implementations for computer clusters, including Myrinet, Gigabit Ethernet, Infiniband and the VIA interface. However, we have tested MySQL Cluster so far only with SCI sockets. See Section 5.4.1, “Configuring MySQL Cluster to use SCI Sockets”, for information on how to set up SCI sockets using ordinary TCP/IP for MySQL Cluster.

Chapter 6 MySQL Cluster Programs

Table of Contents

Using and managing a MySQL Cluster requires several specialized programs, which we describe in this chapter. We discuss the purposes of these programs in a MySQL Cluster, how to use the programs, and what startup options are available for each of them.

These programs include the MySQL Cluster data, management, and SQL node processes (ndbd, ndbmtd, ndb_mgmd, and mysqld) and the management client (ndb_mgm).

Information about the program ndb_setup.py (added in MySQL Cluster NDB 7.3.1), used to start the MySQL Cluster Auto-Installer, is also included in this section. You should be aware that Section 6.23, "ndb_setup.py — Start browser-based Auto-Installer for MySQL Cluster", contains information about the command-line client only; for information about using the GUI installer spawned by this program to configure and deploy a MySQL Cluster, see Section 4.1, "The MySQL Cluster Auto-Installer".

For information about using mysqld as a MySQL Cluster process, see Section 7.4, "MySQL Server Usage for MySQL Cluster".

Other NDB utility, diagnostic, and example programs are included with the MySQL Cluster distribution. These include ndb_restore, ndb_show_tables, and ndb_config. These programs are also covered in this section.

The final portion of this section contains tables of options that are common to all the various MySQL Cluster programs.

6.1 ndbd — The MySQL Cluster Data Node Daemon

ndbd is the process that is used to handle all the data in tables using the NDB Cluster storage engine. This is the process that empowers a data node to accomplish distributed transaction handling, node recovery, checkpointing to disk, online backup, and related tasks.

In a MySQL Cluster, a set of ndbd processes cooperate in handling data. These processes can execute on the same computer (host) or on different computers. The correspondences between data nodes and Cluster hosts is completely configurable.

The following table includes command options specific to the MySQL Cluster data node program ndbd. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndbd), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.1 This table describes command-line options for the ndbd program

Format	Description	Added or Removed
`--initial`	Perform initial start of ndbd, including cleaning the file system. Consult the documentation before using this option	All MySQL 5.6 based releases
`--nostart`, `-n`	Don't start ndbd immediately; ndbd waits for command to start from ndb_mgmd	All MySQL 5.6 based releases
`--daemon`, `-d`	Start ndbd as daemon (default); override with --nodaemon	All MySQL 5.6 based releases
`--nodaemon`	Do not start ndbd as daemon; provided for testing purposes	All MySQL 5.6 based releases
`--foreground`	Run ndbd in foreground, provided for debugging purposes (implies --nodaemon)	All MySQL 5.6 based releases
`--nowait-nodes=list`	Do not wait for these data nodes to start (takes comma-separated list of node IDs). Also requires --ndb-nodeid to be used.	All MySQL 5.6 based releases
`--initial-start`	Perform partial initial start (requires --nowait-nodes)	All MySQL 5.6 based releases
`--bind-address=name`	Local bind address	All MySQL 5.6 based releases
`--install[=name]`	Used to install the data node process as a Windows service. Does not apply on non-Windows platforms.	All MySQL 5.6 based releases
`--remove[=name]`	Used to remove a data node process that was previously installed as a Windows service. Does not apply on non-Windows platforms.	All MySQL 5.6 based releases
`--connect-retries=#`	Set the number of times to retry a connection before giving up; 0 means 1 attempt only (and no retries)	All MySQL 5.6 based releases

Format	Description	Added or Removed
--connect-delay=#	Time to wait between attempts to contact a management server, in seconds; 0 means do not wait between attempts	All MySQL 5.6 based releases
--connect-retry-delay=#	Time to wait between attempts to contact a management server, in seconds; 0 means do not wait between attempts	ADDED: NDB 7.4.9

> **Note**
>
> All of these options also apply to the multi-threaded version of this program (ndbmtd) and you may substitute “ndbmtd” for “ndbd” wherever the latter occurs in this section.

- --bind-address

Command-Line Format	--bind-address=name	
Permitted Values	Type	string
	Default	

  Causes ndbd to bind to a specific network interface (host name or IP address). This option has no default value.

- --daemon, -d

Command-Line Format	--daemon	
Permitted Values	Type	boolean
	Default	TRUE

  Instructs ndbd or ndbmtd to execute as a daemon process. This is the default behavior. --nodaemon can be used to prevent the process from running as a daemon.

  This option has no effect when running ndbd or ndbmtd on Windows platforms.

- --nodaemon

Command-Line Format	--nodaemon	
Permitted Values	Type	boolean
	Default	FALSE

  Prevents ndbd or ndbmtd from executing as a daemon process. This option overrides the --daemon option. This is useful for redirecting output to the screen when debugging the binary.

  The default behavior for ndbd and ndbmtd on Windows is to run in the foreground, making this option unnecessary on Windows platforms, where it has no effect.

- --foreground

Command-Line Format	--foreground	
Permitted Values	Type	boolean
	Default	FALSE

  Causes ndbd or ndbmtd to execute as a foreground process, primarily for debugging purposes. This option implies the --nodaemon option.

This option has no effect when running ndbd or ndbmtd on Windows platforms.

- --initial

Command-Line Format	--initial	
Permitted Values	**Type**	boolean
	Default	FALSE

Instructs ndbd to perform an initial start. An initial start erases any files created for recovery purposes by earlier instances of ndbd. It also re-creates recovery log files. On some operating systems, this process can take a substantial amount of time.

An --initial start is to be used *only* when starting the ndbd process under very special circumstances; this is because this option causes all files to be removed from the MySQL Cluster file system and all redo log files to be re-created. These circumstances are listed here:

- When performing a software upgrade which has changed the contents of any files.
- When restarting the node with a new version of ndbd.
- As a measure of last resort when for some reason the node restart or system restart repeatedly fails. In this case, be aware that this node can no longer be used to restore data due to the destruction of the data files.

Use of this option prevents the StartPartialTimeout and StartPartitionedTimeout configuration parameters from having any effect.

> **Important**
>
> This option does *not* affect either of the following types of files:
>
> - Backup files that have already been created by the affected node
> - MySQL Cluster Disk Data files (see Section 7.12, "MySQL Cluster Disk Data Tables").
>
> This option also has no effect on recovery of data by a data node that is just starting (or restarting) from data nodes that are already running. This recovery of data occurs automatically, and requires no user intervention in a MySQL Cluster that is running normally.

It is permissible to use this option when starting the cluster for the very first time (that is, before any data node files have been created); however, it is *not* necessary to do so.

- --initial-start

Command-Line Format	--initial-start	
Permitted Values	**Type**	boolean
	Default	FALSE

This option is used when performing a partial initial start of the cluster. Each node should be started with this option, as well as --nowait-nodes.

Suppose that you have a 4-node cluster whose data nodes have the IDs 2, 3, 4, and 5, and you wish to perform a partial initial start using only nodes 2, 4, and 5—that is, omitting node 3:

```
shell> ndbd --ndb-nodeid=2 --nowait-nodes=3 --initial-start
shell> ndbd --ndb-nodeid=4 --nowait-nodes=3 --initial-start
```

```
shell> ndbd --ndb-nodeid=5 --nowait-nodes=3 --initial-start
```

When using this option, you must also specify the node ID for the data node being started with the `--ndb-nodeid` option.

> **Important**
>
> Do not confuse this option with the `--nowait-nodes` option for `ndb_mgmd`, which can be used to enable a cluster configured with multiple management servers to be started without all management servers being online.

- `--nowait-nodes=`*`node_id_1`*`[, `*`node_id_2`*`[, ...]]`

Command-Line Format	`--nowait-nodes=list`	
Permitted Values	Type	`string`
	Default	

This option takes a list of data nodes which for which the cluster will not wait for before starting.

This can be used to start the cluster in a partitioned state. For example, to start the cluster with only half of the data nodes (nodes 2, 3, 4, and 5) running in a 4-node cluster, you can start each `ndbd` process with `--nowait-nodes=3,5`. In this case, the cluster starts as soon as nodes 2 and 4 connect, and does *not* wait `StartPartitionedTimeout` milliseconds for nodes 3 and 5 to connect as it would otherwise.

If you wanted to start up the same cluster as in the previous example without one `ndbd` (say, for example, that the host machine for node 3 has suffered a hardware failure) then start nodes 2, 4, and 5 with `--nowait-nodes=3`. Then the cluster will start as soon as nodes 2, 4, and 5 connect and will not wait for node 3 to start.

- `--nostart`, `-n`

Command-Line Format	`--nostart`	
Permitted Values	Type	`boolean`
	Default	`FALSE`

Instructs `ndbd` not to start automatically. When this option is used, `ndbd` connects to the management server, obtains configuration data from it, and initializes communication objects. However, it does not actually start the execution engine until specifically requested to do so by the management server. This can be accomplished by issuing the proper `START` command in the management client (see Section 7.2, “Commands in the MySQL Cluster Management Client”).

- `--install[=`*`name`*`]`

Command-Line Format	`--install[=name]`	
Platform Specific	Windows	
Permitted Values	Type	`string`
	Default	`ndbd`

Causes `ndbd` to be installed as a Windows service. Optionally, you can specify a name for the service; if not set, the service name defaults to `ndbd`. Although it is preferable to specify other `ndbd` program options in a `my.ini` or `my.cnf` configuration file, it is possible to use together with `--install`. However, in such cases, the `--install` option must be specified first, before any other options are given, for the Windows service installation to succeed.

It is generally not advisable to use this option together with the `--initial` option, since this causes the data node file system to be wiped and rebuilt every time the service is stopped and started.

Extreme care should also be taken if you intend to use any of the other ndbd options that affect the starting of data nodes—including --initial-start, --nostart, and --nowait-nodes—together with --install, and you should make absolutely certain you fully understand and allow for any possible consequences of doing so.

The --install option has no effect on non-Windows platforms.

- --remove[=*name*]

Command-Line Format	--remove[=name]	
Platform Specific	Windows	
Permitted Values	Type	string
	Default	ndbd

Causes an ndbd process that was previously installed as a Windows service to be removed. Optionally, you can specify a name for the service to be uninstalled; if not set, the service name defaults to ndbd.

The --remove option has no effect on non-Windows platforms.

- --connect-retries=#

Command-Line Format	--connect-retries=#	
Permitted Values	Type	numeric
	Default	12
	Min Value	0
	Max Value	65535

Set the number of times to retry a connection before giving up; 0 means 1 attempt only (and no retries). The default is 12 attempts. The time to wait between attempts is controlled by the --connect-retry-delay option in MySQL NDB 7.4.9 and later (previously, this was --connect-delay).

- --connect-delay=#

Deprecated	5.6.28-ndb-7.4.9	
Command-Line Format	--connect-delay=#	
Permitted Values	Type	numeric
	Default	5
	Min Value	0
	Max Value	3600

Determines the time to wait between attempts to contact a management server when starting (the number of attempts is controlled by the --connect-retries option). The default is 5 seconds.

This option is deprecated in MySQL Cluster NDB 7.4.9, and is subject to removal in a future release of MySQL Cluster. Use --connect-retry-delay instead.

- --connect-retry-delay=#

Introduced	5.6.28-ndb-7.4.9

Command-Line Format	--connect-retry-delay=#	
Permitted Values (>= 5.6.28-ndb-7.4.9)	**Type**	numeric
	Default	5
	Min Value	0
	Max Value	4294967295

Determines the time to wait between attempts to contact a management server when starting (the time between attempts is controlled by the `--connect-retries` option). The default is 5 seconds.

This option was added in MySQL Cluster NDB 7.4.9, and is intended to take the place of the `--connect-delay` option, which is now deprecated and subject to removal in a future release of MySQL Cluster.

`ndbd` generates a set of log files which are placed in the directory specified by `DataDir` in the `config.ini` configuration file.

These log files are listed below. *`node_id`* is and represents the node's unique identifier. For example, `ndb_2_error.log` is the error log generated by the data node whose node ID is `2`.

- `ndb_node_id_error.log` is a file containing records of all crashes which the referenced `ndbd` process has encountered. Each record in this file contains a brief error string and a reference to a trace file for this crash. A typical entry in this file might appear as shown here:

```
Date/Time: Saturday 30 July 2004 - 00:20:01
Type of error: error
Message: Internal program error (failed ndbrequire)
Fault ID: 2341
Problem data: DbtupFixAlloc.cpp
Object of reference: DBTUP (Line: 173)
ProgramName: NDB Kernel
ProcessID: 14909
TraceFile: ndb_2_trace.log.2
***EOM***
```

  Listings of possible `ndbd` exit codes and messages generated when a data node process shuts down prematurely can be found in ndbd Error Messages.

  > **Important**
  >
  > *The last entry in the error log file is not necessarily the newest one* (nor is it likely to be). Entries in the error log are *not* listed in chronological order; rather, they correspond to the order of the trace files as determined in the `ndb_node_id_trace.log.next` file (see below). Error log entries are thus overwritten in a cyclical and not sequential fashion.

- `ndb_node_id_trace.log.trace_id` is a trace file describing exactly what happened just before the error occurred. This information is useful for analysis by the MySQL Cluster development team.

  It is possible to configure the number of these trace files that will be created before old files are overwritten. *`trace_id`* is a number which is incremented for each successive trace file.

- `ndb_node_id_trace.log.next` is the file that keeps track of the next trace file number to be assigned.

- `ndb_node_id_out.log` is a file containing any data output by the `ndbd` process. This file is created only if `ndbd` is started as a daemon, which is the default behavior.

- ndb_*node_id*.pid is a file containing the process ID of the ndbd process when started as a daemon. It also functions as a lock file to avoid the starting of nodes with the same identifier.
- ndb_*node_id*_signal.log is a file used only in debug versions of ndbd, where it is possible to trace all incoming, outgoing, and internal messages with their data in the ndbd process.

It is recommended not to use a directory mounted through NFS because in some environments this can cause problems whereby the lock on the .pid file remains in effect even after the process has terminated.

To start ndbd, it may also be necessary to specify the host name of the management server and the port on which it is listening. Optionally, one may also specify the node ID that the process is to use.

```
shell> ndbd --connect-string="nodeid=2;host=ndb_mgmd.mysql.com:1186"
```

See Section 5.3.3, “MySQL Cluster Connection Strings”, for additional information about this issue. Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”, describes other command-line options which can be used with ndbd. For information about data node configuration parameters, see Section 5.3.6, “Defining MySQL Cluster Data Nodes”.

When ndbd starts, it actually initiates two processes. The first of these is called the “angel process”; its only job is to discover when the execution process has been completed, and then to restart the ndbd process if it is configured to do so. Thus, if you attempt to kill ndbd using the Unix kill command, it is necessary to kill both processes, beginning with the angel process. The preferred method of terminating an ndbd process is to use the management client and stop the process from there.

The execution process uses one thread for reading, writing, and scanning data, as well as all other activities. This thread is implemented asynchronously so that it can easily handle thousands of concurrent actions. In addition, a watch-dog thread supervises the execution thread to make sure that it does not hang in an endless loop. A pool of threads handles file I/O, with each thread able to handle one open file. Threads can also be used for transporter connections by the transporters in the ndbd process. In a multi-processor system performing a large number of operations (including updates), the ndbd process can consume up to 2 CPUs if permitted to do so.

For a machine with many CPUs it is possible to use several ndbd processes which belong to different node groups; however, such a configuration is still considered experimental and is not supported for MySQL 5.6 in a production setting. See Section 3.6, “Known Limitations of MySQL Cluster”.

6.2 ndbinfo_select_all — Select From ndbinfo Tables

ndbinfo_select_all is a client program that selects all rows and columns from one or more tables in the ndbinfo database. It is included with the MySQL Cluster distribution beginning with MySQL Cluster NDB 7.2.2.

Not all ndbinfo tables can be accessed by this program. ndbinfo_select_all can access the counters, diskpagebuffer, logbuffers, logspaces, nodes, resources, threadblocks, threadstat, and transporters tables.

To select from one or more ndbinfo tables using ndbinfo_select_all, it is necessary to supply the names of the tables when invoking the program as shown here:

```
shell> ndbinfo_select_all table_name1  [table_name2] [...]
```

For example:

```
shell> ndbinfo_select_all logbuffers logspaces
== logbuffers ==
node_id log_type        log_id  log_part        total   used    high
5       0       0       0       33554432        262144  0
6       0       0       0       33554432        262144  0
7       0       0       0       33554432        262144  0
```

```
8       0       0       0       33554432        262144  0
== logspaces ==
node_id log_type        log_id  log_part        total   used    high
5       0       0       0       268435456       0       0
5       0       0       1       268435456       0       0
5       0       0       2       268435456       0       0
5       0       0       3       268435456       0       0
6       0       0       0       268435456       0       0
6       0       0       1       268435456       0       0
6       0       0       2       268435456       0       0
6       0       0       3       268435456       0       0
7       0       0       0       268435456       0       0
7       0       0       1       268435456       0       0
7       0       0       2       268435456       0       0
7       0       0       3       268435456       0       0
8       0       0       0       268435456       0       0
8       0       0       1       268435456       0       0
8       0       0       2       268435456       0       0
8       0       0       3       268435456       0       0
shell>
```

The following table includes options that are specific to `ndbinfo_select_all`. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including `ndbinfo_select_all`), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.2 This table describes command-line options for the ndbinfo_select_all program

Format	Description	Added or Removed
`--delay=#`	Set the delay in seconds between loops. Default is 5.	All MySQL 5.6 based releases
`--loops=#`, `-l`	Set the number of times to perform the select. Default is 1.	All MySQL 5.6 based releases
`--database=db_name`, `-d`	Name of the database where the table located.	All MySQL 5.6 based releases
`--parallelism=#`, `-p`	Set the degree of parallelism.	All MySQL 5.6 based releases

- `--delay=seconds`

Command-Line Format	`--delay=#`	
Permitted Values	**Type**	`numeric`
	Default	`5`
	Min Value	`0`
	Max Value	`MAX_INT`

  This option sets the number of seconds to wait between executing loops. Has no effect if `--loops` is set to 0 or 1.

- `--loops=number`, `-l` *`number`*

Command-Line Format	`--loops=#`	
Permitted Values	**Type**	`numeric`
	Default	`1`

	Min Value	0
	Max Value	MAX_INT

This option sets the number of times to execute the select. Use --delay to set the time between loops.

6.3 ndbmtd — The MySQL Cluster Data Node Daemon (Multi-Threaded)

ndbmtd is a multi-threaded version of ndbd, the process that is used to handle all the data in tables using the NDBCLUSTER storage engine. ndbmtd is intended for use on host computers having multiple CPU cores. Except where otherwise noted, ndbmtd functions in the same way as ndbd; therefore, in this section, we concentrate on the ways in which ndbmtd differs from ndbd, and you should consult Section 6.1, “ndbd — The MySQL Cluster Data Node Daemon”, for additional information about running MySQL Cluster data nodes that apply to both the single-threaded and multi-threaded versions of the data node process.

Command-line options and configuration parameters used with ndbd also apply to ndbmtd. For more information about these options and parameters, see Section 6.1, “ndbd — The MySQL Cluster Data Node Daemon”, and Section 5.3.6, “Defining MySQL Cluster Data Nodes”, respectively.

ndbmtd is also file system-compatible with ndbd. In other words, a data node running ndbd can be stopped, the binary replaced with ndbmtd, and then restarted without any loss of data. (However, when doing this, you must make sure that MaxNoOfExecutionThreads is set to an apppriate value before restarting the node if you wish for ndbmtd to run in multi-threaded fashion.) Similarly, an ndbmtd binary can be replaced with ndbd simply by stopping the node and then starting ndbd in place of the multi-threaded binary. It is not necessary when switching between the two to start the data node binary using --initial.

Using ndbmtd differs from using ndbd in two key respects:

1. Because ndbmtd runs by default in single-threaded mode (that is, it behaves like ndbd), you must configure it to use multiple threads. This can be done by setting an appropriate value in the config.ini file for the MaxNoOfExecutionThreads configuration parameter or the ThreadConfig configuration parameter. Using MaxNoOfExecutionThreads is simpler, but ThreadConfig offers more flexibility. For more information about these configuration parameters and their use, see Multi-Threading Configuration Parameters (ndbmtd).

2. Trace files are generated by critical errors in ndbmtd processes in a somewhat different fashion from how these are generated by ndbd failures. These differences are discussed in more detail in the next few paragraphs.

Like ndbd, ndbmtd generates a set of log files which are placed in the directory specified by DataDir in the config.ini configuration file. Except for trace files, these are generated in the same way and have the same names as those generated by ndbd.

In the event of a critical error, ndbmtd generates trace files describing what happened just prior to the error' occurrence. These files, which can be found in the data node's DataDir, are useful for analysis of problems by the MySQL Cluster Development and Support teams. One trace file is generated for each ndbmtd thread. The names of these files have the following pattern:

```
ndb_node_id_trace.log.trace_id_tthread_id,
```

In this pattern, *node_id* stands for the data node's unique node ID in the cluster, *trace_id* is a trace sequence number, and *thread_id* is the thread ID. For example, in the event of the failure of an ndbmtd process running as a MySQL Cluster data node having the node ID 3 and with MaxNoOfExecutionThreads equal to 4, four trace files are generated in the data node's data directory. If the is the first time this node has failed, then these files are named ndb_3_trace.log.1_t1, ndb_3_trace.log.1_t2, ndb_3_trace.log.1_t3, and ndb_3_trace.log.1_t4. Internally, these trace files follow the same format as ndbd trace files.

The ndbd exit codes and messages that are generated when a data node process shuts down prematurely are also used by ndbmtd. See ndbd Error Messages, for a listing of these.

> **Note**
>
> It is possible to use ndbd and ndbmtd concurrently on different data nodes in the same MySQL Cluster. However, such configurations have not been tested extensively; thus, we cannot recommend doing so in a production setting at this time.

6.4 ndb_mgmd — The MySQL Cluster Management Server Daemon

The management server is the process that reads the cluster configuration file and distributes this information to all nodes in the cluster that request it. It also maintains a log of cluster activities. Management clients can connect to the management server and check the cluster's status.

The following table includes options that are specific to the MySQL Cluster management server program ndb_mgmd. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_mgmd), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.3 This table describes command-line options for the ndb_mgmd program

Format	Description	Added or Removed
--config-file=file, -f	Specify the cluster configuration file; in NDB-6.4.0 and later, needs --reload or --initial to override configuration cache if present	All MySQL 5.6 based releases
--configdir=directory, --config-dir=directory	Specify the cluster management server's configuration cache directory	All MySQL 5.6 based releases
--bind-address=ip_address	Local bind address	All MySQL 5.6 based releases
--print-full-config, -P	Print full configuration and exit	All MySQL 5.6 based releases
--daemon, -d	Run ndb_mgmd in daemon mode (default)	All MySQL 5.6 based releases
--nodaemon	Do not run ndb_mgmd as a daemon	All MySQL 5.6 based releases
--interactive	Run ndb_mgmd in interactive mode (not officially supported in production; for testing purposes only)	All MySQL 5.6 based releases
--log-name=name	A name to use when writing messages applying to this node in the cluster log.	All MySQL 5.6 based releases
--no-nodeid-checks	Do not provide any node id checks	All MySQL 5.6 based releases

Format	Description	Added or Removed
--mycnf	Read cluster configuration data from the my.cnf file	All MySQL 5.6 based releases
--reload	Causes the management server to compare the configuration file with its configuration cache	All MySQL 5.6 based releases
--initial	Causes the management server reload its configuration data from the configuration file, bypassing the configuration cache	All MySQL 5.6 based releases
--nowait-nodes=list	Do not wait for these management nodes when starting this management server. Also requires --ndb-nodeid to be used.	All MySQL 5.6 based releases
--config-cache[=TRUE|FALSE]	Enable the management server configuration cache; TRUE by default.	All MySQL 5.6 based releases
--install[=name]	Used to install the management server process as a Windows service. Does not apply on non-Windows platforms.	All MySQL 5.6 based releases
--remove[=name]	Used to remove a management server process that was previously installed as a Windows service, optionally specifying the name of the service to be removed. Does not apply on non-Windows platforms.	All MySQL 5.6 based releases

- `--bind-address=`*`host`*`[:`*`port`*`]`

Command-Line Format	--bind-address=ip_address	
Permitted Values	Type	string
	Default	[none]

  When specified, this option limits management server connections by management clients to clients at the specified host name or IP address (and possibly port, if this is also specified). In such cases, a management client attempting to connect to the management server from any other address fails with the error `Unable to setup port: `*`host`*`:`*`port`*`!`

  If the *`port`* is not specified, the management client attempts to use port 1186.

- `--no-nodeid-checks`

Command-Line Format	--no-nodeid-checks	
Permitted Values	Type	boolean
	Default	FALSE

  Do not perform any checks of node IDs.

- `--configdir=`*`dir_name`*

Command-Line Format	--configdir=directory	
	--config-dir=directory	
Permitted Values	Type	file name
	Default	$INSTALLDIR/mysql-cluster

Specifies the cluster management server's configuration cache directory. `--config-dir` is an alias for this option.

- `--config-cache`

Command-Line Format	--config-cache[=TRUE|FALSE]	
Permitted Values	Type	boolean
	Default	TRUE

This option, whose default value is `1` (or `TRUE`, or `ON`), can be used to disable the management server's configuration cache, so that it reads its configuration from `config.ini` every time it starts (see Section 5.3, "MySQL Cluster Configuration Files"). You can do this by starting the `ndb_mgmd` process with any one of the following options:

- `--config-cache=0`
- `--config-cache=FALSE`
- `--config-cache=OFF`
- `--skip-config-cache`

Using one of the options just listed is effective only if the management server has no stored configuration at the time it is started. If the management server finds any configuration cache files, then the `--config-cache` option or the `--skip-config-cache` option is ignored. Therefore, to disable configuration caching, the option should be used the *first* time that the management server is started. Otherwise—that is, if you wish to disable configuration caching for a management server that has *already* created a configuration cache—you must stop the management server, delete any existing configuration cache files manually, then restart the management server with `--skip-config-cache` (or with `--config-cache` set equal to 0, `OFF`, or `FALSE`).

Configuration cache files are normally created in a directory named `mysql-cluster` under the installation directory (unless this location has been overridden using the `--configdir` option). Each time the management server updates its configuration data, it writes a new cache file. The files are named sequentially in order of creation using the following format:

```
ndb_node-id_config.bin.seq-number
```

`node-id` is the management server's node ID; *`seq-number`* is a sequence number, beginning with 1. For example, if the management server's node ID is 5, then the first three configuration cache files would, when they are created, be named `ndb_5_config.bin.1`, `ndb_5_config.bin.2`, and `ndb_5_config.bin.3`.

If your intent is to purge or reload the configuration cache without actually disabling caching, you should start `ndb_mgmd` with one of the options `--reload` or `--initial` instead of `--skip-config-cache`.

To re-enable the configuration cache, simply restart the management server, but without the `--config-cache` or `--skip-config-cache` option that was used previously to disable the configuration cache.

ndb_mgmd does not check for the configuration directory (--configdir) or attempts to create one when --skip-config-cache is used. (Bug #13428853)

- --config-file=*filename*, -f *filename*

Command-Line Format	--config-file=file	
Permitted Values	**Type**	file name
	Default	[none]

Instructs the management server as to which file it should use for its configuration file. By default, the management server looks for a file named config.ini in the same directory as the ndb_mgmd executable; otherwise the file name and location must be specified explicitly.

This option has no default value, and is ignored unless the management server is forced to read the configuration file, either because ndb_mgmd was started with the --reload or --initial option, or because the management server could not find any configuration cache. This option is also read if ndb_mgmd was started with --config-cache=OFF. See Section 5.3, “MySQL Cluster Configuration Files”, for more information.

Formerly, using this option together with --initial caused removal of the configuration cache even if the file was not found. This issue was resolved in MySQL Cluster NDB 7.3.2. (Bug #1299289)

- --mycnf

Command-Line Format	--mycnf	
Permitted Values	**Type**	boolean
	Default	FALSE

Read configuration data from the my.cnf file.

- --daemon, -d

Command-Line Format	--daemon	
Permitted Values	**Type**	boolean
	Default	TRUE

Instructs ndb_mgmd to start as a daemon process. This is the default behavior.

This option has no effect when running ndb_mgmd on Windows platforms.

- --interactive

Command-Line Format	--interactive	
Permitted Values	**Type**	boolean
	Default	FALSE

Starts ndb_mgmd in interactive mode; that is, an ndb_mgm client session is started as soon as the management server is running. This option does not start any other MySQL Cluster nodes.

- --initial

Command-Line Format	--initial	
Permitted Values	**Type**	boolean
	Default	FALSE

Configuration data is cached internally, rather than being read from the cluster global configuration file each time the management server is started (see Section 5.3, “MySQL Cluster Configuration Files”). Using the --initial option overrides this behavior, by forcing the management server to delete any existing cache files, and then to re-read the configuration data from the cluster configuration file and to build a new cache.

This differs in two ways from the --reload option. First, --reload forces the server to check the configuration file against the cache and reload its data only if the contents of the file are different from the cache. Second, --reload does not delete any existing cache files.

If ndb_mgmd is invoked with --initial but cannot find a global configuration file, the management server cannot start.

When a management server starts, it checks for another management server in the same MySQL Cluster and tries to use the other management server's configuration data; ndb_mgmd ignores --initial unless it is the only management server running. This behavior also has implications when performing a rolling restart of a MySQL Cluster with multiple management nodes. See Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”, for more information.

Formerly, using this option together with the --config-file option caused removal of the configuration cache even if the file was not found. Starting with MySQL Cluster NDB 7.3.2, the cache is cleared in such cases only if the configuration file is actually found. (Bug #1299289)

- --log-name=*name*

Command-Line Format	--log-name=name	
Permitted Values	Type	string
	Default	MgmtSrvr

Provides a name to be used for this node in the cluster log.

- --nodaemon

Command-Line Format	--nodaemon	
Permitted Values	Type	boolean
	Default	FALSE

Instructs ndb_mgmd not to start as a daemon process.

The default behavior for ndb_mgmd on Windows is to run in the foreground, making this option unnecessary on Windows platforms.

- --print-full-config, -P

Command-Line Format	--print-full-config	
Permitted Values	Type	boolean
	Default	FALSE

Shows extended information regarding the configuration of the cluster. With this option on the command line the ndb_mgmd process prints information about the cluster setup including an extensive list of the cluster configuration sections as well as parameters and their values. Normally used together with the --config-file (-f) option.

- --reload

Command-Line Format	--reload

Permitted Values	**Type**	boolean
	Default	FALSE

In MySQL Cluster NDB 7.3, configuration data is stored internally rather than being read from the cluster global configuration file each time the management server is started (see Section 5.3, "MySQL Cluster Configuration Files"). Using this option forces the management server to check its internal data store against the cluster configuration file and to reload the configuration if it finds that the configuration file does not match the cache. Existing configuration cache files are preserved, but not used.

This differs in two ways from the `--initial` option. First, `--initial` causes all cache files to be deleted. Second, `--initial` forces the management server to re-read the global configuration file and construct a new cache.

If the management server cannot find a global configuration file, then the `--reload` option is ignored.

When a management server starts, it checks for another management server in the same MySQL Cluster and tries to use the other management server's configuration data; `ndb_mgmd` ignores `--reload` unless it is the only management server running. This behavior also has implications when performing a rolling restart of a MySQL Cluster with multiple management nodes. See Section 7.5, "Performing a Rolling Restart of a MySQL Cluster", for more information.

- `--nowait-nodes`

Command-Line Format	--nowait-nodes=list	
Permitted Values	**Type**	numeric
	Default	
	Min Value	1
	Max Value	255

When starting a MySQL Cluster is configured with two management nodes, each management server normally checks to see whether the other `ndb_mgmd` is also operational and whether the other management server's configuration is identical to its own. However, it is sometimes desirable to start the cluster with only one management node (and perhaps to allow the other `ndb_mgmd` to be started later). This option causes the management node to bypass any checks for any other management nodes whose node IDs are passed to this option, permitting the cluster to start as though configured to use only the management node that was started.

For purposes of illustration, consider the following portion of a `config.ini` file (where we have omitted most of the configuration parameters that are not relevant to this example):

```
[ndbd]
NodeId = 1
HostName = 192.168.0.101
[ndbd]
NodeId = 2
HostName = 192.168.0.102
[ndbd]
NodeId = 3
HostName = 192.168.0.103
[ndbd]
NodeId = 4
HostName = 192.168.0.104
[ndb_mgmd]
NodeId = 10
HostName = 192.168.0.150
```

```
[ndb_mgmd]
NodeId = 11
HostName = 192.168.0.151
[api]
NodeId = 20
HostName = 192.168.0.200
[api]
NodeId = 21
HostName = 192.168.0.201
```

Assume that you wish to start this cluster using only the management server having node ID `10` and running on the host having the IP address 192.168.0.150. (Suppose, for example, that the host computer on which you intend to the other management server is temporarily unavailable due to a hardware failure, and you are waiting for it to be repaired.) To start the cluster in this way, use a command line on the machine at 192.168.0.150 to enter the following command:

```
shell> ndb_mgmd --ndb-nodeid=10 --nowait-nodes=11
```

As shown in the preceding example, when using `--nowait-nodes`, you must also use the `--ndb-nodeid` option to specify the node ID of this `ndb_mgmd` process.

You can then start each of the cluster's data nodes in the usual way. If you wish to start and use the second management server in addition to the first management server at a later time without restarting the data nodes, you must start each data node with a connection string that references both management servers, like this:

```
shell> ndbd -c 192.168.0.150,192.168.0.151
```

The same is true with regard to the connection string used with any `mysqld` processes that you wish to start as MySQL Cluster SQL nodes connected to this cluster. See Section 5.3.3, "MySQL Cluster Connection Strings", for more information.

When used with `ndb_mgmd`, this option affects the behavior of the management node with regard to other management nodes only. Do not confuse it with the `--nowait-nodes` option used with `ndbd` or `ndbmtd` to permit a cluster to start with fewer than its full complement of data nodes; when used with data nodes, this option affects their behavior only with regard to other data nodes.

Multiple management node IDs may be passed to this option as a comma-separated list. Each node ID must be no less than 1 and no greater than 255. In practice, it is quite rare to use more than two management servers for the same MySQL Cluster (or to have any need for doing so); in most cases you need to pass to this option only the single node ID for the one management server that you do not wish to use when starting the cluster.

> **Note**
>
> When you later start the "missing" management server, its configuration must match that of the management server that is already in use by the cluster. Otherwise, it fails the configuration check performed by the existing management server, and does not start.

It is not strictly necessary to specify a connection string when starting the management server. However, if you are using more than one management server, a connection string should be provided and each node in the cluster should specify its node ID explicitly.

See Section 5.3.3, "MySQL Cluster Connection Strings", for information about using connection strings. Section 6.4, "`ndb_mgmd` — The MySQL Cluster Management Server Daemon", describes other options for `ndb_mgmd`.

The following files are created or used by `ndb_mgmd` in its starting directory, and are placed in the `DataDir` as specified in the `config.ini` configuration file. In the list that follows, *`node_id`* is the unique node identifier.

- config.ini is the configuration file for the cluster as a whole. This file is created by the user and read by the management server. Chapter 5, *Configuration of MySQL Cluster*, discusses how to set up this file.

- ndb_*node_id*_cluster.log is the cluster events log file. Examples of such events include checkpoint startup and completion, node startup events, node failures, and levels of memory usage. A complete listing of cluster events with descriptions may be found in Chapter 7, *Management of MySQL Cluster*.

  By default, when the size of the cluster log reaches one million bytes, the file is renamed to ndb_*node_id*_cluster.log.*seq_id*, where *seq_id* is the sequence number of the cluster log file. (For example: If files with the sequence numbers 1, 2, and 3 already exist, the next log file is named using the number 4.) You can change the size and number of files, and other characteristics of the cluster log, using the LogDestination configuration parameter.

- ndb_*node_id*_out.log is the file used for stdout and stderr when running the management server as a daemon.

- ndb_*node_id*.pid is the process ID file used when running the management server as a daemon.

- --install[=*name*]

Command-Line Format	--install[=name]	
Platform Specific	Windows	
Permitted Values	**Type**	string
	Default	ndb_mgmd

  Causes ndb_mgmd to be installed as a Windows service. Optionally, you can specify a name for the service; if not set, the service name defaults to ndb_mgmd. Although it is preferable to specify other ndb_mgmd program options in a my.ini or my.cnf configuration file, it is possible to use them together with --install. However, in such cases, the --install option must be specified first, before any other options are given, for the Windows service installation to succeed.

  It is generally not advisable to use this option together with the --initial option, since this causes the configuration cache to be wiped and rebuilt every time the service is stopped and started. Care should also be taken if you intend to use any other ndb_mgmd options that affect the starting of the management server, and you should make absolutely certain you fully understand and allow for any possible consequences of doing so.

  The --install option has no effect on non-Windows platforms.

- --remove[=*name*]

Command-Line Format	--remove[=name]	
Platform Specific	Windows	
Permitted Values	**Type**	string
	Default	ndb_mgmd

  Causes an ndb_mgmd process that was previously installed as a Windows service to be removed. Optionally, you can specify a name for the service to be uninstalled; if not set, the service name defaults to ndb_mgmd.

  The --remove option has no effect on non-Windows platforms.

6.5 ndb_mgm — The MySQL Cluster Management Client

The ndb_mgm management client process is actually not needed to run the cluster. Its value lies in providing a set of commands for checking the cluster's status, starting backups, and performing other administrative functions. The management client accesses the management server using a C API. Advanced users can also employ this API for programming dedicated management processes to perform tasks similar to those performed by ndb_mgm.

To start the management client, it is necessary to supply the host name and port number of the management server:

```
shell> ndb_mgm [host_name [port_num]]
```

For example:

```
shell> ndb_mgm ndb_mgmd.mysql.com 1186
```

The default host name and port number are localhost and 1186, respectively.

The following table includes options that are specific to the MySQL Cluster management client program ndb_mgm. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_mgm), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.4 This table describes command-line options for the ndb_mgm program

Format	Description	Added or Removed
--connect-retries=#	Set the number of times to retry a connection before giving up; 0 means 1 attempt only (and no retries)	ADDED: NDB 7.4.9
--try-reconnect=#, -t	Set the number of times to retry a connection before giving up; synonym for --connect-retries	All MySQL 5.6 based releases
--execute=name, -e	Execute command and exit	All MySQL 5.6 based releases

- --connect-retries=#

Introduced	5.6.28-ndb-7.4.9	
Command-Line Format	--connect-retries=#	
Permitted Values (>= 5.6.28-ndb-7.4.9)	**Type**	numeric
	Default	3
	Min Value	0
	Max Value	4294967295

  This option specifies the number of times following the first attempt to retry a connection before giving up (the client always tries the connection at least once). The length of time to wait per attempt is set using --connect-retry-delay.

  This option was added in MySQL Cluster NDB 7.4.9, and is synonymous with the --try-reconnect option, which is now deprecated.

  The default for this option this option differs from its default when used with other NDB programs. See Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”, for more information.

- --execute=command, -e command

Command-Line Format	`--execute=name`

This option can be used to send a command to the MySQL Cluster management client from the system shell. For example, either of the following is equivalent to executing `SHOW` in the management client:

```
shell> ndb_mgm -e "SHOW"
shell> ndb_mgm --execute="SHOW"
```

This is analogous to how the `--execute` or `-e` option works with the `mysql` command-line client. See Using Options on the Command Line.

> **Note**
>
> If the management client command to be passed using this option contains any space characters, then the command *must* be enclosed in quotation marks. Either single or double quotation marks may be used. If the management client command contains no space characters, the quotation marks are optional.

- `--try-reconnect=`*`number`*

Deprecated	5.6.28-ndb-7.4.9	
Command-Line Format	`--try-reconnect=#`	
Permitted Values	**Type**	`integer`
	Default	`3`
	Min Value	`0`
	Max Value	`4294967295`

If the connection to the management server is broken, the node tries to reconnect to it every 5 seconds until it succeeds. By using this option, it is possible to limit the number of attempts to *`number`* before giving up and reporting an error instead.

This option is deprecated in MySQL Cluster NDB 7.4.9 and later, and is superseded by `--connect-retries`.

Additional information about using `ndb_mgm` can be found in Section 7.2, "Commands in the MySQL Cluster Management Client".

6.6 `ndb_blob_tool` — Check and Repair BLOB and TEXT columns of MySQL Cluster Tables

This tool can be used to check for and remove orphaned BLOB column parts from `NDB` tables, as well as to generate a file listing any orphaned parts. It is sometimes useful in diagnosing and repairing corrupted or damaged `NDB` tables containing `BLOB` or `TEXT` columns.

The basic syntax for `ndb_blob_tool` is shown here:

```
ndb_blob_tool [options] table [column, ...]
```

Unless you use the `--help` option, you must specify an action to be performed by including one or more of the options `--check-orphans`, `--delete-orphans`, or `--dump-file`. These options cause `ndb_blob_tool` to check for orphaned BLOB parts, remove any orphaned BLOB parts, and generate a dump file listing orphaned BLOB parts, respectively, and are described in more detail later in this section.

You must also specify the name of a table when invoking ndb_blob_tool. In addition, you can optionally follow the table name with the (comma-separated) names of one or more BLOB or TEXT columns from that table. If no columns are listed, the tool works on all of the table's BLOB and TEXT columns. If you need to specify a database, use the --database (-d) option.

The --verbose option provides additional information in the output about the tool's progress.

The following table includes options that are specific to ndb_blob_tool. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_blob_tool), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.5 This table describes command-line options for the ndb_blob_tool program

Format	Description	Added or Removed
--check-orphans	Check for orphan blob parts	All MySQL 5.6 based releases
--database=db_name, -d	Database to find the table in.	All MySQL 5.6 based releases
--delete-orphans	Delete orphan blob parts	All MySQL 5.6 based releases
--dump-file=file	Write orphan keys to specified file	All MySQL 5.6 based releases
--verbose, -v	Verbose output	All MySQL 5.6 based releases

- --check-orphans

Command-Line Format	--check-orphans	
Permitted Values	Type	boolean
	Default	FALSE

Check for orphaned BLOB parts in MySQL Cluster tables.

- --database=*db_name*, -d

Command-Line Format	--database=db_name	
Permitted Values	Type	string
	Default	[none]

Specify the database to find the table in.

- --delete-orphans

Command-Line Format	--delete-orphans	
Permitted Values	Type	boolean
	Default	FALSE

Remove orphaned BLOB parts from MySQL Cluster tables.

- --dump-file=*file*

Command-Line Format	--dump-file=file	
Permitted Values	Type	file name

	Default	[none]

Writes a list of orphaned BLOB column parts to *file*. The information written to the file includes the table key and BLOB part number for each orphaned BLOB part.

- `--verbose`

Command-Line Format	--verbose	
Permitted Values	**Type**	boolean
	Default	FALSE

Provide extra information in the tool's output regarding its progress.

Example

First we create an `NDB` table in the `test` database, using the `CREATE TABLE` statement shown here:

```
USE test;
CREATE TABLE btest (
    c0 BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    c1 TEXT,
    c2 BLOB
)   ENGINE=NDB;
```

Then we insert a few rows into this table, using a series of statements similar to this one:

```
INSERT INTO btest VALUES (NULL, 'x', REPEAT('x', 1000));
```

When run with `--check-orphans` against this table, `ndb_blob_tool` generates the following output:

```
shell> ndb_blob_tool --check-orphans --verbose -d test btest
connected
processing 2 blobs
processing blob #0 c1 NDB$BLOB_19_1
NDB$BLOB_19_1: nextResult: res=1
total parts: 0
orphan parts: 0
processing blob #1 c2 NDB$BLOB_19_2
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=1
total parts: 10
orphan parts: 0
disconnected
NDBT_ProgramExit: 0 - OK
```

The tool reports that there are no `NDB` BLOB column parts associated with column `c1`, even though `c1` is a `TEXT` column. This is due to the fact that, in an `NDB` table, only the first 256 bytes of a `BLOB` or `TEXT` column value are stored inline, and only the excess, if any, is stored separately; thus, if there are no values using more than 256 bytes in a given column of one of these types, no `BLOB` column parts are created by `NDB` for this column. See Data Type Storage Requirements, for more information.

6.7 `ndb_config` — Extract MySQL Cluster Configuration Information

This tool extracts current configuration information for data nodes, SQL nodes, and API nodes from one of a number of sources: a MySQL Cluster management node, or its config.ini or my.cnf file. By default, the management node is the source for the configuration data; to override the default, execute ndb_config with the --config-file or --mycnf option. It is also possible to use a data node as the source by specifying its node ID with --config_from_node=*node_id*.

ndb_config can also provide an offline dump of all configuration parameters which can be used, along with their default, maximum, and minimum values and other information. The dump can be produced in either text or XML format; for more information, see the discussion of the --configinfo and --xml [284] options later in this section).

You can filter the results by section (DB, SYSTEM, or CONNECTIONS) using one of the options --nodes, --system, or --connections.

The following table includes options that are specific to ndb_config. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_config), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.6 This table describes command-line options for the ndb_config program

Format	Description	Added or Removed
--nodes	Print node information ([ndbd] or [ndbd default] section of cluster configuration file) only. Cannot be used with --system or --connections.	All MySQL 5.6 based releases
--connections	Print connections information ([tcp], [tcp default], [sci], [sci default], [shm], or [shm default] sections of cluster configuration file) only. Cannot be used with --system or --nodes.	All MySQL 5.6 based releases
--query=string, -q	One or more query options (attributes)	All MySQL 5.6 based releases
--host=name	Specify host	All MySQL 5.6 based releases
--type=name	Specify node type	All MySQL 5.6 based releases
--nodeid [282], --id [282]	Get configuration of node with this ID	All MySQL 5.6 based releases
--fields=string, -f	Field separator	All MySQL 5.6 based releases
--rows=string, -r	Row separator	All MySQL 5.6 based releases
--config-file=file_name	Set the path to config.ini file	All MySQL 5.6 based releases
--mycnf	Read configuration data from my.cnf file	All MySQL 5.6 based releases
-c	Short form for --ndb-connectstring	All MySQL 5.6 based releases
--configinfo	Dumps information about all NDB configuration parameters in text format with default, maximum, and minimum values.	All MySQL 5.6 based releases

Format	Description	Added or Removed
	Use with --xml to obtain XML output.	
--configinfo --xml [284]	Use --xml with --configinfo to obtain a dump of all NDB configuration parameters in XML format with default, maximum, and minimum values.	All MySQL 5.6 based releases
--system	Print SYSTEM section information only (see ndb_config --configinfo output). Cannot be used with --nodes or --connections.	All MySQL 5.6 based releases
--config_from_node=#	Obtain configuration data from the node having this ID (must be a data node).	All MySQL 5.6 based releases

- `--usage`, `--help`, or `-?`

Command-Line Format	--help
	--usage

Causes `ndb_config` to print a list of available options, and then exit.

- `--config_from_node=#`

Command-Line Format	--config_from_node=#	
Permitted Values	Type	numeric
	Default	none
	Min Value	1
	Max Value	48

Obtain the cluster's configuration data from the data node that has this ID.

If the node having this ID is not a data node, `ndb_config` fails with an error. (To obtain configuration data from the management node instead, simply omit this option.)

- `--version`, `-V`

Command-Line Format	--version

Causes `ndb_config` to print a version information string, and then exit.

- `--ndb-connectstring=connection_string`, `-c connection_string`

Command-Line Format	--ndb-connectstring=connectstring	
	--connect-string=connectstring	
Permitted Values	Type	string
	Default	localhost:1186

Specifies the connection string to use in connecting to the management server. The format for the connection string is the same as described in Section 5.3.3, “MySQL Cluster Connection Strings”, and defaults to localhost:1186.

- --config-file=*path-to-file*

Command-Line Format	--config-file=file_name	
Permitted Values	Type	file name
	Default	

Gives the path to the management server's configuration file (config.ini). This may be a relative or absolute path. If the management node resides on a different host from the one on which ndb_config is invoked, then an absolute path must be used.

- --mycnf

Command-Line Format	--mycnf	
Permitted Values	Type	boolean
	Default	FALSE

Read configuration data from the my.cnf file.

- --query=*query-options*, -q *query-options*

Command-Line Format	--query=string	
Permitted Values	Type	string
	Default	

This is a comma-delimited list of *query options*—that is, a list of one or more node attributes to be returned. These include id (node ID), type (node type—that is, ndbd, mysqld, or ndb_mgmd), and any configuration parameters whose values are to be obtained.

For example, --query=id,type,indexmemory,datamemory returns the node ID, node type, DataMemory, and IndexMemory for each node.

> **Note**
>
> If a given parameter is not applicable to a certain type of node, than an empty string is returned for the corresponding value. See the examples later in this section for more information.

- --host=*hostname*

Command-Line Format	--host=name	
Permitted Values	Type	string
	Default	

Specifies the host name of the node for which configuration information is to be obtained.

> **Note**
>
> While the hostname localhost usually resolves to the IP address 127.0.0.1, this may not necessarily be true for all operating platforms and configurations. This means that it is possible, when localhost is used in config.ini, for ndb_config --host=localhost to fail if ndb_config is run on a different host where localhost resolves to a different address

(for example, on some versions of SUSE Linux, this is 127.0.0.2). In general, for best results, you should use numeric IP addresses for all MySQL Cluster configuration values relating to hosts, or verify that all MySQL Cluster hosts handle localhost in the same fashion.

- --id=*node_id*

  --nodeid=*node_id*

Command-Line Format	--ndb-nodeid=#	
Permitted Values	Type	numeric
	Default	0

Either of these options can be used to specify the node ID of the node for which configuration information is to be obtained. --nodeid is the preferred form.

- --nodes

Command-Line Format	--nodes	
Permitted Values	Type	boolean
	Default	FALSE

Tells ndb_config to print information relating only to parameters defined in an [ndbd] or [ndbd default] section of the cluster configuration file (see Section 5.3.6, “Defining MySQL Cluster Data Nodes”).

This option is mutually exclusive with --connections and --system; only one of these 3 options can be used.

- --connections

Command-Line Format	--connections	
Permitted Values	Type	boolean
	Default	FALSE

Tells ndb_config to print CONNECTIONS information only—that is, information about parameters found in the [tcp], [tcp default], [sci], [sci default], [shm], or [shm default] sections of the cluster configuration file (see Section 5.3.9, “MySQL Cluster TCP/IP Connections”, Section 5.3.12, “SCI Transport Connections in MySQL Cluster”, and Section 5.3.11, “MySQL Cluster Shared-Memory Connections”, for more information).

This option is mutually exclusive with --nodes and --system; only one of these 3 options can be used.

- --system

Command-Line Format	--system	
Permitted Values	Type	boolean
	Default	FALSE

Tells ndb_config to print SYSTEM information only. This consists of system variables that cannot be changed at run time; thus, there is no corresponding section of the cluster configuration file for them. They can be seen (prefixed with ****** SYSTEM ******) in the output of ndb_config --configinfo.

This option is mutually exclusive with --nodes and --connections; only one of these 3 options can be used.

- --type=*node_type*

Command-Line Format	--type=name	
Permitted Values	Type	enumeration
	Default	[none]
	Valid Values	ndbd
		mysqld
		ndb_mgmd

Filters results so that only configuration values applying to nodes of the specified *node_type* (ndbd, mysqld, or ndb_mgmd) are returned.

- --fields=*delimiter*, -f *delimiter*

Command-Line Format	--fields=string	
Permitted Values	Type	string
	Default	

Specifies a *delimiter* string used to separate the fields in the result. The default is “,” (the comma character).

> **Note**
>
> If the *delimiter* contains spaces or escapes (such as \n for the linefeed character), then it must be quoted.

- --rows=*separator*, -r *separator*

Command-Line Format	--rows=string	
Permitted Values	Type	string
	Default	

Specifies a *separator* string used to separate the rows in the result. The default is a space character.

> **Note**
>
> If the *separator* contains spaces or escapes (such as \n for the linefeed character), then it must be quoted.

- --configinfo

The --configinfo option causes ndb_config to dump a list of each MySQL Cluster configuration parameter supported by the MySQL Cluster distribution of which ndb_config is a part, including the following information:

- A brief description of each parameter's purpose, effects, and usage
- The section of the config.ini file where the parameter may be used
- The parameter's data type or unit of measurement
- Where applicable, the parameter's default, minimum, and maximum values

- MySQL Cluster release version and build information

By default, this output is in text format. Part of this output is shown here:

```
shell> ndb_config --configinfo
****** SYSTEM ******
Name (String)
Name of system (NDB Cluster)
MANDATORY
PrimaryMGMNode (Non-negative Integer)
Node id of Primary ndb_mgmd(MGM) node
Default: 0 (Min: 0, Max: 4294967039)
ConfigGenerationNumber (Non-negative Integer)
Configuration generation number
Default: 0 (Min: 0, Max: 4294967039)
****** DB ******
MaxNoOfSubscriptions (Non-negative Integer)
Max no of subscriptions (default 0 == MaxNoOfTables)
Default: 0 (Min: 0, Max: 4294967039)
MaxNoOfSubscribers (Non-negative Integer)
Max no of subscribers (default 0 == 2 * MaxNoOfTables)
Default: 0 (Min: 0, Max: 4294967039)
…
```

--configinfo --xml

Command-Line Format	--configinfo --xml	
Permitted Values	Type	boolean
	Default	false

You can obtain the output of ndb_config --configinfo as XML by adding the --xml option. A portion of the resulting output is shown in this example:

```
shell> ndb_config --configinfo --xml
<configvariables protocolversion="1" ndbversionstring="5.6.27-ndb-7.3.12"
                    ndbversion="458758" ndbversionmajor="7" ndbversionminor="0"
                    ndbversionbuild="6">
  <section name="SYSTEM">
    <param name="Name" comment="Name of system (NDB Cluster)" type="string"
              mandatory="true"/>
    <param name="PrimaryMGMNode" comment="Node id of Primary ndb_mgmd(MGM) node"
              type="unsigned" default="0" min="0" max="4294967039"/>
    <param name="ConfigGenerationNumber" comment="Configuration generation number"
              type="unsigned" default="0" min="0" max="4294967039"/>
  </section>
  <section name="NDBD">
    <param name="MaxNoOfSubscriptions"
              comment="Max no of subscriptions (default 0 == MaxNoOfTables)"
              type="unsigned" default="0" min="0" max="4294967039"/>
    <param name="MaxNoOfSubscribers"
              comment="Max no of subscribers (default 0 == 2 * MaxNoOfTables)"
              type="unsigned" default="0" min="0" max="4294967039"/>
    …
  </section>
  …
</configvariables>
```

> **Note**
>
> Normally, the XML output produced by ndb_config --configinfo --xml is formatted using one line per element; we have added extra whitespace in the previous example, as well as the next one, for reasons of legibility. This should not make any difference to applications using this output, since most

> XML processors either ignore nonessential whitespace as a matter of course, or can be instructed to do so.

The XML output also indicates when changing a given parameter requires that data nodes be restarted using the `--initial` option. This is shown by the presence of an `initial="true"` attribute in the corresponding `<param>` element. In addition, the restart type (`system` or `node`) is also shown; if a given parameter requires a system restart, this is indicated by the presence of a `restart="system"` attribute in the corresponding `<param>` element. For example, changing the value set for the `Diskless` parameter requires a system initial restart, as shown here (with the `restart` and `initial` attributes highlighted for visibility):

```
<param name="Diskless" comment="Run wo/ disk" type="bool" default="false"
          restart="system" initial="true"/>
```

Currently, no `initial` attribute is included in the XML output for `<param>` elements corresponding to parameters which do not require initial restarts; in other words, `initial="false"` is the default, and the value `false` should be assumed if the attribute is not present. Similarly, the default restart type is `node` (that is, an online or “rolling” restart of the cluster), but the `restart` attribute is included only if the restart type is `system` (meaning that all cluster nodes must be shut down at the same time, then restarted).

Beginning with MySQL Cluster NDB 7.4.7, deprecated parameters are indicated in the XML output with the addition of a `deprecated` attribute, as shown here:

```
<param name="NoOfDiskPagesToDiskAfterRestartACC" comment="DiskCheckpointSpeed"
       type="unsigned" default="20" min="1" max="4294967039" deprecated="true"/>
```

In such cases, the `comment` refers to one or more parameters that supersede the deprecated parameter. Similarly to `initial`, the `deprecated` attribute is indicated only when the parameter is deprecated, with `deprecated="true"`, and does not appear at all for parameters which are not deprecated. (Bug #21127135)

> **Important**
>
> The `--xml` option can be used only with the `--configinfo` option. Using `--xml` without `--configinfo` fails with an error.

Unlike the options used with this program to obtain current configuration data, `--configinfo` and `--xml` use information obtained from the MySQL Cluster sources when `ndb_config` was compiled. For this reason, no connection to a running MySQL Cluster or access to a `config.ini` or `my.cnf` file is required for these two options.

Combining other `ndb_config` options (such as `--query` or `--type`) with `--configinfo` or `--xml` is not supported. Currently, if you attempt to do so, the usual result is that all other options besides `--configinfo` or `--xml` are simply ignored. *However, this behavior is not guaranteed and is subject to change at any time*. In addition, since `ndb_config`, when used with the `--configinfo` option, does not access the MySQL Cluster or read any files, trying to specify additional options such as `--ndb-connectstring` or `--config-file` with `--configinfo` serves no purpose.

Examples

1. To obtain the node ID and type of each node in the cluster:

```
shell> ./ndb_config --query=id,type --fields=':' --rows='\n'
1:ndbd
2:ndbd
3:ndbd
4:ndbd
```

```
5:ndb_mgmd
6:mysqld
7:mysqld
8:mysqld
9:mysqld
```

In this example, we used the `--fields` options to separate the ID and type of each node with a colon character (`:`), and the `--rows` options to place the values for each node on a new line in the output.

2. To produce a connection string that can be used by data, SQL, and API nodes to connect to the management server:

```
shell> ./ndb_config --config-file=usr/local/mysql/cluster-data/config.ini \
--query=hostname,portnumber --fields=: --rows=, --type=ndb_mgmd
192.168.0.179:1186
```

3. This invocation of `ndb_config` checks only data nodes (using the `--type` option), and shows the values for each node's ID and host name, as well as the values set for its `DataMemory`, `IndexMemory`, and `DataDir` parameters:

```
shell> ./ndb_config --type=ndbd --query=id,host,datamemory,indexmemory,datadir -f ' : ' -r '\n'
1 : 192.168.0.193 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
2 : 192.168.0.112 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
3 : 192.168.0.176 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
4 : 192.168.0.119 : 83886080 : 18874368 : /usr/local/mysql/cluster-data
```

In this example, we used the short options `-f` and `-r` for setting the field delimiter and row separator, respectively.

4. To exclude results from any host except one in particular, use the `--host` option:

```
shell> ./ndb_config --host=192.168.0.176 -f : -r '\n' -q id,type
3:ndbd
5:ndb_mgmd
```

In this example, we also used the short form `-q` to determine the attributes to be queried.

Similarly, you can limit results to a node with a specific ID using the `--id` or `--nodeid` [282] option.

6.8 ndb_cpcd — Automate Testing for NDB Development

A utility having this name was formerly part of an internal automated test framework used in testing and debugging MySQL Cluster. It is no longer included in MySQL Cluster distributions provided by Oracle.

6.9 ndb_delete_all — Delete All Rows from an NDB Table

`ndb_delete_all` deletes all rows from the given `NDB` table. In some cases, this can be much faster than `DELETE` or even `TRUNCATE TABLE`.

Usage

```
ndb_delete_all -c connection_string tbl_name -d db_name
```

This deletes all rows from the table named *tbl_name* in the database named *db_name*. It is exactly equivalent to executing `TRUNCATE db_name.tbl_name` in MySQL.

The following table includes options that are specific to `ndb_delete_all`. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including `ndb_delete_all`), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.7 This table describes command-line options for the ndb_delete_all program

Format	Description	Added or Removed
--database=dbname, -d	Name of the database in which the table is found	All MySQL 5.6 based releases
--transactional, -t	Perform the delete in a single transaction (may run out of operations)	All MySQL 5.6 based releases
--tupscan	Run tup scan	All MySQL 5.6 based releases
--diskscan	Run disk scan	All MySQL 5.6 based releases

- `--transactional`, `-t`

  Use of this option causes the delete operation to be performed as a single transaction.

  > **Warning**
  >
  > With very large tables, using this option may cause the number of operations available to the cluster to be exceeded.

6.10 ndb_desc — Describe NDB Tables

ndb_desc provides a detailed description of one or more NDB tables.

Usage

```
ndb_desc -c connection_string tbl_name -d db_name [options]

ndb_desc -c connection_string index_name -d db_name -t tbl_name
```

Additional options that can be used with ndb_desc are listed later in this section.

Sample Output

MySQL table creation and population statements:

```
USE test;
CREATE TABLE fish (
    id INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    length_mm INT(11) NOT NULL,
    weight_gm INT(11) NOT NULL,
    PRIMARY KEY pk (id),
    UNIQUE KEY uk (name)
) ENGINE=NDB;
INSERT INTO fish VALUES
    ('','guppy', 35, 2), ('','tuna', 2500, 150000),
    ('','shark', 3000, 110000), ('','manta ray', 1500, 50000),
    ('','grouper', 900, 125000), ('','puffer', 250, 2500);
```

Output from ndb_desc:

```
shell> ./ndb_desc -c localhost fish -d test -p
-- fish --
Version: 2
Fragment type: 9
```

```
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
Number of primary keys: 1
Length of frm data: 311
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 2
TableStatus: Retrieved
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
length_mm Int NOT NULL AT=FIXED ST=MEMORY
weight_gm Int NOT NULL AT=FIXED ST=MEMORY
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk$unique(name) - UniqueHashIndex
uk(name) - OrderedIndex
-- Per partition info --
Partition  Row count  Commit count  Frag fixed memory ...
0          2          2             32768             ...
1          4          4             32768             ...
... Frag varsized memory  Extent_space  Free extent_space
... 32768                 0             0
... 32768                 0             0
NDBT_ProgramExit: 0 - OK
```

Information about multiple tables can be obtained in a single invocation of `ndb_desc` by using their names, separated by spaces. All of the tables must be in the same database.

You can obtain additional information about a specific index using the `--table` (short form: `-t`) option and supplying the name of the index as the first argument to `ndb_desc`, as shown here:

```
shell> ./ndb_desc uk -d test -t fish
-- uk --
Version: 3
Base table: fish
Number of attributes: 1
Logging: 0
Index type: OrderedIndex
Index status: Retrieved
-- Attributes --
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
-- IndexTable 10/uk --
Version: 3
Fragment type: FragUndefined
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: yes
Number of attributes: 2
Number of primary keys: 1
Length of frm data: 0
Row Checksum: 1
Row GCI: 1
SingleUserMode: 2
ForceVarPart: 0
FragmentCount: 4
ExtraRowGciBits: 0
ExtraRowAuthorBits: 0
TableStatus: Retrieved
-- Attributes --
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
NDB$TNODE Unsigned [64] PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY
-- Indexes --
PRIMARY KEY(NDB$TNODE) - UniqueHashIndex
```

```
NDBT_ProgramExit: 0 - OK
```

When an index is specified in this way, the `--extra-partition-info` and `--extra-node-info` options have no effect.

The `Version` column in the output contains the table's schema object version. For information about interpreting this value, see NDB Schema Object Versions.

The `Extent_space` and `Free extent_space` columns are applicable only to `NDB` tables having columns on disk; for tables having only in-memory columns, these columns always contain the value `0`.

To illustrate their use, we modify the previous example. First, we must create the necessary Disk Data objects, as shown here:

```
CREATE LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_1.log'
    INITIAL_SIZE 16M
    UNDO_BUFFER_SIZE 2M
    ENGINE NDB;
ALTER LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_2.log'
    INITIAL_SIZE 12M
    ENGINE NDB;
CREATE TABLESPACE ts_1
    ADD DATAFILE 'data_1.dat'
    USE LOGFILE GROUP lg_1
    INITIAL_SIZE 32M
    ENGINE NDB;
ALTER TABLESPACE ts_1
    ADD DATAFILE 'data_2.dat'
    INITIAL_SIZE 48M
    ENGINE NDB;
```

(For more information on the statements just shown and the objects created by them, see Section 7.12.1, “MySQL Cluster Disk Data Objects”, as well as CREATE LOGFILE GROUP Syntax, and CREATE TABLESPACE Syntax.)

Now we can create and populate a version of the `fish` table that stores 2 of its columns on disk (deleting the previous version of the table first, if it already exists):

```
CREATE TABLE fish (
    id INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    length_mm INT(11) NOT NULL,
    weight_gm INT(11) NOT NULL,
    PRIMARY KEY pk (id),
    UNIQUE KEY uk (name)
) TABLESPACE ts_1 STORAGE DISK
ENGINE=NDB;
INSERT INTO fish VALUES
    ('','guppy', 35, 2), ('','tuna', 2500, 150000),
    ('','shark', 3000, 110000), ('','manta ray', 1500, 50000),
    ('','grouper', 900, 125000), ('','puffer', 250, 2500);
```

When run against this version of the table, `ndb_desc` displays the following output:

```
shell> ./ndb_desc -c localhost fish -d test -p
-- fish --
Version: 3
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
Number of primary keys: 1
```

```
Length of frm data: 321
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 2
TableStatus: Retrieved
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
length_mm Int NOT NULL AT=FIXED ST=DISK
weight_gm Int NOT NULL AT=FIXED ST=DISK
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk$unique(name) - UniqueHashIndex
uk(name) - OrderedIndex
-- Per partition info --
Partition  Row count  Commit count  Frag fixed memory ...
0          2          2             32768             ...
1          4          4             32768             ...
... Frag varsized memory  Extent_space  Free extent_space
... 32768                 0             0
... 32768                 0             0
NDBT_ProgramExit: 0 - OK
```

This means that 1048576 bytes are allocated from the tablespace for this table on each partition, of which 1044440 bytes remain free for additional storage. In other words, 1048576 - 1044440 = 4136 bytes per partition is currently being used to store the data from this table's disk-based columns. The number of bytes shown as `Free extent_space` is available for storing on-disk column data from the `fish` table only; for this reason, it is not visible when selecting from the `INFORMATION_SCHEMA.FILES` table.

The following table includes options that are specific to `ndb_desc`. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including `ndb_desc`), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.8 This table describes command-line options for the ndb_desc program

Format	Description	Added or Removed
`--blob-info`, `-b`	Include partition information for BLOB tables in output. Requires that the -p option also be used	All MySQL 5.6 based releases
`--database=dbname`, `-d`	Name of database containing table	All MySQL 5.6 based releases
`--extra-node-info`, `-n`	Include partition-to-data-node mappings in output. Requires that the -p option also be used	All MySQL 5.6 based releases
`--extra-partition-info`, `-p`	Display information about partitions	All MySQL 5.6 based releases
`--retries=#`, `-r`	Number of times to retry the connection (once per second)	All MySQL 5.6 based releases
`--table=tbl_name`, `-t`	Specify the table in which to find an index. When this option is used, -p and -n have no effect and are ignored.	All MySQL 5.6 based releases
`--unqualified`, `-u`	Use unqualified table names	All MySQL 5.6 based releases

- `--blob-info`, `-b`

  Include information about subordinate `BLOB` and `TEXT` columns.

  Use of this option also requires the use of the `--extra-partition-info` (`-p`) option.

- `--database=db_name`, `-d`

  Specify the database in which the table should be found.

- `--extra-node-info`, `-n`

  Include information about the mappings between table partitions and the data nodes upon which they reside. This information can be useful for verifying distribution awareness mechanisms and supporting more efficient application access to the data stored in MySQL Cluster.

  Use of this option also requires the use of the `--extra-partition-info` (`-p`) option.

- `--extra-partition-info`, `-p`

  Print additional information about the table's partitions.

- `--retries=#`, `-r`

  Try to connect this many times before giving up. One connect attempt is made per second.

- `--table=tbl_name`, `-t`

  Specify the table in which to look for an index.

- `--unqualified`, `-u`

  Use unqualified table names.

6.11 ndb_drop_index — Drop Index from an NDB Table

`ndb_drop_index` drops the specified index from an `NDB` table. *It is recommended that you use this utility only as an example for writing NDB API applications*—see the Warning later in this section for details.

Usage

```
ndb_drop_index -c connection_string table_name index -d db_name
```

The statement shown above drops the index named *`index`* from the *`table`* in the *`database`*.

The following table includes options that are specific to `ndb_drop_index`. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including `ndb_drop_index`), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.9 This table describes command-line options for the ndb_drop_index program

Format	Description	Added or Removed
`--database=dbname`, `-d`	Name of the database in which the table is found	All MySQL 5.6 based releases

> **Warning**
>
> *Operations performed on Cluster table indexes using the NDB API are not visible to MySQL and make the table unusable by a MySQL server*. If you use

this program to drop an index, then try to access the table from an SQL node, an error results, as shown here:

```
shell> ./ndb_drop_index -c localhost dogs ix -d ctest1
Dropping index dogs/idx...OK
NDBT_ProgramExit: 0 - OK
shell> ./mysql -u jon -p ctest1
Enter password: *******
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 5.6.27-ndb-7.3.12
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SHOW TABLES;
+------------------+
| Tables_in_ctest1 |
+------------------+
| a                |
| bt1              |
| bt2              |
| dogs             |
| employees        |
| fish             |
+------------------+
6 rows in set (0.00 sec)
mysql> SELECT * FROM dogs;
ERROR 1296 (HY000): Got error 4243 'Index not found' from NDBCLUSTER
```

In such a case, your *only* option for making the table available to MySQL again is to drop the table and re-create it. You can use either the SQL statement`DROP TABLE` or the `ndb_drop_table` utility (see Section 6.12, “ndb_drop_table — Drop an NDB Table”) to drop the table.

6.12 ndb_drop_table — Drop an NDB Table

`ndb_drop_table` drops the specified `NDB` table. (If you try to use this on a table created with a storage engine other than `NDB`, the attempt fails with the error `723: No such table exists`.) This operation is extremely fast; in some cases, it can be an order of magnitude faster than using a MySQL `DROP TABLE` statement on an `NDB` table.

Usage

```
ndb_drop_table -c connection_string tbl_name -d db_name
```

The following table includes options that are specific to `ndb_drop_table`. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including `ndb_drop_table`), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.10 This table describes command-line options for the ndb_drop_table program

Format	Description	Added or Removed
`--database=dbname`, `-d`	Name of the database in which the table is found	All MySQL 5.6 based releases

6.13 ndb_error_reporter — NDB Error-Reporting Utility

`ndb_error_reporter` creates an archive from data node and management node log files that can be used to help diagnose bugs or other problems with a cluster. *It is highly recommended that you make use of this utility when filing reports of bugs in MySQL Cluster.*

The following table includes command options specific to the MySQL Cluster program `ndb_error_reporter`. Additional descriptions follow the table. For options common to most MySQL

Cluster programs (including `ndb_error_reporter`), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

`ndb_error_reporter` did not support the `--help` option prior to MySQL Cluster NDB 7.3.3 (Bug #11756666, Bug #48606). The `--connection-timeout` `--dry-scp`, and `--skip-nodegroup` options were also added in this release (Bug #16602002).

Table 6.11 This table describes command-line options for the ndb_error_reporter program

Format	Description	Added or Removed
`--connection-timeout=timeout`	Number of seconds to wait when connecting to nodes before timing out.	ADDED: NDB 7.3.3
`--dry-scp`	Disable scp with remote hosts; used only for testing.	ADDED: NDB 7.3.3
`--fs`	Include file system data in error report; can use a large amount of disk space	All MySQL 5.6 based releases
`--skip-nodegroup=nodegroup_id`	Skip all nodes in the node group having this ID.	ADDED: NDB 7.3.3

Usage

```
ndb_error_reporter path/to/config-file [username] [options]
```

This utility is intended for use on a management node host, and requires the path to the management host configuration file (usually named `config.ini`). Optionally, you can supply the name of a user that is able to access the cluster's data nodes using SSH, to copy the data node log files. `ndb_error_reporter` then includes all of these files in archive that is created in the same directory in which it is run. The archive is named `ndb_error_report_YYYYMMDDHHMMSS.tar.bz2`, where *`YYYYMMDDHHMMSS`* is a datetime string.

`ndb_error_reporter` also accepts the options listed here:

- `--connection-timeout=timeout`

Introduced	5.6.14-ndb-7.3.3		
Command-Line Format	`--connection-timeout=timeout`		
Permitted Values	**Type**	`integer`	
	Default	`0`	

  Wait this many seconds when trying to connect to nodes before timing out.

- `--dry-scp`

Introduced	5.6.14-ndb-7.3.3		
Command-Line Format	`--dry-scp`		
Permitted Values	**Type**	`boolean`	
	Default	`TRUE`	

  Run `ndb_error_reporter` without using scp from remote hosts. Used for testing only.

- `--fs`

Command-Line Format	`--fs`	
Permitted Values	**Type**	`boolean`

	Default	FALSE

Copy the data node file systems to the management host and include them in the archive.

Because data node file systems can be extremely large, even after being compressed, we ask that you please do *not* send archives created using this option to Oracle unless you are specifically requested to do so.

- `--skip-nodegroup=`*`nodegroup_id`*

Introduced	5.6.14-ndb-7.3.3	
Command-Line Format	--connection-timeout=timeout	
Permitted Values	**Type**	integer
	Default	0

Skip all nodes belong to the node group having the supplied node group ID.

6.14 ndb_index_stat — NDB Index Statistics Utility

`ndb_index_stat` provides per-fragment statistical information about indexes on `NDB` tables. This includes cache version and age, number of index entries per partition, and memory consumption by indexes.

Usage

To obtain basic index statistics about a given `NDB` table, invoke `ndb_index_stat` as shown here, with the name of the table as the first argument and the name of the database containing this table specified immediately following it, using the `--database` (`-d`) option:

```
ndb_index_stat table -d database
```

In this example, we use `ndb_index_stat` to obtain such information about an `NDB` table named `mytable` in the `test` database:

```
shell> ndb_index_stat -d test mytable
table:City index:PRIMARY fragCount:2
sampleVersion:3 loadTime:1399585986 sampleCount:1994 keyBytes:7976
query cache: valid:1 sampleCount:1994 totalBytes:27916
times in ms: save: 7.133 sort: 1.974 sort per sample: 0.000
NDBT_ProgramExit: 0 - OK
```

`sampleVersion` is the version number of the cache from which the statistics data is taken. Running `ndb_index_stat` with the `--update` option causes sampleVersion to be incremented.

`loadTime` shows when the cache was last updated. This is expressed as seconds since the Unix Epoch.

`sampleCount` is the number of index entries found per partition. You can estimate the total number of entries by multiplying this by the number of fragments (shown as `fragCount`).

`sampleCount` can be compared with the cardinality of `SHOW INDEX` or `INFORMATION_SCHEMA.STATISTICS`, although the latter two provide a view of the table as a whole, while `ndb_index_stat` provides a per-fragment average.

`keyBytes` is the number of bytes used by the index. In this example, the primary key is an integer, which requires four bytes for each index, so `keyBytes` can be calculated in this case as shown here:

```
    keyBytes = sampleCount * (4 bytes per index) = 1994 * 4 = 7976
```

This information can also be obtained using the corresponding column definitions from INFORMATION_SCHEMA.COLUMNS (this requires a MySQL Server and a MySQL client application).

totalBytes is the total memory consumed by all indexes on the table, in bytes.

Timings shown in the preceding examples are specific to each invocation of ndb_index_stat.

The --verbose option provides some additional output, as shown here:

```
shell> ndb_index_stat -d test mytable --verbose
random seed 1337010518
connected
loop 1 of 1
table:mytable index:PRIMARY fragCount:4
sampleVersion:2 loadTime:1336751773 sampleCount:0 keyBytes:0
read stats
query cache created
query cache: valid:1 sampleCount:0 totalBytes:0
times in ms: save: 20.766 sort: 0.001
disconnected
NDBT_ProgramExit: 0 - OK
shell>
```

Options

The following table includes options that are specific to the MySQL Cluster ndb_index_stat utility. Additional descriptions are listed following the table. For options common to most MySQL Cluster programs (including ndb_index_stat), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.12 This table describes command-line options for the ndb_index_stat program

Format	Description	Added or Removed
--database=name, -d	Name of the database containing the table.	All MySQL 5.6 based releases
--delete	Delete index statistics for the given table, stopping any auto-update previously configured.	All MySQL 5.6 based releases
--update	Update index statistics for the given table, restarting any auto-update previously configured.	All MySQL 5.6 based releases
--dump	Print the query cache.	All MySQL 5.6 based releases
--query=#	Perform a number of random range queries on first key attr (must be int unsigned).	All MySQL 5.6 based releases
--sys-drop	Drop any statistics tables and events in NDB kernel (all statistics are lost)	All MySQL 5.6 based releases
--sys-create	Create all statistics tables and events in NDB kernel, if none of them already exist	All MySQL 5.6 based releases
--sys-create-if-not-exist	Create any statistics tables and events in NDB kernel that do not already exist.	All MySQL 5.6 based releases
--sys-create-if-not-valid	Create any statistics tables or events that do not already exist in the NDB kernel. after dropping any that are invalid.	All MySQL 5.6 based releases

Format	Description	Added or Removed
--sys-check	Verify that NDB system index statistics and event tables exist.	All MySQL 5.6 based releases
--sys-skip-tables	Do not apply sys-* options to tables.	All MySQL 5.6 based releases
--sys-skip-events	Do not apply sys-* options to events.	All MySQL 5.6 based releases
--verbose, -v	Turn on verbose output	All MySQL 5.6 based releases
--loops=#	Set the number of times to perform a given command. Default is 0.	All MySQL 5.6 based releases

ndb_index_stat statistics options. The following options are used to generate index statistics. They work with a given table and database. They cannot be mixed with system options (see ndb_index_stat system options).

- `--database=name`, `-d name`

Command-Line Format	--database=name	
Permitted Values	Type	string
	Default	[none]
	Min Value	
	Max Value	

  The name of the database that contains the table being queried.

- `--delete`

Command-Line Format	--delete	
Permitted Values	Type	boolean
	Default	false
	Min Value	
	Max Value	

  Delete the index statistics for the given table, stopping any auto-update that was previously configured.

- `--update`

Command-Line Format	--update	
Permitted Values	Type	boolean
	Default	false
	Min Value	
	Max Value	

Update the index statistics for the given table, and restart any auto-update that was previously configured.

- `--dump`

Command-Line Format	`--dump`	
Permitted Values	**Type**	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Dump the contents of the query cache.

- `--query=#`

Command-Line Format	`--query=#`	
Permitted Values	**Type**	`numeric`
	Default	`0`
	Min Value	`0`
	Max Value	`MAX_INT`

Perform random range queries on first key attribute (must be int unsigned).

ndb_index_stat system options. The following options are used to generate and update the statistics tables in the NDB kernel. None of these options can be mixed with statistics options (see ndb_index_stat statistics options).

- `--sys-drop`

Command-Line Format	`--sys-drop`	
Permitted Values	**Type**	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Drop all statistics tables and events in the NDB kernel. *This causes all statistics to be lost*.

- `--sys-create`

Command-Line Format	`--sys-create`	
Permitted Values	**Type**	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Create all statistics tables and events in the NDB kernel. This works only if none of them exist previously.

- `sys-create-if-not-exist`

Command-Line Format	`--sys-create-if-not-exist`	
Permitted Values	Type	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Create any NDB system statistics tables or events (or both) that do not already exist when the program is invoked.

- `--sys-create-if-not-valid`

Command-Line Format	`--sys-create-if-not-valid`	
Permitted Values	Type	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Create any NDB system statistics tables or events that do not already exist, after dropping any that are invalid.

- `--sys-check`

Command-Line Format	`--sys-check`	
Permitted Values	Type	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Verify that all required system statistics tables and events exist in the NDB kernel.

- `--sys-skip-tables`

Command-Line Format	`--sys-skip-tables`	
Permitted Values	Type	`boolean`
	Default	`false`
	Min Value	
	Max Value	

Do not apply any `--sys-*` options to any statistics tables.

- --sys-skip-events

Command-Line Format	--sys-skip-events	
Permitted Values	Type	boolean
	Default	false
	Min Value	
	Max Value	

Do not apply any --sys-* options to any events.

- --verbose

Command-Line Format	--verbose	
Permitted Values	Type	boolean
	Default	false
	Min Value	
	Max Value	

Turn on verbose output.

- --loops=#

Command-Line Format	--loops=#	
Permitted Values	Type	numeric
	Default	0
	Min Value	0
	Max Value	MAX_INT

Repeat commands this number of times (for use in testing).

6.15 ndb_print_backup_file — Print NDB Backup File Contents

ndb_print_backup_file obtains diagnostic information from a cluster backup file.

Usage

```
ndb_print_backup_file file_name
```

file_name is the name of a cluster backup file. This can be any of the files (.Data, .ctl, or .log file) found in a cluster backup directory. These files are found in the data node's backup directory under the subdirectory BACKUP-#, where # is the sequence number for the backup. For more information about cluster backup files and their contents, see Section 7.3.1, “MySQL Cluster Backup Concepts”.

Like ndb_print_schema_file and ndb_print_sys_file (and unlike most of the other NDB utilities that are intended to be run on a management server host or to connect to a management server) ndb_print_backup_file must be run on a cluster data node, since it accesses the data node file system directly. Because it does not make use of the management server, this utility can be used when the management server is not running, and even when the cluster has been completely shut down.

Additional Options

None.

6.16 ndb_print_file — Print NDB Disk Data File Contents

ndb_print_file obtains information from a MySQL Cluster Disk Data file.

Usage

```
ndb_print_file [-v] [-q] file_name+
```

file_name is the name of a MySQL Cluster Disk Data file. Multiple filenames are accepted, separated by spaces.

Like ndb_print_schema_file and ndb_print_sys_file (and unlike most of the other NDB utilities that are intended to be run on a management server host or to connect to a management server) ndb_print_file must be run on a MySQL Cluster data node, since it accesses the data node file system directly. Because it does not make use of the management server, this utility can be used when the management server is not running, and even when the cluster has been completely shut down.

Additional Options

ndb_print_file supports the following options:

- -v: Make output verbose.
- -q: Suppress output (quiet mode).
- --help, -h, -?: Print help message.

  This option did not work correctly prior to MySQL Cluster NDB 7.3.7. (Bug #17069285)

For more information, see Section 7.12, "MySQL Cluster Disk Data Tables".

6.17 ndb_print_schema_file — Print NDB Schema File Contents

ndb_print_schema_file obtains diagnostic information from a cluster schema file.

Usage

```
ndb_print_schema_file file_name
```

file_name is the name of a cluster schema file. For more information about cluster schema files, see MySQL Cluster Data Node File System Directory Files.

Like ndb_print_backup_file and ndb_print_sys_file (and unlike most of the other NDB utilities that are intended to be run on a management server host or to connect to a management server) ndb_schema_backup_file must be run on a cluster data node, since it accesses the data node file system directly. Because it does not make use of the management server, this utility can be used when the management server is not running, and even when the cluster has been completely shut down.

Additional Options

None.

6.18 ndb_print_sys_file — Print NDB System File Contents

ndb_print_sys_file obtains diagnostic information from a MySQL Cluster system file.

Usage

```
ndb_print_sys_file file_name
```

file_name is the name of a cluster system file (sysfile). Cluster system files are located in a data node's data directory (DataDir); the path under this directory to system files matches the pattern ndb_#_fs/D#/DBDIH/P#.sysfile. In each case, the # represents a number (not necessarily the same number). For more information, see MySQL Cluster Data Node File System Directory Files.

Like ndb_print_backup_file and ndb_print_schema_file (and unlike most of the other NDB utilities that are intended to be run on a management server host or to connect to a management server) ndb_print_backup_file must be run on a cluster data node, since it accesses the data node file system directly. Because it does not make use of the management server, this utility can be used when the management server is not running, and even when the cluster has been completely shut down.

Additional Options

None.

6.19 ndbd_redo_log_reader — Check and Print Content of Cluster Redo Log

Reads a redo log file, checking it for errors, printing its contents in a human-readable format, or both. ndbd_redo_log_reader is intended for use primarily by MySQL Cluster developers and Support personnel in debugging and diagnosing problems.

This utility remains under development, and its syntax and behavior are subject to change in future MySQL Cluster releases.

The C++ source files for ndbd_redo_log_reader can be found in the directory /storage/ndb/src/kernel/blocks/dblqh/redoLogReader.

The following table includes options that are specific to the MySQL Cluster program ndbd_redo_log_reader. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndbd_redo_log_reader), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.13 This table describes command-line options for the ndbd_redo_log_reader program

Format	Description	Added or Removed
-noprint [302]	Do not print records	All MySQL 5.6 based releases
-nocheck [302]	Do not check records for errors	All MySQL 5.6 based releases
--help [302]	Print usage information	ADDED: NDB 7.3.4

Usage

```
ndbd_redo_log_reader file_name [options]
```

file_name is the name of a cluster redo log file. redo log files are located in the numbered directories under the data node's data directory (DataDir); the path under this directory to the redo log files matches the pattern ndb_#_fs/D#/LCP/#/T#F#.Data. In each case, the # represents a number (not necessarily the same number). For more information, see MySQL Cluster Data Node File System Directory Files.

The name of the file to be read may be followed by one or more of the options listed here:

Command-Line Format	-noprint	
Permitted Values	**Type**	boolean
	Default	FALSE

  -noprint: Do not print the contents of the log file.

Command-Line Format	-nocheck	
Permitted Values	**Type**	boolean
	Default	FALSE

  -nocheck: Do not check the log file for errors.

Introduced	5.6.15-ndb-7.3.4
Command-Line Format	--help

  --help: Print usage information.

  Added in MySQL Cluster NDB 7.3.4. (Bug #11749591, Bug #36805)

Like ndb_print_backup_file and ndb_print_schema_file (and unlike most of the NDB utilities that are intended to be run on a management server host or to connect to a management server) ndbd_redo_log_reader must be run on a cluster data node, since it accesses the data node file system directly. Because it does not make use of the management server, this utility can be used when the management server is not running, and even when the cluster has been completely shut down.

6.20 ndb_restore — Restore a MySQL Cluster Backup

The cluster restoration program is implemented as a separate command-line utility ndb_restore, which can normally be found in the MySQL bin directory. This program reads the files created as a result of the backup and inserts the stored information into the database.

ndb_restore must be executed once for each of the backup files that were created by the START BACKUP command used to create the backup (see Section 7.3.2, “Using The MySQL Cluster Management Client to Create a Backup”). This is equal to the number of data nodes in the cluster at the time that the backup was created.

> **Note**
>
> Before using ndb_restore, it is recommended that the cluster be running in single user mode, unless you are restoring multiple data nodes in parallel. See Section 7.8, “MySQL Cluster Single User Mode”, for more information.

The following table includes options that are specific to the MySQL Cluster native backup restoration program ndb_restore. Additional descriptions follow the table. For options common to most MySQL

Cluster programs (including ndb_restore), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.14 This table describes command-line options for the ndb_restore program

Format	Description	Added or Removed
--connect [306], -c [306]	Alias for --connectstring.	All MySQL 5.6 based releases
--nodeid=# [306], -n [306]	Restore backup files to node with this ID	All MySQL 5.6 based releases
--backupid=# [307], -b [307]	Restore from the backup with the given ID	All MySQL 5.6 based releases
--restore_data [308], -r [308]	Restore table data and logs into NDB Cluster using the NDB API	All MySQL 5.6 based releases
--restore_meta [308], -m [308]	Restore metadata to NDB Cluster using the NDB API	All MySQL 5.6 based releases
--no-upgrade [308], -u [308]	Do not upgrade array type for varsize attributes which do not already resize VAR data, and do not change column attributes	All MySQL 5.6 based releases
--promote-attributes [306], -A [306]	Allow attributes to be promoted when restoring data from backup	All MySQL 5.6 based releases
--preserve-trailing-spaces [307], -P [307]	Allow preservation of trailing spaces (including padding) when promoting fixed-width string types to variable-width types	All MySQL 5.6 based releases
--no-restore-disk-objects [310], -d [310]	Do not restore objects relating to Disk Data	All MySQL 5.6 based releases
--restore_epoch [307], -e [307]	Restore epoch info into the status table. Convenient on a MySQL Cluster replication slave for starting replication. The row in mysql.ndb_apply_status with id 0 will be updated/inserted.	All MySQL 5.6 based releases
--skip-table-check [306], -s [306]	Skip table structure check during restoring of data	All MySQL 5.6 based releases
--parallelism=# [310], -p [310]	Number of parallel transactions to use while restoring data	All MySQL 5.6 based releases
--print [310]	Print metadata, data and log to stdout (equivalent to --print_meta --print_data --print_log)	All MySQL 5.6 based releases
--print_meta [310]	Print metadata to stdout	All MySQL 5.6 based releases

Format	Description	Added or Removed
--print_data [308]	Print data to stdout	All MySQL 5.6 based releases
--print_log [310]	Print to stdout	All MySQL 5.6 based releases
--backup_path=dir_name [308]	Path to backup files directory	All MySQL 5.6 based releases
--dont_ignore_systab_0 [310], -f [310]	Do not ignore system table during restore. Experimental only; not for production use	All MySQL 5.6 based releases
--ndb-nodegroup-map=map [310], -z [310]	Nodegroup map for NDBCLUSTER storage engine. Syntax: list of (source_nodegroup, destination_nodegroup)	All MySQL 5.6 based releases
--fields-enclosed-by=char	Fields are enclosed with the indicated character	All MySQL 5.6 based releases
--fields-terminated-by=char	Fields are terminated by the indicated character	All MySQL 5.6 based releases
--fields-optionally-enclosed-by	Fields are optionally enclosed with the indicated character	All MySQL 5.6 based releases
--lines-terminated-by=char	Lines are terminated by the indicated character	All MySQL 5.6 based releases
--hex	Print binary types in hexadecimal format	All MySQL 5.6 based releases
--tab=dir_name, -T dir_name	Creates a tab-separated .txt file for each table in the given path	All MySQL 5.6 based releases
--append	Append data to a tab-delimited file	All MySQL 5.6 based releases
--progress-frequency=# [311]	Print status of restoration each given number of seconds	All MySQL 5.6 based releases
--no-binlog [310]	If a mysqld is connected and using binary logging, do not log the restored data	All MySQL 5.6 based releases
--verbose=# [311]	Level of verbosity in output	All MySQL 5.6 based releases
--include-databases=db-list [311]	List of one or more databases to restore (excludes those not named)	All MySQL 5.6 based releases
--exclude-databases=db-list [312]	List of one or more databases to exclude (includes those not named)	All MySQL 5.6 based releases
--include-tables=table-list [311]	List of one or more tables to restore (excludes those in same database that are not named); each table reference must include the database name	All MySQL 5.6 based releases
--exclude-tables=table-list [312]	List of one or more tables to exclude (includes those in the same database that are not named); each table reference must include the database name	All MySQL 5.6 based releases

Format	Description	Added or Removed
--exclude-missing-columns [313]	Causes columns from the backup version of a table that are missing from the version of the table in the database to be ignored.	All MySQL 5.6 based releases
--exclude-missing-tables [313]	Causes tables from the backup that are missing from the database to be ignored.	ADDED: NDB 7.3.7
--disable-indexes [313]	Causes indexes from a backup to be ignored; may decrease time needed to restore data.	All MySQL 5.6 based releases
--rebuild-indexes [313]	Causes multi-threaded rebuilding of ordered indexes found in the backup. Number of threads used is determined by setting BuildIndexThreads parameter.	All MySQL 5.6 based releases
--skip-broken-objects [314]	Causes missing blob tables in the backup file to be ignored.	All MySQL 5.6 based releases
--skip-unknown-objects [314]	Causes schema objects not recognized by ndb_restore to be ignored when restoring a backup made from a newer MySQL Cluster version to an older version.	All MySQL 5.6 based releases
--rewrite-database=olddb,newdb [314]	Restores to a database with a different name than the original	All MySQL 5.6 based releases
--lossy-conversions [307], -L [307]	Allow lossy conversions of column values (type demotions or changes in sign) when restoring data from backup	All MySQL 5.6 based releases
--restore-privilege-tables [308]	Restore MySQL privilege tables that were previously moved to NDB.	All MySQL 5.6 based releases
--exclude-intermediate-sql-tables[=TRUE|FALSE] [314]	If TRUE (the default), do not restore any intermediate tables (having names prefixed with '#sql-') that were left over from copying ALTER TABLE operations.	ADDED: NDB 7.3.6

Typical options for this utility are shown here:

Normally, when restoring from a MySQL Cluster backup, ndb_restore requires at a minimum the --nodeid [306] (short form: -n), --backupid [307] (short form: -b), and --backup_path [308] options. In addition, beginning with MySQL Cluster NDB 7.3.11 and MySQL Cluster NDB 7.4.8, when ndb_restore is used to restore any tables containing unique indexes, you must include --disable-indexes [313] or --rebuild-indexes [313]. (Bug #57782, Bug #11764893)

```
ndb_restore [-c connection_string] -n node_id -b backup_id \
      [-m] -r --backup_path=/path/to/backup/files
```

The -c option is used to specify a connection string which tells ndb_restore where to locate the cluster management server. (See Section 5.3.3, “MySQL Cluster Connection Strings”, for information on connection strings.) If this option is not used, then ndb_restore attempts to connect to a management server on localhost:1186. This utility acts as a cluster API node, and so requires a free connection “slot” to connect to the cluster management server. This means that there must be at least one [api] or [mysqld] section that can be used by it in the cluster config.ini file. It is a good idea to keep at least one empty [api] or [mysqld] section in config.ini that is not being used for a MySQL server or other application for this reason (see Section 5.3.7, “Defining SQL and Other API Nodes in a MySQL Cluster”).

You can verify that ndb_restore is connected to the cluster by using the SHOW command in the ndb_mgm management client. You can also accomplish this from a system shell, as shown here:

```
shell> ndb_mgm -e "SHOW"
```

The --nodeid or -n is used to specify the node ID of the data node on which the backup should be restored.

The first time you run the ndb_restore restoration program, you also need to restore the metadata. In other words, you must re-create the database tables—this can be done by running it with the --restore_meta (-m) option. Restoring the metdata need be done only on a single data node; this is sufficient to restore it to the entire cluster. Note that the cluster should have an empty database when starting to restore a backup. (In other words, you should start ndbd with --initial prior to performing the restore.)

It is possible to restore data without restoring table metadata. The default behavior when doing this is for ndb_restore to fail with an error if table data do not match the table schema; this can be overridden using the --skip-table-check or -s option.

Some of the restrictions on mismatches in column definitions when restoring data using ndb_restore are relaxed; when one of these types of mismatches is encountered, ndb_restore does not stop with an error as it did previously, but rather accepts the data and inserts it into the target table while issuing a warning to the user that this is being done. This behavior occurs whether or not either of the options --skip-table-check or --promote-attributes [306] is in use. These differences in column definitions are of the following types:

- Different COLUMN_FORMAT settings (FIXED, DYNAMIC, DEFAULT)
- Different STORAGE settings (MEMORY, DISK)
- Different default values
- Different distribution key settings

ndb_restore supports limited *attribute promotion* in much the same way that it is supported by MySQL replication; that is, data backed up from a column of a given type can generally be restored to a column using a “larger, similar” type. For example, data from a CHAR(20) column can be restored to a column declared as VARCHAR(20), VARCHAR(30), or CHAR(30); data from a MEDIUMINT column can be restored to a column of type INT or BIGINT. See Replication of Columns Having Different Data Types, for a table of type conversions currently supported by attribute promotion.

Attribute promotion by ndb_restore must be enabled explicitly, as follows:

1. Prepare the table to which the backup is to be restored. ndb_restore cannot be used to re-create the table with a different definition from the original; this means that you must either create the table manually, or alter the columns which you wish to promote using ALTER TABLE after restoring the table metadata but before restoring the data.

2. Invoke ndb_restore with the --promote-attributes [306] option (short form -A) when restoring the table data. Attribute promotion does not occur if this option is not used; instead, the restore operation fails with an error.

Prior to MySQL Cluster NDB 7.3.3, conversions between character data types and TEXT or BLOB were not handled correctly (Bug #17325051).

Prior to MySQL Cluster NDB 7.3.7, demotion of TEXT to TINYTEXT was not handled correctly (Bug #18875137).

When converting between character data types and TEXT or BLOB, only conversions between character types (CHAR and VARCHAR) and binary types (BINARY and VARBINARY) can be performed at the same time. For example, you cannot promote an INT column to BIGINT while promoting a VARCHAR column to TEXT in the same invocation of ndb_restore.

Converting between TEXT columns using different character sets is not supported. Beginning with MySQL Cluster NDB 7.3.7, it is expressly disallowed (Bug #18875137).

When performing conversions of character or binary types to TEXT or BLOB with ndb_restore, you may notice that it creates and uses one or more staging tables named *table_name*$ST*node_id*. These tables are not needed afterwards, and are normally deleted by ndb_restore following a successful restoration.

--lossy-conversions, -L

Command-Line Format	--lossy-conversions	
Permitted Values	Type	boolean
	Default	FALSE

This option is intended to complement the --promote-attributes [306] option. Using --lossy-conversions allows lossy conversions of column values (type demotions or changes in sign) when restoring data from backup. With some exceptions, the rules governing demotion are the same as for MySQL replication; see Replication of Columns Having Different Data Types, for information about specific type conversions currently supported by attribute demotion.

ndb_restore reports any truncation of data that it performs during lossy conversions once per attribute and column.

The --preserve-trailing-spaces option (short form -R) causes trailing spaces to be preserved when promoting a fixed-width character data type to its variable-width equivalent—that is, when promoting a CHAR column value to VARCHAR or a BINARY column value to VARBINARY. Otherwise, any trailing spaces are dropped from such column values when they are inserted into the new columns.

Note

Although you can promote CHAR columns to VARCHAR and BINARY columns to VARBINARY, you cannot promote VARCHAR columns to CHAR or VARBINARY columns to BINARY.

The -b option is used to specify the ID or sequence number of the backup, and is the same number shown by the management client in the Backup *backup_id* completed message displayed upon completion of a backup. (See Section 7.3.2, “Using The MySQL Cluster Management Client to Create a Backup”.)

Important

When restoring cluster backups, you must be sure to restore all data nodes from backups having the same backup ID. Using files from different backups will at best result in restoring the cluster to an inconsistent state, and may fail altogether.

--restore_epoch (short form: -e) adds (or restores) epoch information to the cluster replication status table. This is useful for starting replication on a MySQL Cluster replication slave. When this option is used, the row in the mysql.ndb_apply_status having 0 in the id column is updated if it already exists; such a row is inserted if it does not already exist. (See Section 8.9, “MySQL Cluster Backups With MySQL Cluster Replication”.)

`--restore_data`

This option causes ndb_restore to output NDB table data and logs.

`--restore_meta`

This option causes ndb_restore to print NDB table metadata. Generally, you need only use this option when restoring the first data node of a cluster; additional data nodes can obtain the metadata from the first one.

`--restore-privilege-tables`

ndb_restore does not by default restore distributed MySQL privilege tables. This option causes ndb_restore to restore the privilege tables.

This works only if the privilege tables were converted to NDB before the backup was taken. For more information, see Section 7.14, “Distributed MySQL Privileges for MySQL Cluster”.

`--backup_path`

The path to the backup directory is required; this is supplied to ndb_restore using the --backup_path option, and must include the subdirectory corresponding to the ID backup of the backup to be restored. For example, if the data node's DataDir is /var/lib/mysql-cluster, then the backup directory is /var/lib/mysql-cluster/BACKUP, and the backup files for the backup with the ID 3 can be found in /var/lib/mysql-cluster/BACKUP/BACKUP-3. The path may be absolute or relative to the directory in which the ndb_restore executable is located, and may be optionally prefixed with backup_path=.

It is possible to restore a backup to a database with a different configuration than it was created from. For example, suppose that a backup with backup ID 12, created in a cluster with two database nodes having the node IDs 2 and 3, is to be restored to a cluster with four nodes. Then ndb_restore must be run twice—once for each database node in the cluster where the backup was taken. However, ndb_restore cannot always restore backups made from a cluster running one version of MySQL to a cluster running a different MySQL version. See Section 4.8, “Upgrading and Downgrading MySQL Cluster”, for more information.

> **Important**
>
> It is not possible to restore a backup made from a newer version of MySQL Cluster using an older version of ndb_restore. You can restore a backup made from a newer version of MySQL to an older cluster, but you must use a copy of ndb_restore from the newer MySQL Cluster version to do so.
>
> For example, to restore a cluster backup taken from a cluster running MySQL Cluster NDB 7.4.5 to a cluster running MySQL Cluster NDB 7.3.8, you must use the ndb_restore that comes with the MySQL Cluster NDB 7.4.5 distribution.

For more rapid restoration, the data may be restored in parallel, provided that there is a sufficient number of cluster connections available. That is, when restoring to multiple nodes in parallel, you must have an [api] or [mysqld] section in the cluster config.ini file available for each concurrent ndb_restore process. However, the data files must always be applied before the logs.

`--no-upgrade`

When using ndb_restore to restore a backup, VARCHAR columns created using the old fixed format are resized and recreated using the variable-width format now employed. This behavior can be overridden using the --no-upgrade [308] option (short form: -u) when running ndb_restore.

`--print_data`

The `--print_data` option causes `ndb_restore` to direct its output to `stdout`.

`TEXT` and `BLOB` column values are always truncated. In MySQL Cluster NDB 7.3.7 and earlier, such values are truncated to the first 240 bytes in the output; in MySQL Cluster NDB 7.3.8 and later, they are truncated to 256 bytes. (Bug #14571512, Bug #65467) This cannot currently be overridden when using `--print_data`.

Several additional options are available for use with the `--print_data` option in generating data dumps, either to `stdout`, or to a file. These are similar to some of the options used with `mysqldump`, and are shown in the following list:

- `--tab`, `-T`

Command-Line Format	`--tab=dir_name`	
Permitted Values	**Type**	`directory name`

  This option causes `--print_data` [308] to create dump files, one per table, each named *`tbl_name`*`.txt`. It requires as its argument the path to the directory where the files should be saved; use `.` for the current directory.

- `--fields-enclosed-by=`*`string`*

Command-Line Format	`--fields-enclosed-by=char`	
Permitted Values	**Type**	`string`
	Default	

  Each column values are enclosed by the string passed to this option (regardless of data type; see next item).

- `--fields-optionally-enclosed-by=`*`string`*

Command-Line Format	`--fields-optionally-enclosed-by`	
Permitted Values	**Type**	`string`
	Default	

  The string passed to this option is used to enclose column values containing character data (such as `CHAR`, `VARCHAR`, `BINARY`, `TEXT`, or `ENUM`).

- `--fields-terminated-by=`*`string`*

Command-Line Format	`--fields-terminated-by=char`	
Permitted Values	**Type**	`string`
	Default	`\t (tab)`

  The string passed to this option is used to separate column values. The default value is a tab character (`\t`).

- `--hex`

Command-Line Format	`--hex`

  If this option is used, all binary values are output in hexadecimal format.

- `--fields-terminated-by=`*`string`*

Command-Line Format	`--fields-terminated-by=char`	
Permitted Values	**Type**	`string`

	Default	\t (tab)

This option specifies the string used to end each line of output. The default is a linefeed character (\n).

- --append

Command-Line Format	--append

When used with the --tab and --print_data [308] options, this causes the data to be appended to any existing files having the same names.

> **Note**
>
> If a table has no explicit primary key, then the output generated when using the --print_data [308] option includes the table's hidden primary key.

--print_meta

This option causes ndb_restore to print all metadata to stdout.

--print_log

The --print_log option causes ndb_restore to output its log to stdout.

--print

Causes ndb_restore to print all data, metadata, and logs to stdout. Equivalent to using the --print_data [308], --print_meta [310], and --print_log [310] options together.

> **Note**
>
> Use of --print or any of the --print_* options is in effect performing a dry run. Including one or more of these options causes any output to be redirected to stdout; in such cases, ndb_restore makes no attempt to restore data or metadata to a MySQL Cluster.

--dont_ignore_systab_0

Normally, when restoring table data and metadata, ndb_restore ignores the copy of the NDB system table that is present in the backup. --dont_ignore_systab_0 causes the system table to be restored. *This option is intended for experimental and development use only, and is not recommended in a production environment*.

--ndb-nodegroup-map, -z

This option can be used to restore a backup taken from one node group to a different node group. Its argument is a list of the form *source_node_group, target_node_group*.

--no-binlog

This option prevents any connected SQL nodes from writing data restored by ndb_restore to their binary logs.

--no-restore-disk-objects, -d

This option stops ndb_restore from restoring any MySQL Cluster Disk Data objects, such as tablespaces and log file groups; see Section 7.12, “MySQL Cluster Disk Data Tables”, for more information about these.

--parallelism=#, -p

Determines the maximum number of parallel transactions that ndb_restore tries to use. By default, this is 128; the minimum is 1, and the maximum is 1024.

`--progress-frequency=N`

Print a status report each *N* seconds while the backup is in progress. 0 (the default) causes no status reports to be printed. The maximum is 65535.

`--verbose=#`

Sets the level for the verbosity of the output. The minimum is 0; the maximum is 255. The default value is 1.

It is possible to restore only selected databases, or selected tables from a single database, using the syntax shown here:

```
ndb_restore other_options db_name,[db_name[,...] | tbl_name[,tbl_name][,...]]
```

In other words, you can specify either of the following to be restored:

- All tables from one or more databases
- One or more tables from a single database

`--include-databases=db_name[,db_name][,...]`

Command-Line Format	`--include-databases=db-list`	
Permitted Values	**Type**	`string`
	Default	

`--include-tables=db_name.tbl_name[,db_name.tbl_name][,...]`

Command-Line Format	`--include-tables=table-list`	
Permitted Values	**Type**	`string`
	Default	

Use the `--include-databases` [311] option or the `--include-tables` option for restoring only specific databases or tables, respectively. `--include-databases` takes a comma-delimited list of databases to be restored. `--include-tables` takes a comma-delimited list of tables (in *`database.table`* format) to be restored.

When `--include-databases` or `--include-tables` is used, only those databases or tables named by the option are restored; all other databases and tables are excluded by `ndb_restore`, and are not restored.

The following table shows several invocations of `ndb_restore` using `--include-*` options (other options possibly required have been omitted for clarity), and the effects these have on restoring from a MySQL Cluster backup:

Option Used	Result
`--include-databases=db1`	Only tables in database `db1` are restored; all tables in all other databases are ignored
`--include-databases=db1,db2` (or `--include-databases=db1 --include-databases=db2`)	Only tables in databases `db1` and `db2` are restored; all tables in all other databases are ignored
`--include-tables=db1.t1`	Only table `t1` in database `db1` is restored; no other tables in `db1` or in any other database are restored
`--include-tables=db1.t2,db2.t1` (or `--include-tables=db1.t2 --include-tables=db2.t1`)	Only the table `t2` in database `db1` and the table `t1` in database `db2` are restored; no other tables in `db1`, `db2`, or any other database are restored

You can also use these two options together. For example, the following causes all tables in databases db1 and db2, together with the tables t1 and t2 in database db3, to be restored (and no other databases or tables):

```
shell> ndb_restore [...] --include-databases=db1,db2 --include-tables=db3.t1,db3.t2
```

(Again we have omitted other, possibly required, options in the example just shown.)

--exclude-databases=*db_name*[,*db_name*][,...]

Command-Line Format	--exclude-databases=db-list	
Permitted Values	Type	string
	Default	

--exclude-tables=*db_name.tbl_name*[,*db_name.tbl_name*][,...]

Command-Line Format	--exclude-tables=table-list	
Permitted Values	Type	string
	Default	

It is possible to prevent one or more databases or tables from being restored using the ndb_restore options --exclude-databases and --exclude-tables. --exclude-databases takes a comma-delimited list of one or more databases which should not be restored. --exclude-tables takes a comma-delimited list of one or more tables (using *database.table* format) which should not be restored.

When --exclude-databases or --exclude-tables is used, only those databases or tables named by the option are excluded; all other databases and tables are restored by ndb_restore.

This table shows several invocations of ndb_restore usng --exclude-* options (other options possibly required have been omitted for clarity), and the effects these options have on restoring from a MySQL Cluster backup:

Option Used	Result
--exclude-databases=db1	All tables in all databases except db1 are restored; no tables in db1 are restored
--exclude-databases=db1,db2 (or --exclude-databases=db1 --exclude-databases=db2)	All tables in all databases except db1 and db2 are restored; no tables in db1 or db2 are restored
--exclude-tables=db1.t1	All tables except t1 in database db1 are restored; all other tables in db1 are restored; all tables in all other databases are restored
--exclude-tables=db1.t2,db2.t1 (or --exclude-tables=db1.t2 --exclude-tables=db2.t1)	All tables in database db1 except for t2 and all tables in database db2 except for table t1 are restored; no other tables in db1 or db2 are restored; all tables in all other databases are restored

You can use these two options together. For example, the following causes all tables in all databases *except for* databases db1 and db2, and tables t1 and t2 in database db3, to be restored:

```
shell> ndb_restore [...] --exclude-databases=db1,db2 --exclude-tables=db3.t1,db3.t2
```

(Again, we have omitted other possibly necessary options in the interest of clarity and brevity from the example just shown.)

You can use --include-* and --exclude-* options together, subject to the following rules:

- The actions of all --include-* and --exclude-* options are cumulative.
- All --include-* and --exclude-* options are evaluated in the order passed to ndb_restore, from right to left.
- In the event of conflicting options, the first (rightmost) option takes precedence. In other words, the first option (going from right to left) that matches against a given database or table “wins”.

For example, the following set of options causes ndb_restore to restore all tables from database db1 except db1.t1, while restoring no other tables from any other databases:

```
--include-databases=db1 --exclude-tables=db1.t1
```

However, reversing the order of the options just given simply causes all tables from database db1 to be restored (including db1.t1, but no tables from any other database), because the --include-databases [311] option, being farthest to the right, is the first match against database db1 and thus takes precedence over any other option that matches db1 or any tables in db1:

```
--exclude-tables=db1.t1 --include-databases=db1
```

--exclude-missing-columns

Command-Line Format	--exclude-missing-columns

It is also possible to restore only selected table columns using the --exclude-missing-columns option. When this option is used, ndb_restore ignores any columns missing from tables being restored as compared to the versions of those tables found in the backup. This option applies to all tables being restored. If you wish to apply this option only to selected tables or databases, you can use it in combination with one or more of the options described in the previous paragraph to do so, then restore data to the remaining tables using a complementary set of these options.

--exclude-missing-tables

Introduced	5.6.21-ndb-7.3.7
Command-Line Format	--exclude-missing-tables

Beginning with MySQL Cluster NDB 7.3.7, it is also possible to restore only selected tables columns using this option, which causes ndb_restore to ignore any tables from the backup that are not found in the target database.

--disable-indexes

Command-Line Format	--disable-indexes

Disable restoration of indexes during restoration of the data from a native NDB backup. Afterwards, you can restore indexes for all tables at once with multi-threaded building of indexes using --rebuild-indexes [313], which should be faster than rebuilding indexes concurrently for very large tables.

--rebuild-indexes

Command-Line Format	--rebuild-indexes

You can use this option with ndb_restore to cause multi-threaded rebuilding of the ordered indexes while restoring a native NDB backup. The number of threads used for building ordered indexes by ndb_restore with this option is controlled by the BuildIndexThreads data node configuration parameter.

It is necessary to use this option only for the first run of ndb_restore; this causes all ordered indexes to be rebuilt without using --rebuild-indexes again when restoring subsequent nodes. You should use this option prior to inserting new rows into the database; otherwise, it is possible for a row to be inserted that later causes a unique constraint violation when trying to rebuild the indexes.

Rebuilding of unique indexes uses disk write bandwidth for redo logging and local checkpointing. An insufficient amount of this bandwith can lead to redo buffer overload or log overload errors. In such cases you can run `ndb_restore --rebuild-indexes` again; the process resumes at the point where the error occurred. You can also do this when you have encountered temporarary errors. You can repeat execution of `ndb_restore --rebuild-indexes` indefinitely; you may be able to stop such errors by reducing the value of `DiskCheckpointSpeed` to provide additional disk bandwidth to redo logging.

`--skip-broken-objects`

Command-Line Format	`--skip-broken-objects`

This option causes `ndb_restore` to ignore corrupt tables while reading a native `NDB` backup, and to continue restoring any remaining tables (that are not also corrupted). Currently, the `--skip-broken-objects` option works only in the case of missing blob parts tables.

`--skip-unknown-objects`

Command-Line Format	`--skip-unknown-objects`

This option causes `ndb_restore` to ignore any schema objects it does not recognize while reading a native `NDB` backup. This can be used for restoring a backup made from a cluster running MySQL Cluster NDB 7.3 to a cluster running MySQL Cluster NDB 7.2.

`--rewrite-database=`*`old_dbname,new_dbname`*

Command-Line Format	`--rewrite-database=olddb,newdb`	
Permitted Values	**Type**	`string`
	Default	`none`

This option makes it possible to restore to a database having a different name from that used in the backup. For example, if a backup is made of a database named `products`, you can restore the data it contains to a database named `inventory`, use this option as shown here (omitting any other options that might be required):

```
shell> ndb_restore --rewrite-database=product,inventory
```

The option can be employed multiple times in a single invocation of `ndb_restore`. Thus it is possible to restore simultaneously from a database named `db1` to a database named `db2` and from a database named `db3` to one named `db4` using `--rewrite-database=db1,db2 --rewrite-database=db3,db4`. Other `ndb_restore` options may be used between multiple occurrences of `--rewrite-database`.

In the event of conflicts between multiple `--rewrite-database` options, the last `--rewrite-database` option used, reading from left to right, is the one that takes effect. For example, if `--rewrite-database=db1,db2 --rewrite-database=db1,db3` is used, only `--rewrite-database=db1,db3` is honored, and `--rewrite-database=db1,db2` is ignored. It is also possible to restore from multiple databases to a single database, so that `--rewrite-database=db1,db3 --rewrite-database=db2,db3` restores all tables and data from databases `db1` and `db2` into database `db3`.

> **Important**
>
> When restoring from multiple backup databases into a single target database using `--rewrite-database`, no check is made for collisions between table or other object names, and the order in which rows are restored is not guaranteed. This means that it is possible in such cases for rows to be overwritten and updates to be lost.

`--exclude-intermediate-sql-tables[=TRUE|FALSE]`

Introduced	5.6.17-ndb-7.3.6		
Command-Line Format	--exclude-intermediate-sql-tables[=TRUE|FALSE]		
Permitted Values	**Type**	boolean	
	Default	TRUE	

When performing copying ALTER TABLE operations, mysqld creates intermediate tables (whose names are prefixed with #sql-). When TRUE, the --exclude-intermediate-sql-tables option keeps ndb_restore from restoring such tables that may have been left over from such operations. This option is TRUE by default.

The --exclude-intermediate-sql-tables option was introduced in MySQL Cluster NDB 7.3.6. (Bug #17882305)

Error reporting.
ndb_restore reports both temporary and permanent errors. In the case of temporary errors, it may able to recover from them, and reports Restore successful, but encountered temporary error, please look at configuration in such cases.

> **Important**
>
> After using ndb_restore to initialize a MySQL Cluster for use in circular replication, binary logs on the SQL node acting as the replication slave are not automatically created, and you must cause them to be created manually. To cause the binary logs to be created, issue a SHOW TABLES statement on that SQL node before running START SLAVE. This is a known issue in MySQL Cluster.

6.21 ndb_select_all — Print Rows from an NDB Table

ndb_select_all prints all rows from an NDB table to stdout.

Usage

```
ndb_select_all -c connection_string tbl_name -d db_name [> file_name]
```

The following table includes options that are specific to the MySQL Cluster native backup restoration program ndb_select_all. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_select_all), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.15 This table describes command-line options for the ndb_select_all program

Format	Description	Added or Removed
--database=dbname, -d	Name of the database in which the table is found	All MySQL 5.6 based releases
--parallelism=#, -p	Degree of parallelism	All MySQL 5.6 based releases
--lock=#, -l	Lock type	All MySQL 5.6 based releases
--order=index, -o	Sort resultset according to index whose name is supplied	All MySQL 5.6 based releases
--descending, -z	Sort resultset in descending order (requires order flag)	All MySQL 5.6 based releases

Format	Description	Added or Removed
`--header`, `-h`	Print header (set to 0|FALSE to disable headers in output)	All MySQL 5.6 based releases
`--useHexFormat`, `-x`	Output numbers in hexadecimal format	All MySQL 5.6 based releases
`--delimiter=char`, `-D`	Set a column delimiter	All MySQL 5.6 based releases
`--disk`	Print disk references (useful only for Disk Data tables having nonindexed columns)	All MySQL 5.6 based releases
`--rowid`	Print rowid	All MySQL 5.6 based releases
`--gci`	Include GCI in output	All MySQL 5.6 based releases
`--gci64`	Include GCI and row epoch in output	All MySQL 5.6 based releases
`--tup`, `-t`	Scan in tup order	All MySQL 5.6 based releases
`--nodata`	Do not print table column data	All MySQL 5.6 based releases

- `--database=`*`dbname`*, `-d` *`dbname`*

  Name of the database in which the table is found. The default value is `TEST_DB`.

- `parallelism=`*`#`*, `-p` *`#`*

  Specifies the degree of parallelism.

- `--lock=`*`lock_type`*, `-l` *`lock_type`*

  Employs a lock when reading the table. Possible values for *`lock_type`* are:

  - `0`: Read lock
  - `1`: Read lock with hold
  - `2`: Exclusive read lock

  There is no default value for this option.

- `--order=`*`index_name`*, `-o` *`index_name`*

  Orders the output according to the index named *`index_name`*.

  > **Note**
  >
  > This is the name of an index, not of a column, and that the index must have been explicitly named when created.

- `--descending`, `-z`

  Sorts the output in descending order. This option can be used only in conjunction with the `-o` (`--order`) option.

- `--header=FALSE`

  Excludes column headers from the output.

- `--useHexFormat -x`

  Causes all numeric values to be displayed in hexadecimal format. This does not affect the output of numerals contained in strings or datetime values.

- `--delimiter=character`, `-D character`

  Causes the *`character`* to be used as a column delimiter. Only table data columns are separated by this delimiter.

  The default delimiter is the tab character.

- `--disk`

  Adds a disk reference column to the output. The column is nonempty only for Disk Data tables having nonindexed columns.

- `--rowid`

  Adds a `ROWID` column providing information about the fragments in which rows are stored.

- `--gci`

  Adds a `GCI` column to the output showing the global checkpoint at which each row was last updated. See Chapter 3, *MySQL Cluster Overview*, and Section 7.6.2, “MySQL Cluster Log Events”, for more information about checkpoints.

- `--gci64`

  Adds a `ROW$GCI64` column to the output showing the global checkpoint at which each row was last updated, as well as the number of the epoch in which this update occurred.

- `--tupscan`, `-t`

  Scan the table in the order of the tuples.

- `--nodata`

  Causes any table data to be omitted.

Sample Output

Output from a MySQL `SELECT` statement:

```
mysql> SELECT * FROM ctest1.fish;
+----+-----------+
| id | name      |
+----+-----------+
|  3 | shark     |
|  6 | puffer    |
|  2 | tuna      |
|  4 | manta ray |
|  5 | grouper   |
|  1 | guppy     |
+----+-----------+
6 rows in set (0.04 sec)
```

Output from the equivalent invocation of `ndb_select_all`:

```
shell> ./ndb_select_all -c localhost fish -d ctest1
id      name
3       [shark]
6       [puffer]
2       [tuna]
```

```
4       [manta ray]
5       [grouper]
1       [guppy]
6 rows returned
NDBT_ProgramExit: 0 - OK
```

All string values are enclosed by square brackets ("[...]") in the output of ndb_select_all. For another example, consider the table created and populated as shown here:

```
CREATE TABLE dogs (
    id INT(11) NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL,
    breed VARCHAR(50) NOT NULL,
    PRIMARY KEY pk (id),
    KEY ix (name)
)
TABLESPACE ts STORAGE DISK
ENGINE=NDBCLUSTER;
INSERT INTO dogs VALUES
    ('', 'Lassie', 'collie'),
    ('', 'Scooby-Doo', 'Great Dane'),
    ('', 'Rin-Tin-Tin', 'Alsatian'),
    ('', 'Rosscoe', 'Mutt');
```

This demonstrates the use of several additional ndb_select_all options:

```
shell> ./ndb_select_all -d ctest1 dogs -o ix -z --gci --disk
GCI     id name          breed        DISK_REF
834461  2  [Scooby-Doo] [Great Dane] [ m_file_no: 0 m_page: 98 m_page_idx: 0 ]
834878  4  [Rosscoe]     [Mutt]       [ m_file_no: 0 m_page: 98 m_page_idx: 16 ]
834463  3  [Rin-Tin-Tin] [Alsatian]   [ m_file_no: 0 m_page: 34 m_page_idx: 0 ]
835657  1  [Lassie]      [Collie]     [ m_file_no: 0 m_page: 66 m_page_idx: 0 ]
4 rows returned
NDBT_ProgramExit: 0 - OK
```

6.22 ndb_select_count — Print Row Counts for NDB Tables

ndb_select_count prints the number of rows in one or more NDB tables. With a single table, the result is equivalent to that obtained by using the MySQL statement SELECT COUNT(*) FROM *tbl_name*.

Usage

```
ndb_select_count [-c connection_string] -ddb_name tbl_name[, tbl_name2[, ...]]
```

The following table includes options that are specific to the MySQL Cluster native backup restoration program ndb_select_count. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_select_count), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.16 This table describes command-line options for the ndb_select_count program

Format	Description	Added or Removed
--database=dbname, -d	Name of the database in which the table is found	All MySQL 5.6 based releases
--parallelism=#, -p	Degree of parallelism	All MySQL 5.6 based releases
--lock=#, -l	Lock type	All MySQL 5.6 based releases

You can obtain row counts from multiple tables in the same database by listing the table names separated by spaces when invoking this command, as shown under **Sample Output**.

Sample Output

```
shell> ./ndb_select_count -c localhost -d ctest1 fish dogs
6 records in table fish
4 records in table dogs
NDBT_ProgramExit: 0 - OK
```

6.23 ndb_setup.py — Start browser-based Auto-Installer for MySQL Cluster

`ndb_setup.py` starts the MySQL Cluster Auto-Installer and opens the installer's Start page in the default Web browser.

This section describes usage of and program options for the command-line tool only. For information about using the Auto-Installer GUI that is spawned when `ndb_setup.py` is invoked, see Section 4.1, "The MySQL Cluster Auto-Installer".

Usage

All platforms:

```
ndb_setup.py [options]
```

Additionally, on Windows platforms only:

```
setup.bat [options]
```

The following table includes all options that are supported by the MySQL Cluster installation and configuration program `ndb_setup.py`. Additional descriptions follow the table.

Table 6.17 This table describes command-line options for the ndb_setup.py program

Format	Description	Added or Removed
`--browser-start-page=filename`, `-s`	Page that the web browser opens when starting.	All MySQL 5.6 based releases
`--ca-certs-file=filename`, `-a`	File containing list of client certificates allowed to connect to the server	All MySQL 5.6 based releases
`--cert-file=filename`, `-c`	File containing X509 certificate that identifies the server. (Default: cfg.pem)	All MySQL 5.6 based releases
`--debug-level=level`, `-d`	Python logging module debug level. One of DEBUG, INFO, WARNING (default), ERROR, or CRITICAL.	All MySQL 5.6 based releases
`--help`, `-h`	Print help message	All MySQL 5.6 based releases

Format	Description	Added or Removed
`--key-file=file`, `-k`	Specify file containing private key (if not included in --cert-file)	All MySQL 5.6 based releases
`--no-browser`, `-n`	Do not open the start page in a browser, merely start the tool	All MySQL 5.6 based releases
`--port=#`, `-p`	Specify the port used by the web server	All MySQL 5.6 based releases
`--server-log-file=file`, `o`	Log requests to this file. Use '-' to force logging to stderr instead.	All MySQL 5.6 based releases
`--server-name=name`, `-N`	The name of the server to connect with	All MySQL 5.6 based releases
`--use-https`, `-S`	Use secure (HTTPS) client-server connection	All MySQL 5.6 based releases

- `--browser-start-page=`*`file`*, `-s`

Command-Line Format	`--browser-start-page=filename`	
Permitted Values	Type	`string`
	Default	`index.html`

Specify the file to open in the browser as the installation and configuration Start page. The default is `index.html`.

- `--ca-certs-file=`*`file`*, `-a`

Command-Line Format	`--ca-certs-file=filename`	
Permitted Values	Type	`file name`
	Default	`[none]`

Specify a file containing a list of client certificates which are allowed to connect to the server. The default is an empty string, which means that no client authentication is used.

- `--cert-file=`*`file`*, `-c`

Command-Line Format	`--cert-file=filename`	
Permitted Values	Type	`file name`
	Default	`cfg.pem`

Specify a file containing an X509 certificate which identifies the server. It is possible for the certificate to be self-signed. The default is `cfg.pem`.

- `--debug-level=`*`level`*, `-d`

Command-Line Format	`--debug-level=level`	
Permitted Values	Type	`enumeration`
	Default	`WARNING`
	Valid Values	`WARNING`

		DEBUG
		INFO
		ERROR
		CRITICAL

Set the Python logging module debug level. This is one of `DEBUG`, `INFO`, `WARNING`, `ERROR`, or `CRITICAL`. `WARNING` is the default.

- `--help`, `-h`

Command-Line Format	`--help`

Print a help message.

- `--key-file=`*`file`*, `-d`

Command-Line Format	`--key-file=file`	
Permitted Values	**Type**	`file name`
	Default	`[none]`

Specify a file containing the private key if this is not included in the X509 certificate file (`--cert-file`). The default is an empty string, which means that no such file is used.

- `--no-browser`, `-n`

Command-Line Format	`--no-browser`

Start the installation and configuration tool, but do not open the Start page in a browser.

- `--port=`*`#`*, `-p`

Command-Line Format	`--port=#`	
Permitted Values	**Type**	`numeric`
	Default	`8081`
	Min Value	`1`
	Max Value	`65535`

Set the port used by the web server. The default is 8081.

- `--server-log-file=`*`file`*, `-o`

Command-Line Format	`--server-log-file=file`	
	`o`	
Permitted Values	**Type**	`file name`
	Default	`ndb_setup.log`
	Valid Values	`ndb_setup.log`
		`-` (Log to stderr)

Log requests to this file. The default is `ndb_setup.log`. To specify logging to `stderr`, rather than to a file, use a `-` (dash character) for the file name.

- `--server-name=`*`host`*, `-N`

Command-Line Format	--server-name=name	
Permitted Values	Type	string
	Default	localhost

Specify the host name or IP address for the browser to use when connecting. The default is localhost.

- --use-https, -S

Command-Line Format	--use-https

Make the browser use a secure (HTTPS) connection with the server.

6.24 ndb_show_tables — Display List of NDB Tables

ndb_show_tables displays a list of all NDB database objects in the cluster. By default, this includes not only both user-created tables and NDB system tables, but NDB-specific indexes, internal triggers, and MySQL Cluster Disk Data objects as well.

The following table includes options that are specific to the MySQL Cluster native backup restoration program ndb_show_tables. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_show_tables), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.18 This table describes command-line options for the ndb_show_tables program

Format	Description	Added or Removed
--database=string, -d	Specifies the database in which the table is found	All MySQL 5.6 based releases
--loops=#, -l	Number of times to repeat output	All MySQL 5.6 based releases
--type=#, -t	Limit output to objects of this type	All MySQL 5.6 based releases
--unqualified, -u	Do not qualify table names	All MySQL 5.6 based releases
--parsable, -p	Return output suitable for MySQL LOAD DATA INFILE statement	All MySQL 5.6 based releases
--show-temp-status	Show table temporary flag	All MySQL 5.6 based releases

Usage

```
ndb_show_tables [-c connection_string]
```

- --database, -d

  Specifies the name of the database in which the tables are found. In MySQL Cluster NDB 7.4.8 and later, if this option has not been specified, and no tables are found in the TEST_DB database, ndb_show_tables issues a warning (Bug #50633, Bug #11758430).

- --loops, -l

Specifies the number of times the utility should execute. This is 1 when this option is not specified, but if you do use the option, you must supply an integer argument for it.

- `--parsable`, `-p`

  Using this option causes the output to be in a format suitable for use with `LOAD DATA INFILE`.

- `--show-temp-status`

  If specified, this causes temporary tables to be displayed.

- `--type`, `-t`

  Can be used to restrict the output to one type of object, specified by an integer type code as shown here:

  - `1`: System table
  - `2`: User-created table
  - `3`: Unique hash index

  Any other value causes all `NDB` database objects to be listed (the default).

- `--unqualified`, `-u`

  If specified, this causes unqualified object names to be displayed.

> **Note**
>
> Only user-created MySQL Cluster tables may be accessed from MySQL; system tables such as `SYSTAB_0` are not visible to `mysqld`. However, you can examine the contents of system tables using `NDB` API applications such as `ndb_select_all` (see Section 6.21, "`ndb_select_all` — Print Rows from an NDB Table").

6.25 `ndb_size.pl` — NDBCLUSTER Size Requirement Estimator

This is a Perl script that can be used to estimate the amount of space that would be required by a MySQL database if it were converted to use the `NDBCLUSTER` storage engine. Unlike the other utilities discussed in this section, it does not require access to a MySQL Cluster (in fact, there is no reason for it to do so). However, it does need to access the MySQL server on which the database to be tested resides.

Requirements

- A running MySQL server. The server instance does not have to provide support for MySQL Cluster.
- A working installation of Perl.
- The `DBI` module, which can be obtained from CPAN if it is not already part of your Perl installation. (Many Linux and other operating system distributions provide their own packages for this library.)
- A MySQL user account having the necessary privileges. If you do not wish to use an existing account, then creating one using `GRANT USAGE ON db_name.*`—where *`db_name`* is the name of the database to be examined—is sufficient for this purpose.

`ndb_size.pl` can also be found in the MySQL sources in `storage/ndb/tools`.

The following table includes options that are specific to the MySQL Cluster program `ndb_size.pl`. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_size.pl), see Section 6.27, “Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs”.

Table 6.19 This table describes command-line options for the ndb_size.pl program

Format	Description	Added or Removed
--database=dbname [324]	The database or databases to examine; accepts a comma-delimited list; the default is ALL (use all databases found on the server)	All MySQL 5.6 based releases
--hostname[:port] [324]	Specify host and optional port as host[:port]	All MySQL 5.6 based releases
--socket=file_name [324]	Specify a socket to connect to	All MySQL 5.6 based releases
--user=string [324]	Specify a MySQL user name	All MySQL 5.6 based releases
--password=string [324]	Specify a MySQL user password	All MySQL 5.6 based releases
--format=string [325]	Set output format (text or HTML)	All MySQL 5.6 based releases
--excludetables=tbl_list [324]	Skip any tables in a comma-separated list of tables	All MySQL 5.6 based releases
--excludedbs=db_list [324]	Skip any databases in a comma-separated list of databases	All MySQL 5.6 based releases
--savequeries=file [325]	Saves all queries to the database into the file specified	All MySQL 5.6 based releases
--loadqueries=file [325]	Loads all queries from the file specified; does not connect to a database	All MySQL 5.6 based releases
--real_table_name=table [324]	Designates a table to handle unique index size calculations	All MySQL 5.6 based releases

Usage

```
perl ndb_size.pl [--database={db_name|ALL}] [--hostname=host[:port]] [--socket=socket] \
      [--user=user] [--password=password]  \
      [--help|-h] [--format={html|text}] \
      [--loadqueries=file_name] [--savequeries=file_name]
```

By default, this utility attempts to analyze all databases on the server. You can specify a single database using the --database option; the default behavior can be made explicit by using ALL for the name of the database. You can also exclude one or more databases by using the --excludedbs option with a comma-separated list of the names of the databases to be skipped. Similarly, you can cause specific tables to be skipped by listing their names, separated by commas, following the optional --excludetables option. A host name can be specified using --hostname; the default is localhost. You can specify a port in addition to the host using *host*:*port* format for the value of --hostname. The default port number is 3306. If necessary, you can also specify a socket; the default is /var/lib/mysql.sock. A MySQL user name and password can be specified the corresponding options shown. It also possible to control the format of the output using the --format option; this can take either of the values html or text, with text being the default. An example of the text output is shown here:

```
shell> ndb_size.pl --database=test --socket=/tmp/mysql.sock
ndb_size.pl report for database: 'test' (1 tables)
--------------------------------------------------
Connected to: DBI:mysql:host=localhost;mysql_socket=/tmp/mysql.sock
Including information for versions: 4.1, 5.0, 5.1
test.t1
-------
```

```
DataMemory for Columns (* means varsized DataMemory):
         Column Name          Type  Varsized   Key  4.1  5.0   5.1
     HIDDEN_NDB_PKEY        bigint             PRI    8    8     8
                  c2   varchar(50)         Y         52   52    4*
                  c1       int(11)                    4    4     4
                                                     --   --    --
Fixed Size Columns DM/Row                            64   64    12
   Varsize Columns DM/Row                             0    0     4
DataMemory for Indexes:
   Index Name                 Type        4.1        5.0        5.1
      PRIMARY                BTREE         16         16         16
                                           --         --         --
       Total Index DM/Row                  16         16         16
IndexMemory for Indexes:
               Index Name        4.1        5.0        5.1
                  PRIMARY         33         16         16
                                  --         --         --
           Indexes IM/Row         33         16         16
Summary (for THIS table):
                                 4.1        5.0        5.1
    Fixed Overhead DM/Row         12         12         16
           NULL Bytes/Row          4          4          4
           DataMemory/Row         96         96         48
                    (Includes overhead, bitmap and indexes)
  Varsize Overhead DM/Row          0          0          8
   Varsize NULL Bytes/Row          0          0          4
       Avg Varside DM/Row          0          0         16
                 No. Rows          0          0          0
        Rows/32kb DM Page        340        340        680
Fixedsize DataMemory (KB)          0          0          0
Rows/32kb Varsize DM Page          0          0       2040
  Varsize DataMemory (KB)          0          0          0
         Rows/8kb IM Page        248        512        512
         IndexMemory (KB)          0          0          0
Parameter Minimum Requirements
------------------------------
* indicates greater than default
                Parameter     Default        4.1         5.0         5.1
          DataMemory (KB)       81920          0           0           0
       NoOfOrderedIndexes         128          1           1           1
               NoOfTables         128          1           1           1
         IndexMemory (KB)       18432          0           0           0
    NoOfUniqueHashIndexes          64          0           0           0
           NoOfAttributes        1000          3           3           3
             NoOfTriggers         768          5           5           5
```

For debugging purposes, the Perl arrays containing the queries run by this script can be read from the file specified using can be saved to a file using `--savequeries`; a file containing such arrays to be read in during script execution can be specified using `--loadqueries`. Neither of these options has a default value.

To produce output in HTML format, use the `--format` option and redirect the output to a file, as shown here:

```
shell> ndb_size.pl --database=test --socket=/tmp/mysql.sock --format=html > ndb_size.html
```

(Without the redirection, the output is sent to `stdout`.)

The output from this script includes the following information:

- Minimum values for the `DataMemory`, `IndexMemory`, `MaxNoOfTables`, `MaxNoOfAttributes`, `MaxNoOfOrderedIndexes`, `MaxNoOfUniqueHashIndexes`, and `MaxNoOfTriggers` configuration parameters required to accommodate the tables analyzed.
- Memory requirements for all of the tables, attributes, ordered indexes, and unique hash indexes defined in the database.
- The `IndexMemory` and `DataMemory` required per table and table row.

6.26 ndb_waiter — Wait for MySQL Cluster to Reach a Given Status

ndb_waiter repeatedly (each 100 milliseconds) prints out the status of all cluster data nodes until either the cluster reaches a given status or the --timeout limit is exceeded, then exits. By default, it waits for the cluster to achieve STARTED status, in which all nodes have started and connected to the cluster. This can be overridden using the --no-contact and --not-started options.

The node states reported by this utility are as follows:

- NO_CONTACT: The node cannot be contacted.
- UNKNOWN: The node can be contacted, but its status is not yet known. Usually, this means that the node has received a START or RESTART command from the management server, but has not yet acted on it.
- NOT_STARTED: The node has stopped, but remains in contact with the cluster. This is seen when restarting the node using the management client's RESTART command.
- STARTING: The node's ndbd process has started, but the node has not yet joined the cluster.
- STARTED: The node is operational, and has joined the cluster.
- SHUTTING_DOWN: The node is shutting down.
- SINGLE USER MODE: This is shown for all cluster data nodes when the cluster is in single user mode.

The following table includes options that are specific to the MySQL Cluster native backup restoration program ndb_waiter. Additional descriptions follow the table. For options common to most MySQL Cluster programs (including ndb_waiter), see Section 6.27, "Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs".

Table 6.20 This table describes command-line options for the ndb_waiter program

Format	Description	Added or Removed
--no-contact, -n	Wait for cluster to reach NO CONTACT state	All MySQL 5.6 based releases
--not-started	Wait for cluster to reach NOT STARTED state	All MySQL 5.6 based releases
--single-user	Wait for cluster to enter single user mode	All MySQL 5.6 based releases
--timeout=#, -t	Wait this many seconds, then exit whether or not cluster has reached desired state; default is 2 minutes (120 seconds)	All MySQL 5.6 based releases
--nowait-nodes=list	List of nodes not to be waited for.	All MySQL 5.6 based releases
--wait-nodes=list, -w	List of nodes to be waited for.	All MySQL 5.6 based releases

Usage

```
ndb_waiter [-c connection_string]
```

Additional Options

- `--no-contact`, `-n`

  Instead of waiting for the `STARTED` state, `ndb_waiter` continues running until the cluster reaches `NO_CONTACT` status before exiting.

- `--not-started`

  Instead of waiting for the `STARTED` state, `ndb_waiter` continues running until the cluster reaches `NOT_STARTED` status before exiting.

- `--timeout=`*`seconds`*, `-t` *`seconds`*

  Time to wait. The program exits if the desired state is not achieved within this number of seconds. The default is 120 seconds (1200 reporting cycles).

- `--single-user`

  The program waits for the cluster to enter single user mode.

- `--nowait-nodes=`*`list`*

  When this option is used, ndb_waiter does not wait for the nodes whose IDs are listed. The list is comma-delimited; ranges can be indicated by dashes, as shown here:

  ```
  shell> ndb_waiter --nowait-nodes=1,3,7-9
  ```

  > **Important**
  >
  > Do *not* use this option together with the `--wait-nodes` option.

- `--wait-nodes=`*`list`*, `-w` *`list`*

  When this option is used, `ndb_waiter` waits only for the nodes whose IDs are listed. The list is comma-delimited; ranges can be indicated by dashes, as shown here:

  ```
  shell> ndb_waiter --wait-nodes=2,4-6,10
  ```

  > **Important**
  >
  > Do *not* use this option together with the `--nowait-nodes` option.

Sample Output. Shown here is the output from `ndb_waiter` when run against a 4-node cluster in which two nodes have been shut down and then started again manually. Duplicate reports (indicated by "`...`") are omitted.

```
shell> ./ndb_waiter -c localhost
Connecting to mgmsrv at (localhost)
State node 1 STARTED
State node 2 NO_CONTACT
State node 3 STARTED
State node 4 NO_CONTACT
Waiting for cluster enter state STARTED
...
State node 1 STARTED
State node 2 UNKNOWN
State node 3 STARTED
State node 4 NO_CONTACT
Waiting for cluster enter state STARTED
...
State node 1 STARTED
State node 2 STARTING
```

```
State node 3 STARTED
State node 4 NO_CONTACT
Waiting for cluster enter state STARTED
...
State node 1 STARTED
State node 2 STARTING
State node 3 STARTED
State node 4 UNKNOWN
Waiting for cluster enter state STARTED
...
State node 1 STARTED
State node 2 STARTING
State node 3 STARTED
State node 4 STARTING
Waiting for cluster enter state STARTED
...
State node 1 STARTED
State node 2 STARTED
State node 3 STARTED
State node 4 STARTING
Waiting for cluster enter state STARTED
...
State node 1 STARTED
State node 2 STARTED
State node 3 STARTED
State node 4 STARTED
Waiting for cluster enter state STARTED
NDBT_ProgramExit: 0 - OK
```

> **Note**
>
> If no connection string is specified, then `ndb_waiter` tries to connect to a management on `localhost`, and reports `Connecting to mgmsrv at (null)`.

6.27 Options Common to MySQL Cluster Programs — Options Common to MySQL Cluster Programs

All MySQL Cluster programs accept the options described in this section, with the following exceptions:

- `mysqld`
- `ndb_print_backup_file`
- `ndb_print_schema_file`
- `ndb_print_sys_file`

> **Note**
>
> Users of earlier MySQL Cluster versions should note that some of these options have been changed to make them consistent with one another, and also with `mysqld`. You can use the `--help` option with any MySQL Cluster program—with the exception of `ndb_print_backup_file`, `ndb_print_schema_file`, and `ndb_print_sys_file`—to view a list of the options which the program supports.

The options in the following table are common to all MySQL Cluster executables (except those noted previously in this section).

Table 6.21 This table describes command-line options common to all MySQL Cluster programs

Format	Description	Added or Removed
`--help`, `--usage`,	Display help message and exit	All MySQL 5.6 based releases

Format	Description	Added or Removed
`-?`		
`--character-sets-dir=dir_name`	Directory where character sets are installed	All MySQL 5.6 based releases
`--connect-retries=#`	Set the number of times to retry a connection before giving up	ADDED: NDB 7.4.9
`--connect-retry-delay=#`	Time to wait between attempts to contact a management server, in seconds	ADDED: NDB 7.4.9
`--core-file`	Write core on errors (defaults to TRUE in debug builds)	All MySQL 5.6 based releases
`--debug=options`	Enable output from debug calls. Can be used only for versions compiled with debugging enabled	All MySQL 5.6 based releases
`--ndb-connectstring=connectstring`, `--connect-string=connectstring`, `-c`	Set connection string for connecting to ndb_mgmd. Syntax: [nodeid=<id>;][host=]<hostname>[:<port>]. Overrides entries specified in NDB_CONNECTSTRING or my.cnf.	All MySQL 5.6 based releases
`--ndb-mgmd-host=host[:port]`	Set the host (and port, if desired) for connecting to management server	All MySQL 5.6 based releases
`--ndb-nodeid=#`	Set node id for this node	All MySQL 5.6 based releases
`--ndb-optimized-node-selection`	Select nodes for transactions in a more optimal way	All MySQL 5.6 based releases
`--version`, `-V`	Output version information and exit	All MySQL 5.6 based releases

For options specific to individual MySQL Cluster programs, see Chapter 6, *MySQL Cluster Programs*.

See Section 5.3.8.1, “MySQL Server Options for MySQL Cluster”, for `mysqld` options relating to MySQL Cluster.

- `--help`, `--usage`, `-?`

Command-Line Format	`--help`
	`--usage`

Prints a short list with descriptions of the available command options.

- `--character-sets-dir=name`

Command-Line Format	`--character-sets-dir=dir_name`	
Permitted Values	Type	`directory name`
	Default	

Tells the program where to find character set information.

- `--ndb-connectstring=connection_string`, `--connect-string=connection_string`, `-c connection_string`

Command-Line Format	`--ndb-connectstring=connectstring`	
	`--connect-string=connectstring`	
Permitted Values	**Type**	`string`
	Default	`localhost:1186`

This option takes a MySQL Cluster connection string that specifies the management server for the application to connect to, as shown here:

```
shell> ndbd --ndb-connectstring="nodeid=2;host=ndb_mgmd.mysql.com:1186"
```

For more information, see Section 5.3.3, “MySQL Cluster Connection Strings”.

- `--connect-retries=#`

Introduced	5.6.28-ndb-7.4.9	
Command-Line Format	`--connect-retries=#`	
Permitted Values (>= 5.6.28-ndb-7.4.9)	**Type**	`numeric`
	Default	`12`
	Min Value	`0`
	Max Value	`4294967295`

This option specifies the number of times following the first attempt to retry a connection before giving up (the client always tries the connection at least once). The length of time to wait per attempt is set using `--connect-retry-delay`.

> **Note**
>
> When used with `ndb_mgm`, this option has 3 as its default. See Section 6.5, “`ndb_mgm` — The MySQL Cluster Management Client”, for more information.

This option was added in MySQL Cluster NDB 7.4.9.

- `--connect-retry-delay=#`

Introduced	5.6.28-ndb-7.4.9	
Command-Line Format	`--connect-retry-delay=#`	
Permitted Values (>= 5.6.28-ndb-7.4.9)	**Type**	`numeric`
	Default	`5`
	Min Value	`0`
	Max Value	`4294967295`

This option specifies the length of time to wait per attempt a connection before giving up. The number of times to try connecting is set by `--connect-retries`.

This option was added in MySQL Cluster NDB 7.4.9.

- `--core-file`

Command-Line Format	`--core-file`

Permitted Values	**Type**	`boolean`
	Default	`FALSE`

Write a core file if the program dies. The name and location of the core file are system-dependent. (For MySQL Cluster programs nodes running on Linux, the default location is the program's working directory—for a data node, this is the node's `DataDir`.) For some systems, there may be restrictions or limitations; for example, it might be necessary to execute `ulimit -c unlimited` before starting the server. Consult your system documentation for detailed information.

If MySQL Cluster was built using the `--debug` option for `configure`, then `--core-file` is enabled by default. For regular builds, `--core-file` is disabled by default.

- `--debug[=`*`options`*`]`

Command-Line Format	`--debug=options`	
Permitted Values	**Type**	`string`
	Default	`d:t:O,/tmp/ndb_restore.trace`

This option can be used only for versions compiled with debugging enabled. It is used to enable output from debug calls in the same manner as for the `mysqld` process.

- `--ndb-mgmd-host=`*`host`*`[:`*`port`*`]`

Command-Line Format	`--ndb-mgmd-host=host[:port]`	
Permitted Values	**Type**	`string`
	Default	`localhost:1186`

Can be used to set the host and port number of a single management server for the program to connect to. If the program requires node IDs or references to multiple management servers (or both) in its connection information, use the `--ndb-connectstring` option instead.

- `--ndb-nodeid=#`

Command-Line Format	`--ndb-nodeid=#`	
Permitted Values	**Type**	`numeric`
	Default	`0`

Sets this node's MySQL Cluster node ID. *The range of permitted values depends on the node's type (data, management, or API) and the MySQL Cluster software version*. See Section 3.6.2, “Limits and Differences of MySQL Cluster from Standard MySQL Limits”, for more information.

- `--ndb-optimized-node-selection`

Command-Line Format	`--ndb-optimized-node-selection`	
Permitted Values	**Type**	`boolean`
	Default	`TRUE`

Optimize selection of nodes for transactions. Enabled by default.

- `--version`, `-V`

Command-Line Format	`--version`

Prints the MySQL Cluster version number of the executable. The version number is relevant because not all versions can be used together, and the MySQL Cluster startup process verifies that the versions of the binaries being used can co-exist in the same cluster. This is also important when performing an online (rolling) software upgrade or downgrade of MySQL Cluster.

See Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”), for more information.

Chapter 7 Management of MySQL Cluster

Table of Contents

Managing a MySQL Cluster involves a number of tasks, the first of which is to configure and start MySQL Cluster. This is covered in Chapter 5, *Configuration of MySQL Cluster*, and Chapter 6, *MySQL Cluster Programs*.

The next few sections cover the management of a running MySQL Cluster.

For information about security issues relating to management and deployment of a MySQL Cluster, see Section 7.11, "MySQL Cluster Security Issues".

There are essentially two methods of actively managing a running MySQL Cluster. The first of these is through the use of commands entered into the management client whereby cluster status can be checked, log levels changed, backups started and stopped, and nodes stopped and started. The second method involves studying the contents of the cluster log `ndb_node_id_cluster.log`; this is usually found in the management server's `DataDir` directory, but this location can be overridden using the `LogDestination` option. (Recall that *`node_id`* represents the unique identifier of the node whose activity is being logged.) The cluster log contains event reports generated by `ndbd`. It is also possible to send cluster log entries to a Unix system log.

Some aspects of the cluster's operation can be also be monitored from an SQL node using the `SHOW ENGINE NDB STATUS` statement.

More detailed information about MySQL Cluster operations is available in real time through an SQL interface using the `ndbinfo` database. For more information, see Section 7.10, "The ndbinfo MySQL Cluster Information Database".

NDB statistics counters provide improved monitoring using the `mysql` client. These counters, implemented in the NDB kernel, relate to operations performed by or affecting `Ndb` objects, such as starting, closing, and aborting transactions; primary key and unique key operations; table, range, and pruned scans; blocked threads waiting for various operations to complete; and data and events sent and received by MySQL Cluster. The counters are incremented by the NDB kernel whenever NDB API calls are made or data is sent to or received by the data nodes.

`mysqld` exposes the NDB API statistics counters as system status variables, which can be identified from the prefix common to all of their names (`Ndb_api_`). The values of these variables can be read in the `mysql` client from the output of a `SHOW STATUS` statement, or by querying either the `SESSION_STATUS` table or the `GLOBAL_STATUS` table (in the `INFORMATION_SCHEMA` database). By comparing the values of the status variables before and after the execution of an SQL statement that acts on `NDB` tables, you can observe the actions taken on the NDB API level that correspond to this statement, which can be beneficial for monitoring and performance tuning of MySQL Cluster.

MySQL Cluster Manager provides an advanced command-line interface that simplifies many otherwise complex MySQL Cluster management tasks, such as starting, stopping, or restarting a MySQL Cluster with a large number of nodes. The MySQL Cluster Manager client also supports commands for getting and setting the values of most node configuration parameters as well as `mysqld` server options and variables relating to MySQL Cluster. See MySQL™ Cluster Manager 1.3.6 User Manual, for more information.

7.1 Summary of MySQL Cluster Start Phases

This section provides a simplified outline of the steps involved when MySQL Cluster data nodes are started. More complete information can be found in MySQL Cluster Start Phases, in the `NDB` *Internals Guide*.

These phases are the same as those reported in the output from the `node_id STATUS` command in the management client (see Section 7.2, "Commands in the MySQL Cluster Management Client"). These start phases are also reported in the `start_phase` column of the `ndbinfo.nodes` table.

Start types. There are several different startup types and modes, as shown in the following list:

- **Initial start.** The cluster starts with a clean file system on all data nodes. This occurs either when the cluster started for the very first time, or when all data nodes are restarted using the `--initial` option.

  > **Note**
  >
  > Disk Data files are not removed when restarting a node using `--initial`.

- **System restart.** The cluster starts and reads data stored in the data nodes. This occurs when the cluster has been shut down after having been in use, when it is desired for the cluster to resume operations from the point where it left off.

- **Node restart.** This is the online restart of a cluster node while the cluster itself is running.

- **Initial node restart.** This is the same as a node restart, except that the node is reinitialized and started with a clean file system.

Setup and initialization (phase -1). Prior to startup, each data node (`ndbd` process) must be initialized. Initialization consists of the following steps:

1. Obtain a node ID
2. Fetch configuration data
3. Allocate ports to be used for inter-node communications
4. Allocate memory according to settings obtained from the configuration file

When a data node or SQL node first connects to the management node, it reserves a cluster node ID. To make sure that no other node allocates the same node ID, this ID is retained until the node has managed to connect to the cluster and at least one `ndbd` reports that this node is connected. This retention of the node ID is guarded by the connection between the node in question and `ndb_mgmd`.

After each data node has been initialized, the cluster startup process can proceed. The stages which the cluster goes through during this process are listed here:

- **Phase 0.** The `NDBFS` and `NDBCNTR` blocks start (see NDB Kernel Blocks). Data node file systems are cleared on those data nodes that were started with `--initial` option.

- **Phase 1.** In this stage, all remaining `NDB` kernel blocks are started. MySQL Cluster connections are set up, inter-block communications are established, and heartbeats are started. In the case of a node restart, API node connections are also checked.

  > **Note**
  >
  > When one or more nodes hang in Phase 1 while the remaining node or nodes hang in Phase 2, this often indicates network problems. One possible cause of such issues is one or more cluster hosts having multiple network interfaces. Another common source of problems causing this condition is the blocking of TCP/IP ports needed for communications between cluster nodes. In the latter case, this is often due to a misconfigured firewall.

- **Phase 2.** The `NDBCNTR` kernel block checks the states of all existing nodes. The master node is chosen, and the cluster schema file is initialized.

- **Phase 3.** The DBLQH and DBTC kernel blocks set up communications between them. The startup type is determined; if this is a restart, the DBDIH block obtains permission to perform the restart.
- **Phase 4.** For an initial start or initial node restart, the redo log files are created. The number of these files is equal to NoOfFragmentLogFiles.

  For a system restart:

  - Read schema or schemas.
  - Read data from the local checkpoint.
  - Apply all redo information until the latest restorable global checkpoint has been reached.

  For a node restart, find the tail of the redo log.
- **Phase 5.** Most of the database-related portion of a data node start is performed during this phase. For an initial start or system restart, a local checkpoint is executed, followed by a global checkpoint. Periodic checks of memory usage begin during this phase, and any required node takeovers are performed.
- **Phase 6.** In this phase, node groups are defined and set up.
- **Phase 7.** The arbitrator node is selected and begins to function. The next backup ID is set, as is the backup disk write speed. Nodes reaching this start phase are marked as Started. It is now possible for API nodes (including SQL nodes) to connect to the cluster.
- **Phase 8.** If this is a system restart, all indexes are rebuilt (by DBDIH).
- **Phase 9.** The node internal startup variables are reset.
- **Phase 100 (OBSOLETE).** Formerly, it was at this point during a node restart or initial node restart that API nodes could connect to the node and begin to receive events. Currently, this phase is empty.
- **Phase 101.** At this point in a node restart or initial node restart, event delivery is handed over to the node joining the cluster. The newly-joined node takes over responsibility for delivering its primary data to subscribers. This phase is also referred to as *SUMA handover phase*.

After this process is completed for an initial start or system restart, transaction handling is enabled. For a node restart or initial node restart, completion of the startup process means that the node may now act as a transaction coordinator.

7.2 Commands in the MySQL Cluster Management Client

In addition to the central configuration file, a cluster may also be controlled through a command-line interface available through the management client ndb_mgm. This is the primary administrative interface to a running cluster.

Commands for the event logs are given in Section 7.6, “Event Reports Generated in MySQL Cluster”; commands for creating backups and restoring from them are provided in Section 7.3, “Online Backup of MySQL Cluster”.

Using ndb_mgm with MySQL Cluster Manager. MySQL Cluster Manager handles starting and stopping processes and tracks their states internally, so it is not necessary to use ndb_mgm for these tasks for a MySQL Cluster that is under MySQL Cluster Manager control. it is recommended *not* to use the ndb_mgm command-line client that comes with the MySQL Cluster distribution to perform operations that involve starting or stopping nodes. These include but are not limited to the START, STOP, RESTART, and SHUTDOWN commands. For more information, see MySQL Cluster Manager Process Commands.

The management client has the following basic commands. In the listing that follows, *node_id* denotes either a database node ID or the keyword ALL, which indicates that the command should be applied to all of the cluster's data nodes.

- HELP

  Displays information on all available commands.

- SHOW

  Displays information on the cluster's status. Possible node status values include UNKNOWN, NO_CONTACT, NOT_STARTED, STARTING, STARTED, SHUTTING_DOWN, and RESTARTING. The output from this command also indicates when the cluster is in single user mode (status SINGLE USER MODE).

- *node_id* START

  Brings online the data node identified by *node_id* (or all data nodes).

  ALL START works on all data nodes only, and does not affect management nodes.

  > **Important**
  >
  > To use this command to bring a data node online, the data node must have been started using ndbd --nostart or ndbd -n.

- *node_id* STOP [-a] [-f]

  Stops the data or management node identified by *node_id*.

  > **Note**
  >
  > ALL STOP works to stop all data nodes only, and does not affect management nodes.

  A node affected by this command disconnects from the cluster, and its associated ndbd or ndb_mgmd process terminates.

  The -a option causes the node to be stopped immediately, without waiting for the completion of any pending transactions.

  Normally, STOP fails if the result would cause an incomplete cluster. The -f option forces the node to shut down without checking for this. If this option is used and the result is an incomplete cluster, the cluster immediately shuts down.

  > **Warning**
  >
  > Use of the -a option also disables the safety check otherwise performed when STOP is invoked to insure that stopping the node does not cause an incomplete cluster. In other words, you should exercise extreme care when using the -a option with the STOP command, due to the fact that this option makes it possible for the cluster to undergo a forced shutdown because it no longer has a complete copy of all data stored in NDB.

- *node_id* RESTART [-n] [-i] [-a] [-f]

  Restarts the data node identified by *node_id* (or all data nodes).

  Using the -i option with RESTART causes the data node to perform an initial restart; that is, the node's file system is deleted and recreated. The effect is the same as that obtained from stopping the data node process and then starting it again using ndbd --initial from the system shell.

> **Note**
>
> Backup files and Disk Data files are not removed when this option is used.

Using the `-n` option causes the data node process to be restarted, but the data node is not actually brought online until the appropriate `START` command is issued. The effect of this option is the same as that obtained from stopping the data node and then starting it again using `ndbd --nostart` or `ndbd -n` from the system shell.

Using the `-a` causes all current transactions relying on this node to be aborted. No GCP check is done when the node rejoins the cluster.

Normally, `RESTART` fails if taking the node offline would result in an incomplete cluster. The `-f` option forces the node to restart without checking for this. If this option is used and the result is an incomplete cluster, the entire cluster is restarted.

- `node_id STATUS`

  Displays status information for the data node identified by *`node_id`* (or for all data nodes).

  The output from this command also indicates when the cluster is in single user mode.

- `node_id REPORT report-type`

  Displays a report of type *`report-type`* for the data node identified by *`node_id`*, or for all data nodes using `ALL`.

  Currently, there are three accepted values for *`report-type`*:

  - `BackupStatus` provides a status report on a cluster backup in progress

  - `MemoryUsage` displays how much data memory and index memory is being used by each data node as shown in this example:

    ```
    ndb_mgm> ALL REPORT MEMORY
    Node 1: Data usage is 5%(177 32K pages of total 3200)
    Node 1: Index usage is 0%(108 8K pages of total 12832)
    Node 2: Data usage is 5%(177 32K pages of total 3200)
    Node 2: Index usage is 0%(108 8K pages of total 12832)
    ```

    This information is also available from the `ndbinfo.memoryusage` table.

  - `EventLog` reports events from the event log buffers of one or more data nodes.

  `report-type` is case-insensitive and “fuzzy”; for `MemoryUsage`, you can use `MEMORY` (as shown in the prior example), `memory`, or even simply `MEM` (or `mem`). You can abbreviate `BackupStatus` in a similar fashion.

- `ENTER SINGLE USER MODE node_id`

  Enters single user mode, whereby only the MySQL server identified by the node ID *`node_id`* is permitted to access the database.

  Currently, it is not possible for data nodes to join a MySQL Cluster while it is running in single user mode. (Bug #20395)

- `EXIT SINGLE USER MODE`

  Exits single user mode, enabling all SQL nodes (that is, all running `mysqld` processes) to access the database.

> **Note**
>
> It is possible to use `EXIT SINGLE USER MODE` even when not in single user mode, although the command has no effect in this case.

- `QUIT`, `EXIT`

  Terminates the management client.

  This command does not affect any nodes connected to the cluster.

- `SHUTDOWN`

  Shuts down all cluster data nodes and management nodes. To exit the management client after this has been done, use `EXIT` or `QUIT`.

  This command does *not* shut down any SQL nodes or API nodes that are connected to the cluster.

- `CREATE NODEGROUP nodeid[, nodeid, ...]`

  Creates a new MySQL Cluster node group and causes data nodes to join it.

  This command is used after adding new data nodes online to a MySQL Cluster, and causes them to join a new node group and thus to begin participating fully in the cluster. The command takes as its sole parameter a comma-separated list of node IDs—these are the IDs of the nodes just added and started that are to join the new node group. The number of nodes must be the same as the number of nodes in each node group that is already part of the cluster (each MySQL Cluster node group must have the same number of nodes). In other words, if the MySQL Cluster has 2 node groups of 2 data nodes each, then the new node group must also have 2 data nodes.

  The node group ID of the new node group created by this command is determined automatically, and always the next highest unused node group ID in the cluster; it is not possible to set it manually.

  For more information, see Section 7.13, “Adding MySQL Cluster Data Nodes Online”.

- `DROP NODEGROUP nodegroup_id`

  Drops the MySQL Cluster node group with the given *`nodegroup_id`*.

  This command can be used to drop a node group from a MySQL Cluster. `DROP NODEGROUP` takes as its sole argument the node group ID of the node group to be dropped.

  `DROP NODEGROUP` acts only to remove the data nodes in the effected node group from that node group. It does not stop data nodes, assign them to a different node group, or remove them from the cluster's configuration. A data node that does not belong to a node group is indicated in the output of the management client `SHOW` command with `no nodegroup` in place of the node group ID, like this (indicated using bold text):

  ```
  id=3    @10.100.2.67  (5.6.27-ndb-7.4.9, no nodegroup)
  ```

  Prior to MySQL Cluster NDB 7.0.4, the `SHOW` output was not updated correctly following `DROP NODEGROUP`. (Bug #43413)

  `DROP NODEGROUP` works only when all data nodes in the node group to be dropped are completely empty of any table data and table definitions. Since there is currently no way using `ndb_mgm` or the `mysql` client to remove all data from a specific data node or node group, this means that the command succeeds only in the two following cases:

  1. After issuing `CREATE NODEGROUP` in the `ndb_mgm` client, but before issuing any `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statements in the `mysql` client.

2. After dropping all `NDBCLUSTER` tables using `DROP TABLE`.

   `TRUNCATE TABLE` does not work for this purpose because this removes only the table data; the data nodes continue to store an `NDBCLUSTER` table's definition until a `DROP TABLE` statement is issued that causes the table metadata to be dropped.

For more information about `DROP NODEGROUP`, see Section 7.13, "Adding MySQL Cluster Data Nodes Online".

7.3 Online Backup of MySQL Cluster

The next few sections describe how to prepare for and then to create a MySQL Cluster backup using the functionality for this purpose found in the `ndb_mgm` management client. To distinguish this type of backup from a backup made using `mysqldump`, we sometimes refer to it as a "native" MySQL Cluster backup. (For information about the creation of backups with `mysqldump`, see `mysqldump` — A Database Backup Program.) Restoration of MySQL Cluster backups is done using the `ndb_restore` utility provided with the MySQL Cluster distribution; for information about `ndb_restore` and its use in restoring MySQL Cluster backups, see Section 6.20, "`ndb_restore` — Restore a MySQL Cluster Backup".

7.3.1 MySQL Cluster Backup Concepts

A backup is a snapshot of the database at a given time. The backup consists of three main parts:

- **Metadata.** The names and definitions of all database tables
- **Table records.** The data actually stored in the database tables at the time that the backup was made
- **Transaction log.** A sequential record telling how and when data was stored in the database

Each of these parts is saved on all nodes participating in the backup. During backup, each node saves these three parts into three files on disk:

- `BACKUP-`*`backup_id`*`.`*`node_id`*`.ctl`

  A control file containing control information and metadata. Each node saves the same table definitions (for all tables in the cluster) to its own version of this file.

- `BACKUP-`*`backup_id`*`-0.`*`node_id`*`.data`

  A data file containing the table records, which are saved on a per-fragment basis. That is, different nodes save different fragments during the backup. The file saved by each node starts with a header that states the tables to which the records belong. Following the list of records there is a footer containing a checksum for all records.

- `BACKUP-`*`backup_id`*`.`*`node_id`*`.log`

  A log file containing records of committed transactions. Only transactions on tables stored in the backup are stored in the log. Nodes involved in the backup save different records because different nodes host different database fragments.

In the listing above, *`backup_id`* stands for the backup identifier and *`node_id`* is the unique identifier for the node creating the file.

7.3.2 Using The MySQL Cluster Management Client to Create a Backup

Before starting a backup, make sure that the cluster is properly configured for performing one. (See Section 7.3.3, "Configuration for MySQL Cluster Backups".)

The `START BACKUP` command is used to create a backup:

```
START BACKUP [backup_id] [wait_option] [snapshot_option]
wait_option:
WAIT {STARTED | COMPLETED} | NOWAIT
snapshot_option:
SNAPSHOTSTART | SNAPSHOTEND
```

Successive backups are automatically identified sequentially, so the *`backup_id`*, an integer greater than or equal to 1, is optional; if it is omitted, the next available value is used. If an existing *`backup_id`* value is used, the backup fails with the error `Backup failed: file already exists`. If used, the *`backup_id`* must follow `START BACKUP` immediately, before any other options are used.

The *`wait_option`* can be used to determine when control is returned to the management client after a `START BACKUP` command is issued, as shown in the following list:

- If `NOWAIT` is specified, the management client displays a prompt immediately, as seen here:

  ```
  ndb_mgm> START BACKUP NOWAIT
  ndb_mgm>
  ```

  In this case, the management client can be used even while it prints progress information from the backup process.

- With `WAIT STARTED` the management client waits until the backup has started before returning control to the user, as shown here:

  ```
  ndb_mgm> START BACKUP WAIT STARTED
  Waiting for started, this may take several minutes
  Node 2: Backup 3 started from node 1
  ndb_mgm>
  ```

- **`WAIT COMPLETED`** causes the management client to wait until the backup process is complete before returning control to the user.

`WAIT COMPLETED` is the default.

A *`snapshot_option`* can be used to determine whether the backup matches the state of the cluster when `START BACKUP` was issued, or when it was completed. `SNAPSHOTSTART` causes the backup to match the state of the cluster when the backup began; `SNAPSHOTEND` causes the backup to reflect the state of the cluster when the backup was finished. `SNAPSHOTEND` is the default, and matches the behavior found in previous MySQL Cluster releases.

> **Note**
>
> If you use the `SNAPSHOTSTART` option with `START BACKUP`, and the `CompressedBackup` parameter is enabled, only the data and control files are compressed—the log file is not compressed.

If both a *`wait_option`* and a *`snapshot_option`* are used, they may be specified in either order. For example, all of the following commands are valid, assuming that there is no existing backup having 4 as its ID:

```
START BACKUP WAIT STARTED SNAPSHOTSTART
START BACKUP SNAPSHOTSTART WAIT STARTED
START BACKUP 4 WAIT COMPLETED SNAPSHOTSTART
START BACKUP SNAPSHOTEND WAIT COMPLETED
START BACKUP 4 NOWAIT SNAPSHOTSTART
```

The procedure for creating a backup consists of the following steps:

1. Start the management client (ndb_mgm), if it not running already.

2. Execute the **START BACKUP** command. This produces several lines of output indicating the progress of the backup, as shown here:

```
ndb_mgm> START BACKUP
Waiting for completed, this may take several minutes
Node 2: Backup 1 started from node 1
Node 2: Backup 1 started from node 1 completed
 StartGCP: 177 StopGCP: 180
 #Records: 7362 #LogRecords: 0
 Data: 453648 bytes Log: 0 bytes
ndb_mgm>
```

3. When the backup has started the management client displays this message:

```
Backup backup_id started from node node_id
```

   backup_id is the unique identifier for this particular backup. This identifier is saved in the cluster log, if it has not been configured otherwise. *node_id* is the identifier of the management server that is coordinating the backup with the data nodes. At this point in the backup process the cluster has received and processed the backup request. It does not mean that the backup has finished. An example of this statement is shown here:

```
Node 2: Backup 1 started from node 1
```

4. The management client indicates with a message like this one that the backup has started:

```
Backup backup_id started from node node_id completed
```

   As is the case for the notification that the backup has started, *backup_id* is the unique identifier for this particular backup, and *node_id* is the node ID of the management server that is coordinating the backup with the data nodes. This output is accompanied by additional information including relevant global checkpoints, the number of records backed up, and the size of the data, as shown here:

```
Node 2: Backup 1 started from node 1 completed
 StartGCP: 177 StopGCP: 180
 #Records: 7362 #LogRecords: 0
 Data: 453648 bytes Log: 0 bytes
```

It is also possible to perform a backup from the system shell by invoking ndb_mgm with the -e or --execute option, as shown in this example:

```
shell> ndb_mgm -e "START BACKUP 6 WAIT COMPLETED SNAPSHOTSTART"
```

When using START BACKUP in this way, you must specify the backup ID.

Cluster backups are created by default in the BACKUP subdirectory of the DataDir on each data node. This can be overridden for one or more data nodes individually, or for all cluster data nodes in the config.ini file using the BackupDataDir configuration parameter. The backup files created for a backup with a given *backup_id* are stored in a subdirectory named BACKUP-*backup_id* in the backup directory.

To abort a backup that is already in progress:

1. Start the management client.

2. Execute this command:

```
ndb_mgm> ABORT BACKUP backup_id
```

The number *backup_id* is the identifier of the backup that was included in the response of the management client when the backup was started (in the message `Backup backup_id started from node management_node_id`).

3. The management client will acknowledge the abort request with `Abort of backup backup_id ordered`.

> **Note**
>
> At this point, the management client has not yet received a response from the cluster data nodes to this request, and the backup has not yet actually been aborted.

4. After the backup has been aborted, the management client will report this fact in a manner similar to what is shown here:

```
Node 1: Backup 3 started from 5 has been aborted.
  Error: 1321 - Backup aborted by user request: Permanent error: User defined error
Node 3: Backup 3 started from 5 has been aborted.
  Error: 1323 - 1323: Permanent error: Internal error
Node 2: Backup 3 started from 5 has been aborted.
  Error: 1323 - 1323: Permanent error: Internal error
Node 4: Backup 3 started from 5 has been aborted.
  Error: 1323 - 1323: Permanent error: Internal error
```

In this example, we have shown sample output for a cluster with 4 data nodes, where the sequence number of the backup to be aborted is `3`, and the management node to which the cluster management client is connected has the node ID `5`. The first node to complete its part in aborting the backup reports that the reason for the abort was due to a request by the user. (The remaining nodes report that the backup was aborted due to an unspecified internal error.)

> **Note**
>
> There is no guarantee that the cluster nodes respond to an `ABORT BACKUP` command in any particular order.

The `Backup backup_id started from node management_node_id has been aborted` messages mean that the backup has been terminated and that all files relating to this backup have been removed from the cluster file system.

It is also possible to abort a backup in progress from a system shell using this command:

```
shell> ndb_mgm -e "ABORT BACKUP backup_id"
```

> **Note**
>
> If there is no backup having the ID *backup_id* running when an `ABORT BACKUP` is issued, the management client makes no response, nor is it indicated in the cluster log that an invalid abort command was sent.

7.3.3 Configuration for MySQL Cluster Backups

Five configuration parameters are essential for backup:

- `BackupDataBufferSize`

  The amount of memory used to buffer data before it is written to disk.

- `BackupLogBufferSize`

  The amount of memory used to buffer log records before these are written to disk.

- `BackupMemory`

  The total memory allocated in a data node for backups. This should be the sum of the memory allocated for the backup data buffer and the backup log buffer.

- `BackupWriteSize`

  The default size of blocks written to disk. This applies for both the backup data buffer and the backup log buffer.

- `BackupMaxWriteSize`

  The maximum size of blocks written to disk. This applies for both the backup data buffer and the backup log buffer.

More detailed information about these parameters can be found in Backup Parameters.

7.3.4 MySQL Cluster Backup Troubleshooting

If an error code is returned when issuing a backup request, the most likely cause is insufficient memory or disk space. You should check that there is enough memory allocated for the backup.

> **Important**
>
> If you have set `BackupDataBufferSize` and `BackupLogBufferSize` and their sum is greater than 4MB, then you must also set `BackupMemory` as well.

You should also make sure that there is sufficient space on the hard drive partition of the backup target.

`NDB` does not support repeatable reads, which can cause problems with the restoration process. Although the backup process is “hot”, restoring a MySQL Cluster from backup is not a 100% “hot” process. This is due to the fact that, for the duration of the restore process, running transactions get nonrepeatable reads from the restored data. This means that the state of the data is inconsistent while the restore is in progress.

7.4 MySQL Server Usage for MySQL Cluster

`mysqld` is the traditional MySQL server process. To be used with MySQL Cluster, `mysqld` needs to be built with support for the `NDB` storage engine, as it is in the precompiled binaries available from http://dev.mysql.com/downloads/. If you build MySQL from source, you must invoke `CMake` with the `-DWITH_NDBCLUSTER=1` option to include support for `NDB`.

For more information about compiling MySQL Cluster from source, see Section 4.2.3, “Building MySQL Cluster from Source on Linux”, and Section 4.3.2, “Compiling and Installing MySQL Cluster from Source on Windows”.

(For information about `mysqld` options and variables, in addition to those discussed in this section, which are relevant to MySQL Cluster, see Section 5.3.8, “MySQL Server Options and Variables for MySQL Cluster”.)

If the `mysqld` binary has been built with Cluster support, the `NDBCLUSTER` storage engine is still disabled by default. You can use either of two possible options to enable this engine:

- Use `--ndbcluster` as a startup option on the command line when starting `mysqld`.

- Insert a line containing `ndbcluster` in the `[mysqld]` section of your `my.cnf` file.

An easy way to verify that your server is running with the `NDBCLUSTER` storage engine enabled is to issue the `SHOW ENGINES` statement in the MySQL Monitor (`mysql`). You should see the value `YES` as the `Support` value in the row for `NDBCLUSTER`. If you see `NO` in this row or if there is no such row displayed in the output, you are not running an `NDB`-enabled version of MySQL. If you see `DISABLED` in this row, you need to enable it in either one of the two ways just described.

To read cluster configuration data, the MySQL server requires at a minimum three pieces of information:

- The MySQL server's own cluster node ID
- The host name or IP address for the management server (MGM node)
- The number of the TCP/IP port on which it can connect to the management server

Node IDs can be allocated dynamically, so it is not strictly necessary to specify them explicitly.

The `mysqld` parameter `ndb-connectstring` is used to specify the connection string either on the command line when starting `mysqld` or in `my.cnf`. The connection string contains the host name or IP address where the management server can be found, as well as the TCP/IP port it uses.

In the following example, `ndb_mgmd.mysql.com` is the host where the management server resides, and the management server listens for cluster messages on port 1186:

```
shell> mysqld --ndbcluster --ndb-connectstring=ndb_mgmd.mysql.com:1186
```

See Section 5.3.3, “MySQL Cluster Connection Strings”, for more information on connection strings.

Given this information, the MySQL server will be a full participant in the cluster. (We often refer to a `mysqld` process running in this manner as an SQL node.) It will be fully aware of all cluster data nodes as well as their status, and will establish connections to all data nodes. In this case, it is able to use any data node as a transaction coordinator and to read and update node data.

You can see in the `mysql` client whether a MySQL server is connected to the cluster using `SHOW PROCESSLIST`. If the MySQL server is connected to the cluster, and you have the `PROCESS` privilege, then the first row of the output is as shown here:

```
mysql> SHOW PROCESSLIST \G
*************************** 1. row ***************************
     Id: 1
   User: system user
   Host:
     db:
Command: Daemon
   Time: 1
  State: Waiting for event from ndbcluster
   Info: NULL
```

Important

To participate in a MySQL Cluster, the `mysqld` process must be started with *both* the options `--ndbcluster` and `--ndb-connectstring` (or their equivalents in `my.cnf`). If `mysqld` is started with only the `--ndbcluster` option, or if it is unable to contact the cluster, it is not possible to work with `NDB` tables, *nor is it possible to create any new tables regardless of storage engine*. The latter restriction is a safety measure intended to prevent the creation of tables having the same names as `NDB` tables while the SQL node is not connected to the cluster. If you wish to create tables using a different storage engine while the mysqld process is not participating in a MySQL Cluster, you must restart the server *without* the --ndbcluster option.

7.5 Performing a Rolling Restart of a MySQL Cluster

This section discusses how to perform a *rolling restart* of a MySQL Cluster installation, so called because it involves stopping and starting (or restarting) each node in turn, so that the cluster itself remains operational. This is often done as part of a *rolling upgrade* or *rolling downgrade*, where high availability of the cluster is mandatory and no downtime of the cluster as a whole is permissible. Where we refer to upgrades, the information provided here also generally applies to downgrades as well.

There are a number of reasons why a rolling restart might be desirable. These are described in the next few paragraphs.

Configuration change.
To make a change in the cluster's configuration, such as adding an SQL node to the cluster, or setting a configuration parameter to a new value.

MySQL Cluster software upgrade or downgrade. To upgrade the cluster to a newer version of the MySQL Cluster software (or to downgrade it to an older version). This is usually referred to as a "rolling upgrade" (or "rolling downgrade", when reverting to an older version of MySQL Cluster).

Change on node host. To make changes in the hardware or operating system on which one or more MySQL Cluster node processes are running.

System reset (cluster reset).
To reset the cluster because it has reached an undesirable state. In such cases it is often desirable to reload the data and metadata of one or more data nodes. This can be done in any of three ways:

- Start each data node process (ndbd or possibly ndbmtd) with the --initial option, which forces the data node to clear its file system and to reload all MySQL Cluster data and metadata from the other data nodes.

- Create a backup using the ndb_mgm client BACKUP command prior to performing the restart. Following the upgrade, restore the node or nodes using ndb_restore.

  See Section 7.3, "Online Backup of MySQL Cluster", and Section 6.20, "ndb_restore — Restore a MySQL Cluster Backup", for more information.

- Use mysqldump to create a backup prior to the upgrade; afterward, restore the dump using LOAD DATA INFILE.

Resource Recovery.
To free memory previously allocated to a table by successive INSERT and DELETE operations, for re-use by other MySQL Cluster tables.

The process for performing a rolling restart may be generalized as follows:

1. Stop all cluster management nodes (ndb_mgmd processes), reconfigure them, then restart them. (See Rolling restarts with multiple management servers.)

2. Stop, reconfigure, then restart each cluster data node (ndbd process) in turn.

3. Stop, reconfigure, then restart each cluster SQL node (mysqld process) in turn.

The specifics for implementing a given rolling upgrade depend upon the changes being made. A more detailed view of the process is presented here:

Figure 7.1 MySQL Cluster Rolling Restarts By Type

<table>
<tr><th colspan="6">RESTART TYPE:</th></tr>
<tr><th colspan="2">Cluster Configuration Change</th><th>Cluster Software Upgrade or Downgrade</th><th>Change on Node Host</th><th colspan="2">Cluster Reset</th></tr>
<tr><td colspan="6">A. Management node (ndb_mgmd) processes...</td></tr>
<tr><td colspan="2" rowspan="2">1. Stop all ndb_mgmd processes
2. Make changes in global configuration file(s)
3. Start all ndb_mgmd processes</td><td rowspan="2">1. Stop all ndb_mgmd processes
2. Replace each ndb_mgmd binary with new version
3. Start ndb_mgmd processes</td><td rowspan="2">1. Stop all ndb_mgmd processes
2. Make desired changes in hardware, operating system, or both
3. Start all ndb_mgmd processes</td><td colspan="2">(OR)</td></tr>
<tr><td>1. Stop all ndb_mgmd processes
2. Start all ndb_mgmd processes</td><td>Restart all ndb_mgmd processes (optional)</td></tr>
<tr><td colspan="6">B. For each data node (ndbd) process...</td></tr>
<tr><td colspan="2">(OR)</td><td rowspan="2">1. Stop ndbd
2. Replace ndbd binary with new version
3. Start ndbd</td><td rowspan="2">1. Stop ndbd
2. Make desired changes in hardware, operating system, or both
3. Start ndbd</td><td colspan="2">(OR)</td></tr>
<tr><td>1. Stop ndbd
2. Start ndbd</td><td>Restart ndbd</td><td>1. Stop ndbd
2. Start ndbd</td><td>Restart ndbd</td></tr>
<tr><td colspan="6">C. For each SQL node (mysqld) process...</td></tr>
<tr><td colspan="2">(OR)</td><td rowspan="2">1. Stop mysqld
2. Replace mysqld binary with new version
3. Start mysqld</td><td rowspan="2">1. Stop mysqld
2. Make desired changes in hardware, operating system, or both
3. Start mysqld</td><td colspan="2">(OR)</td></tr>
<tr><td>1. Stop mysqld
2. Start mysqld</td><td>Restart mysqld</td><td>1. Stop mysqld
2. Start mysqld</td><td>Restart mysqld</td></tr>
</table>

In the previous diagram, the **Stop** and **Start** steps indicate that the process must be stopped completely using a shell command (such as `kill` on most Unix systems) or the management client `STOP` command, then started again from a system shell by invoking the `ndbd` or `ndb_mgmd` executable as appropriate. On Windows, you can also use the system `NET START` and `NET STOP` commands or the Windows Service Manager to start and stop nodes which have been installed as Windows services (see Section 4.3.4, "Installing MySQL Cluster Processes as Windows Services").

Restart indicates that the process may be restarted using the `ndb_mgm` management client `RESTART` command (see Section 7.2, "Commands in the MySQL Cluster Management Client").

MySQL Cluster supports a flexible order for upgrading nodes. When upgrading a MySQL Cluster, you may upgrade API nodes (including SQL nodes) before upgrading the management nodes, data nodes, or both. In other words, you are permitted to upgrade the API and SQL nodes in any order. This is subject to the following provisions:

- This functionality is intended for use as part of an online upgrade only. A mix of node binaries from different MySQL Cluster releases is neither intended nor supported for continuous, long-term use in a production setting.

- All management nodes must be upgraded before any data nodes are upgraded. This remains true regardless of the order in which you upgrade the cluster's API and SQL nodes.

- Features specific to the “new” version must not be used until all management nodes and data nodes have been upgraded.

  This also applies to any MySQL Server version change that may apply, in addition to the NDB engine version change, so do not forget to take this into account when planning the upgrade. (This is true for online upgrades of MySQL Cluster in general.)

See also Bug #48528 and Bug #49163.

Rolling restarts with multiple management servers. When performing a rolling restart of a MySQL Cluster with multiple management nodes, you should keep in mind that `ndb_mgmd` checks to see if any other management node is running, and, if so, tries to use that node's configuration data. To keep this from occurring, and to force `ndb_mgmd` to reread its configuration file, perform the following steps:

1. Stop all MySQL Cluster `ndb_mgmd` processes.

2. Update all `config.ini` files.

3. Start a single `ndb_mgmd` with `--reload`, `--initial`, or both options as desired.

4. If you started the first `ndb_mgmd` with the `--initial` option, you must also start any remaining `ndb_mgmd` processes using `--initial`.

   Regardless of any other options used when starting the first `ndb_mgmd`, you should not start any remaining `ndb_mgmd` processes after the first one using `--reload`.

5. Complete the rolling restarts of the data nodes and API nodes as normal.

When performing a rolling restart to update the cluster's configuration, you can use the `config_generation` column of the `ndbinfo.nodes` table to keep track of which data nodes have been successfully restarted with the new configuration. See Section 7.10.18, “The ndbinfo nodes Table”.

7.6 Event Reports Generated in MySQL Cluster

In this section, we discuss the types of event logs provided by MySQL Cluster, and the types of events that are logged.

MySQL Cluster provides two types of event log:

- The *cluster log*, which includes events generated by all cluster nodes. The cluster log is the log recommended for most uses because it provides logging information for an entire cluster in a single location.

  By default, the cluster log is saved to a file named `ndb_node_id_cluster.log`, (where *`node_id`* is the node ID of the management server) in the management server's `DataDir`.

  Cluster logging information can also be sent to `stdout` or a `syslog` facility in addition to or instead of being saved to a file, as determined by the values set for the `DataDir` and `LogDestination` configuration parameters. See Section 5.3.5, “Defining a MySQL Cluster Management Server”, for more information about these parameters.

- *Node logs* are local to each node.

  Output generated by node event logging is written to the file `ndb_node_id_out.log` (where *`node_id`* is the node's node ID) in the node's `DataDir`. Node event logs are generated for both management nodes and data nodes.

  Node logs are intended to be used only during application development, or for debugging application code.

Both types of event logs can be set to log different subsets of events.

Each reportable event can be distinguished according to three different criteria:

- *Category*: This can be any one of the following values: `STARTUP`, `SHUTDOWN`, `STATISTICS`, `CHECKPOINT`, `NODERESTART`, `CONNECTION`, `ERROR`, or `INFO`.
- *Priority*: This is represented by one of the numbers from 0 to 15 inclusive, where 0 indicates “most important” and 15 “least important.”
- *Severity Level*: This can be any one of the following values: `ALERT`, `CRITICAL`, `ERROR`, `WARNING`, `INFO`, or `DEBUG`.

Both the cluster log and the node log can be filtered on these properties.

The format used in the cluster log is as shown here:

```
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 1: Data usage is 2%(60 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 1: Index usage is 1%(24 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 1: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 2: Data usage is 2%(76 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 2: Index usage is 1%(24 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 2: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 3: Data usage is 2%(58 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 3: Index usage is 1%(25 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 3: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 4: Data usage is 2%(74 32K pages of total 2560)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 4: Index usage is 1%(25 8K pages of total 2336)
2007-01-26 19:35:55 [MgmSrvr] INFO     -- Node 4: Resource 0 min: 0 max: 639 curr: 0
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 4: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 1: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 1: Node 9: API 5.6.27-ndb-7.4.9
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 2: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 2: Node 9: API 5.6.27-ndb-7.4.9
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 3: Node 9 Connected
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 3: Node 9: API 5.6.27-ndb-7.4.9
2007-01-26 19:39:42 [MgmSrvr] INFO     -- Node 4: Node 9: API 5.6.27-ndb-7.4.9
2007-01-26 19:59:22 [MgmSrvr] ALERT    -- Node 2: Node 7 Disconnected
2007-01-26 19:59:22 [MgmSrvr] ALERT    -- Node 2: Node 7 Disconnected
```

Each line in the cluster log contains the following information:

- A timestamp in *`YYYY-MM-DD HH:MM:SS`* format.
- The type of node which is performing the logging. In the cluster log, this is always `[MgmSrvr]`.
- The severity of the event.
- The ID of the node reporting the event.
- A description of the event. The most common types of events to appear in the log are connections and disconnections between different nodes in the cluster, and when checkpoints occur. In some cases, the description may contain status information.

7.6.1 MySQL Cluster Logging Management Commands

`ndb_mgm` supports a number of management commands related to the cluster log. In the listing that follows, *`node_id`* denotes either a database node ID or the keyword `ALL`, which indicates that the command should be applied to all of the cluster's data nodes.

- `CLUSTERLOG ON`

  Turns the cluster log on.

- `CLUSTERLOG OFF`

  Turns the cluster log off.

- `CLUSTERLOG INFO`

  Provides information about cluster log settings.

- *`node_id`* `CLUSTERLOG` *`category=threshold`*

  Logs *`category`* events with priority less than or equal to *`threshold`* in the cluster log.

- `CLUSTERLOG FILTER` *`severity_level`*

  Toggles cluster logging of events of the specified *`severity_level`*.

The following table describes the default setting (for all data nodes) of the cluster log category threshold. If an event has a priority with a value lower than or equal to the priority threshold, it is reported in the cluster log.

> **Note**
>
> Events are reported per data node, and that the threshold can be set to different values on different nodes.

Category	Default threshold (All data nodes)
`STARTUP`	**7**
`SHUTDOWN`	**7**
`STATISTICS`	**7**
`CHECKPOINT`	**7**
`NODERESTART`	**7**
`CONNECTION`	**7**
`ERROR`	**15**
`INFO`	**7**

The `STATISTICS` category can provide a great deal of useful data. See Section 7.6.3, “Using CLUSTERLOG STATISTICS in the MySQL Cluster Management Client”, for more information.

Thresholds are used to filter events within each category. For example, a `STARTUP` event with a priority of 3 is not logged unless the threshold for `STARTUP` is set to 3 or higher. Only events with priority 3 or lower are sent if the threshold is 3.

The following table shows the event severity levels.

> **Note**
>
> These correspond to Unix `syslog` levels, except for `LOG_EMERG` and `LOG_NOTICE`, which are not used or mapped.

Severity Level Value	Severity	Description
1	ALERT	A condition that should be corrected immediately, such as a corrupted system database
2	CRITICAL	Critical conditions, such as device errors or insufficient resources
3	ERROR	Conditions that should be corrected, such as configuration errors
4	WARNING	Conditions that are not errors, but that might require special handling
5	INFO	Informational messages
6	DEBUG	Debugging messages used for NDBCLUSTER development

Event severity levels can be turned on or off (using CLUSTERLOG FILTER—see above). If a severity level is turned on, then all events with a priority less than or equal to the category thresholds are logged. If the severity level is turned off then no events belonging to that severity level are logged.

Important

Cluster log levels are set on a per ndb_mgmd, per subscriber basis. This means that, in a MySQL Cluster with multiple management servers, using a CLUSTERLOG command in an instance of ndb_mgm connected to one management server affects only logs generated by that management server but not by any of the others. This also means that, should one of the management servers be restarted, only logs generated by that management server are affected by the resetting of log levels caused by the restart.

7.6.2 MySQL Cluster Log Events

An event report reported in the event logs has the following format:

```
datetime [string] severity -- message
```

For example:

```
09:19:30 2005-07-24 [NDB] INFO -- Node 4 Start phase 4 completed
```

This section discusses all reportable events, ordered by category and severity level within each category.

In the event descriptions, GCP and LCP mean “Global Checkpoint” and “Local Checkpoint”, respectively.

CONNECTION Events

These events are associated with connections between Cluster nodes.

Event	Priority	Severity Level	Description
Connected	8	INFO	Data nodes connected
Disconnected	8	ALERT	Data nodes disconnected
CommunicationClosed	8	INFO	SQL node or data node connection closed
CommunicationOpened	8	INFO	SQL node or data node connection open
ConnectedApiVersion	8	INFO	Connection using API version

CHECKPOINT Events

The logging messages shown here are associated with checkpoints.

Event	Priority	Severity Level	Description
GlobalCheckpointStarted	9	INFO	Start of GCP: REDO log is written to disk
GlobalCheckpointCompleted	10	INFO	GCP finished
LocalCheckpointStarted	7	INFO	Start of LCP: data written to disk
LocalCheckpointCompleted	7	INFO	LCP completed normally
LCPStoppedInCalcKeepGci	0	ALERT	LCP stopped
LCPFragmentCompleted	11	INFO	LCP on a fragment has been completed
UndoLogBlocked	7	INFO	UNDO logging blocked; buffer near overflow
RedoStatus	7	INFO	Redo status

STARTUP Events

The following events are generated in response to the startup of a node or of the cluster and of its success or failure. They also provide information relating to the progress of the startup process, including information concerning logging activities.

Event	Priority	Severity Level	Description
NDBStartStarted	1	INFO	Data node start phases initiated (all nodes starting)
NDBStartCompleted	1	INFO	Start phases completed, all data nodes
STTORRYRecieved	15	INFO	Blocks received after completion of restart
StartPhaseCompleted	4	INFO	Data node start phase *X* completed
CM_REGCONF	3	INFO	Node has been successfully included into the cluster; shows the node, managing node, and dynamic ID
CM_REGREF	8	INFO	Node has been refused for inclusion in the cluster; cannot be included in cluster due to misconfiguration, inability to establish communication, or other problem
FIND_NEIGHBOURS	8	INFO	Shows neighboring data nodes
NDBStopStarted	1	INFO	Data node shutdown initiated
NDBStopCompleted	1	INFO	Data node shutdown complete
NDBStopForced	1	ALERT	Forced shutdown of data node
NDBStopAborted	1	INFO	Unable to shut down data node normally
StartREDOLog	4	INFO	New redo log started; GCI keep *X*, newest restorable GCI *Y*
StartLog	10	INFO	New log started; log part *X*, start MB *Y*, stop MB *Z*
UNDORecordsExecuted	15	INFO	Undo records executed
StartReport	4	INFO	Report started
LogFileInitStatus	7	INFO	Log file initialization status
LogFileInitCompStatus	7	INFO	Log file completion status
StartReadLCP	10	INFO	Start read for local checkpoint

Event	Priority	Severity Level	Description
ReadLCPComplete	10	INFO	Read for local checkpoint completed
RunRedo	8	INFO	Running the redo log
RebuildIndex	10	INFO	Rebuilding indexes

NODERESTART Events

The following events are generated when restarting a node and relate to the success or failure of the node restart process.

Event	Priority	Severity Level	Description
NR_CopyDict	7	INFO	Completed copying of dictionary information
NR_CopyDistr	7	INFO	Completed copying distribution information
NR_CopyFragsStarted	7	INFO	Starting to copy fragments
NR_CopyFragDone	10	INFO	Completed copying a fragment
NR_CopyFragsCompleted	7	INFO	Completed copying all fragments
NodeFailCompleted	8	ALERT	Node failure phase completed
NODE_FAILREP	8	ALERT	Reports that a node has failed
ArbitState	6	INFO	Report whether an arbitrator is found or not; there are seven different possible outcomes when seeking an arbitrator, listed here: • Management server restarts arbitration thread [state=*X*] • Prepare arbitrator node *X* [ticket=*Y*] • Receive arbitrator node *X* [ticket=*Y*] • Started arbitrator node *X* [ticket=*Y*] • Lost arbitrator node *X* - process failure [state=*Y*] • Lost arbitrator node *X* - process exit [state=*Y*] • Lost arbitrator node *X* <error msg> [state=*Y*]
ArbitResult	2	ALERT	Report arbitrator results; there are eight different possible results for arbitration attempts, listed here: • Arbitration check failed: less than 1/2 nodes left • Arbitration check succeeded: node group majority • Arbitration check failed: missing node group • Network partitioning: arbitration required

Event	Priority	Severity Level	Description
			• Arbitration succeeded: affirmative response from node *X* • Arbitration failed: negative response from node *X* • Network partitioning: no arbitrator available • Network partitioning: no arbitrator configured
GCP_TakeoverStarted	7	INFO	GCP takeover started
GCP_TakeoverCompleted	7	INFO	GCP takeover complete
LCP_TakeoverStarted	7	INFO	LCP takeover started
LCP_TakeoverCompleted	7	INFO	LCP takeover complete (state = *X*)
ConnectCheckStarted	6	INFO	Connection check started
ConnectCheckCompleted	6	INFO	Connection check completed
NodeFailRejected	6	ALERT	Node failure phase failed

STATISTICS Events

The following events are of a statistical nature. They provide information such as numbers of transactions and other operations, amount of data sent or received by individual nodes, and memory usage.

Event	Priority	Severity Level	Description
TransReportCounters	8	INFO	Report transaction statistics, including numbers of transactions, commits, reads, simple reads, writes, concurrent operations, attribute information, and aborts
OperationReportCounters	8	INFO	Number of operations
TableCreated	7	INFO	Report number of tables created
JobStatistic	9	INFO	Mean internal job scheduling statistics
ThreadConfigLoop	9	INFO	Number of thread configuration loops
SendBytesStatistic	9	INFO	Mean number of bytes sent to node *X*
ReceiveBytesStatistic	9	INFO	Mean number of bytes received from node *X*
MemoryUsage	5	INFO	Data and index memory usage (80%, 90%, and 100%)
MTSignalStatistics	9	INFO	Multi-threaded signals

SCHEMA Events

These events relate to MySQL Cluster schema operations.

Event	Priority	Severity	Description
CreateSchemaObject	8	INFO	Schema objected created
AlterSchemaObject	8	INFO	Schema object updated
DropSchemaObject	8	INFO	Schema object dropped

ERROR Events

These events relate to Cluster errors and warnings. The presence of one or more of these generally indicates that a major malfunction or failure has occurred.

Event	Priority	Severity	Description
TransporterError	2	ERROR	Transporter error
TransporterWarning	8	WARNING	Transporter warning
MissedHeartbeat	8	WARNING	Node *X* missed heartbeat number *Y*
DeadDueToHeartbeat	8	ALERT	Node *X* declared “dead” due to missed heartbeat
WarningEvent	2	WARNING	General warning event
SubscriptionStatus	4	WARNING	Change in subscription status

INFO Events

These events provide general information about the state of the cluster and activities associated with Cluster maintenance, such as logging and heartbeat transmission.

Event	Priority	Severity	Description
SentHeartbeat	12	INFO	Sent heartbeat
CreateLogBytes	11	INFO	Create log: Log part, log file, size in MB
InfoEvent	2	INFO	General informational event
EventBufferStatus	7	INFO	Event buffer status

Note

SentHeartbeat events are available only if MySQL Cluster was compiled with VM_TRACE enabled.

SINGLEUSER Events

These events are associated with entering and exiting single user mode.

Event	Priority	Severity	Description
SingleUser	7	INFO	Entering or exiting single user mode

BACKUP Events

These events provide information about backups being created or restored.

Event	Priority	Severity	Description
BackupStarted	7	INFO	Backup started
BackupStatus	7	INFO	Backup status
BackupCompleted	7	INFO	Backup completed
BackupFailedToStart	7	ALERT	Backup failed to start
BackupAborted	7	ALERT	Backup aborted by user
RestoreStarted	7	INFO	Started restoring from backup
RestoreMetaData	7	INFO	Restoring metadata
RestoreData	7	INFO	Restoring data
RestoreLog	7	INFO	Restoring log files
RestoreCompleted	7	INFO	Completed restoring from backup
SavedEvent	7	INFO	Event saved

7.6.3 Using CLUSTERLOG STATISTICS in the MySQL Cluster Management Client

The `NDB` management client's `CLUSTERLOG STATISTICS` command can provide a number of useful statistics in its output. Counters providing information about the state of the cluster are updated at 5-second reporting intervals by the transaction coordinator (TC) and the local query handler (LQH), and written to the cluster log.

Transaction coordinator statistics. Each transaction has one transaction coordinator, which is chosen by one of the following methods:

- In a round-robin fashion
- By communication proximity
- (*Beginning with MySQL Cluster NDB 6.3.4*:) By supplying a data placement hint when the transaction is started

> **Note**
>
> You can determine which TC selection method is used for transactions started from a given SQL node using the `ndb_optimized_node_selection` system variable.

All operations within the same transaction use the same transaction coordinator, which reports the following statistics:

- **Trans count.** This is the number transactions started in the last interval using this TC as the transaction coordinator. Any of these transactions may have committed, have been aborted, or remain uncommitted at the end of the reporting interval.

  > **Note**
  >
  > Transactions do not migrate between TCs.

- **Commit count.** This is the number of transactions using this TC as the transaction coordinator that were committed in the last reporting interval. Because some transactions committed in this reporting interval may have started in a previous reporting interval, it is possible for `Commit count` to be greater than `Trans count`.
- **Read count.** This is the number of primary key read operations using this TC as the transaction coordinator that were started in the last reporting interval, including simple reads. This count also includes reads performed as part of unique index operations. A unique index read operation generates 2 primary key read operations—1 for the hidden unique index table, and 1 for the table on which the read takes place.
- **Simple read count.** This is the number of simple read operations using this TC as the transaction coordinator that were started in the last reporting interval.
- **Write count.** This is the number of primary key write operations using this TC as the transaction coordinator that were started in the last reporting interval. This includes all inserts, updates, writes and deletes, as well as writes performed as part of unique index operations.

  > **Note**
  >
  > A unique index update operation can generate multiple PK read and write operations on the index table and on the base table.

- **AttrInfoCount.** This is the number of 32-bit data words received in the last reporting interval for primary key operations using this TC as the transaction coordinator. For reads, this is proportional to the number of columns requested. For inserts and updates, this is proportional to the number of columns written, and the size of their data. For delete operations, this is usually zero.

Unique index operations generate multiple PK operations and so increase this count. However, data words sent to describe the PK operation itself, and the key information sent, are *not* counted here. Attribute information sent to describe columns to read for scans, or to describe ScanFilters, is also not counted in `AttrInfoCount`.

- **Concurrent Operations.** This is the number of primary key or scan operations using this TC as the transaction coordinator that were started during the last reporting interval but that were not completed. Operations increment this counter when they are started and decrement it when they are completed; this occurs after the transaction commits. Dirty reads and writes—as well as failed operations—decrement this counter.

  The maximum value that `Concurrent Operations` can have is the maximum number of operations that a TC block can support; currently, this is `(2 * MaxNoOfConcurrentOperations) + 16 + MaxNoOfConcurrentTransactions`. (For more information about these configuration parameters, see the *Transaction Parameters* section of Section 5.3.6, “Defining MySQL Cluster Data Nodes”.)

- **Abort count.** This is the number of transactions using this TC as the transaction coordinator that were aborted during the last reporting interval. Because some transactions that were aborted in the last reporting interval may have started in a previous reporting interval, `Abort count` can sometimes be greater than `Trans count`.

- **Scans.** This is the number of table scans using this TC as the transaction coordinator that were started during the last reporting interval. This does not include range scans (that is, ordered index scans).

- **Range scans.** This is the number of ordered index scans using this TC as the transaction coordinator that were started in the last reporting interval.

Local query handler statistics (Operations). There is 1 cluster event per local query handler block (that is, 1 per data node process). Operations are recorded in the LQH where the data they are operating on resides.

> **Note**
>
> A single transaction may operate on data stored in multiple LQH blocks.

The `Operations` statistic provides the number of local operations performed by this LQH block in the last reporting interval, and includes all types of read and write operations (insert, update, write, and delete operations). This also includes operations used to replicate writes. For example, in a 2-replica cluster, the write to the primary replica is recorded in the primary LQH, and the write to the backup will be recorded in the backup LQH. Unique key operations may result in multiple local operations; however, this does *not* include local operations generated as a result of a table scan or ordered index scan, which are not counted.

Process scheduler statistics. In addition to the statistics reported by the transaction coordinator and local query handler, each `ndbd` process has a scheduler which also provides useful metrics relating to the performance of a MySQL Cluster. This scheduler runs in an infinite loop; during each loop the scheduler performs the following tasks:

1. Read any incoming messages from sockets into a job buffer.
2. Check whether there are any timed messages to be executed; if so, put these into the job buffer as well.
3. Execute (in a loop) any messages in the job buffer.
4. Send any distributed messages that were generated by executing the messages in the job buffer.
5. Wait for any new incoming messages.

Process scheduler statistics include the following:

- **Mean Loop Counter.** This is the number of loops executed in the third step from the preceding list. This statistic increases in size as the utilization of the TCP/IP buffer improves. You can use this to monitor changes in performance as you add new data node processes.
- **Mean send size and Mean receive size.** These statistics enable you to gauge the efficiency of, respectively writes and reads between nodes. The values are given in bytes. Higher values mean a lower cost per byte sent or received; the maximum value is 64K.

To cause all cluster log statistics to be logged, you can use the following command in the `NDB` management client:

```
ndb_mgm> ALL CLUSTERLOG STATISTICS=15
```

> **Note**
>
> Setting the threshold for `STATISTICS` to 15 causes the cluster log to become very verbose, and to grow quite rapidly in size, in direct proportion to the number of cluster nodes and the amount of activity in the MySQL Cluster.

For more information about MySQL Cluster management client commands relating to logging and reporting, see Section 7.6.1, "MySQL Cluster Logging Management Commands".

7.7 MySQL Cluster Log Messages

This section contains information about the messages written to the cluster log in response to different cluster log events. It provides additional, more specific information on `NDB` transporter errors.

7.7.1 MySQL Cluster: Messages in the Cluster Log

The following table lists the most common `NDB` cluster log messages. For information about the cluster log, log events, and event types, see Section 7.6, "Event Reports Generated in MySQL Cluster". These log messages also correspond to log event types in the MGM API; see The `Ndb_logevent_type` Type, for related information of interest to Cluster API developers.

Log Message	Description	Event Name	Event Type	Priority	Severity
`Node` *`mgm_node_id`*`: Node` *`data_node_id`* `Connected`	The data node having node ID *`node_id`* has connected to the management server (node *`mgm_node_id`*).	`Connected`	`Connection`	8	`INFO`
`Node` *`mgm_node_id`*`: Node` *`data_node_id`* `Disconnected`	The data node having node ID *`data_node_id`* has disconnected from the management server (node *`mgm_node_id`*).	`Disconnected`	`Connection`	8	`ALERT`
`Node` *`data_node_id`*`: Communication to Node` *`api_node_id`* `closed`	The API node or SQL node having node ID *`api_node_id`* is no longer communicating with data node *`data_node_id`*.	`CommunicationClosed`	`Connection`	8	`INFO`

Log Message	Description	Event Name	Event Type	Priority	Severity
Node *data_node_id*: Communication to Node *api_node_id* opened	The API node or SQL node having node ID *api_node_id* is now communicating with data node *data_node_id*.	CommunicationOpened	Connection	8	INFO
Node *mgm_node_id*: Node *api_node_id*: API *version*	The API node having node ID *api_node_id* has connected to management node *mgm_node_id* using NDB API version *version* (generally the same as the MySQL version number).	ConnectedApiVersion	Connection	8	INFO
Node *node_id*: Global checkpoint *gci* started	A global checkpoint with the ID *gci* has been started; node *node_id* is the master responsible for this global checkpoint.	GlobalCheckpointStarted	Checkpoint	9	INFO
Node *node_id*: Global checkpoint *gci* completed	The global checkpoint having the ID *gci* has been completed; node *node_id* was the master responsible for this global checkpoint.	GlobalCheckpointCompleted	Checkpoint	10	INFO
Node *node_id*: Local checkpoint *lcp* started. Keep GCI = *current_gci* oldest restorable GCI = *old_gci*	The local checkpoint having sequence ID *lcp* has been started on node *node_id*. The most recent GCI that can be used has the index *current_gci*, and the oldest GCI from which the cluster can be restored has the index *old_gci*.	LocalCheckpointStarted	Checkpoint	7	INFO
Node *node_id*: Local checkpoint *lcp* completed	The local checkpoint having sequence ID *lcp* on node	LocalCheckpointCompleted	Checkpoint	8	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
	node_id has been completed.				
Node *node_id*: Local Checkpoint stopped in CALCULATED_KEEP_GCI	The node was unable to determine the most recent usable GCI.	LCPStoppedInCalcKeepGci	Checkpoint	0	ALERT
Node *node_id*: Table ID = *table_id*, fragment ID = *fragment_id* has completed LCP on Node *node_id* maxGciStarted: *started_gci* maxGciCompleted: *completed_gci*	A table fragment has been checkpointed to disk on node *node_id*. The GCI in progress has the index *started_gci*, and the most recent GCI to have been completed has the index *completed_gci*.	LCPFragmentCompleted	Checkpoint	11	INFO
Node *node_id*: ACC Blocked *num_1* and TUP Blocked *num_2* times last second	Undo logging is blocked because the log buffer is close to overflowing.	UndoLogBlocked	Checkpoint	7	INFO
Node *node_id*: Start initiated *version*	Data node *node_id*, running NDB version *version*, is beginning its startup process.	NDBStartStarted	StartUp	1	INFO
Node *node_id*: Started *version*	Data node *node_id*, running NDB version *version*, has started successfully.	NDBStartCompleted	StartUp	1	INFO
Node *node_id*: STTORRY received after restart finished	The node has received a signal indicating that a cluster restart has completed.	STTORRYRecieved	StartUp	15	INFO
Node *node_id*: Start phase *phase* completed (*type*)	The node has completed start phase *phase* of a *type* start. For a listing of start phases, see Section 7.1, “Summary of MySQL Cluster Start Phases”. (*type* is one	StartPhaseCompleted	StartUp	4	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
	of `initial`, `system`, `node`, `initial node`, or `<Unknown>`.)				
`Node` *`node_id`*`: CM_REGCONF president =` *`president_id`*`, own Node =` *`own_id`*`, our dynamic id =` *`dynamic_id`*	Node *`president_id`* has been selected as "president". *`own_id`* and *`dynamic_id`* should always be the same as the ID (*`node_id`*) of the reporting node.	`CM_REGCONF`	`StartUp`	3	`INFO`
`Node` *`node_id`*`: CM_REGREF from Node` *`president_id`* `to our Node` *`node_id`*`. Cause =` *`cause`*	The reporting node (ID *`node_id`*) was unable to accept node *`president_id`* as president. The *`cause`* of the problem is given as one of `Busy`, `Election with wait = false`, `Not president`, `Election without selecting new candidate`, or `No such cause`.	`CM_REGREF`	`StartUp`	8	`INFO`
`Node` *`node_id`*`: We are Node` *`own_id`* `with dynamic ID` *`dynamic_id`*`, our left neighbor is Node` *`id_1`*`, our right is Node` *`id_2`*	The node has discovered its neighboring nodes in the cluster (node *`id_1`* and node *`id_2`*). *`node_id`*, *`own_id`*, and *`dynamic_id`* should always be the same; if they are not, this indicates a serious misconfiguration of the cluster nodes.	`FIND_NEIGHBOURS`	`StartUp`	8	`INFO`
`Node` *`node_id`*`:` *`type`* `shutdown initiated`	The node has received a shutdown signal. The *`type`* of shutdown is either `Cluster` or `Node`.	`NDBStopStarted`	`StartUp`	1	`INFO`
`Node` *`node_id`*`: Node shutdown completed`	The node has been shut down. This report may include	`NDBStopCompleted`	`StartUp`	1	`INFO`

Log Message	Description	Event Name	Event Type	Priority	Severity
[, *action*] [Initiated by signal *signal*.]	an *action*, which if present is one of restarting, no start, or initial. The report may also include a reference to an NDB Protocol *signal*; for possible signals, refer to Operations and Signals.				
Node *node_id*: Forced node shutdown completed [, action]. [Occured during startphase *start_phase*.] [Initiated by *signal*.] [Caused by error *error_code*: '*error_message*(*error_classification*). *error_status*'. [(extra info *extra_code*)]]	The node has been forcibly shut down. The *action* (one of restarting, no start, or initial) subsequently being taken, if any, is also reported. If the shutdown occurred while the node was starting, the report includes the *start_phase* during which the node failed. If this was a result of a *signal* sent to the node, this information is also provided (see Operations and Signals, for more information). If the error causing the failure is known, this is also included; for more information about NDB error messages and classifications, see MySQL Cluster API Errors.	NDBStopForced	StartUp	1	ALERT
Node *node_id*: Node shutdown aborted	The node shutdown process was aborted by the user.	NDBStopAborted	StartUp	1	INFO
Node *node_id*: StartLog: [GCI Keep: *keep_pos* LastCompleted: *last_pos*	This reports global checkpoints referenced during a node start. The redo log prior	StartREDOLog	StartUp	4	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
NewestRestorable *restore_pos*]	to *keep_pos* is dropped. *last_pos* is the last global checkpoint in which data node the participated; *restore_pos* is the global checkpoint which is actually used to restore all data nodes.				
startup_message [*Listed separately; see below.*]	There are a number of possible startup messages that can be logged under different circumstances. These are listed separately; see Section 7.7.2, "MySQL Cluster Log Startup Messages".	StartReport	StartUp	4	INFO
Node *node_id*: Node restart completed copy of dictionary information	Copying of data dictionary information to the restarted node has been completed.	NR_CopyDict	NodeRestart	8	INFO
Node *node_id*: Node restart completed copy of distribution information	Copying of data distribution information to the restarted node has been completed.	NR_CopyDistr	NodeRestart	8	INFO
Node *node_id*: Node restart starting to copy the fragments to Node *node_id*	Copy of fragments to starting data node *node_id* has begun	NR_CopyFragsStarted	NodeRestart	8	INFO
Node *node_id*: Table ID = *table_id*, fragment ID = *fragment_id* have been copied to Node *node_id*	Fragment *fragment_id* from table *table_id* has been copied to data node *node_id*	NR_CopyFragDone	NodeRestart	10	INFO
Node *node_id*: Node restart completed copying the	Copying of all table fragments to restarting data	NR_CopyFragsCompleted	NodeRestart	8	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
fragments to Node *node_id*	node *node_id* has been completed				
Node *node_id*: Node *node1_id* completed failure of Node *node2_id*	Data node *node1_id* has detected the failure of data node *node2_id*	NodeFailCompleted	NodeRestart	8	ALERT
All nodes completed failure of Node *node_id*	All (remaining) data nodes have detected the failure of data node *node_id*	NodeFailCompleted	NodeRestart	8	ALERT
Node failure of *node_id*block completed	The failure of data node *node_id* has been detected in the *block*NDB kernel block, where block is 1 of DBTC, DBDICT, DBDIH, or DBLQH; for more information, see NDB Kernel Blocks	NodeFailCompleted	NodeRestart	8	ALERT
Node *mgm_node_id*: Node *data_node_id* has failed. The Node state at failure was *state_code*	A data node has failed. Its state at the time of failure is described by an arbitration state code *state_code*: possible state code values can be found in the file include/kernel/signaldata/ArbitSignalData.hpp.	NODE_FAILREP	NodeRestart	8	ALERT
President restarts arbitration thread [state=*state_code*] or Prepare arbitrator node *node_id* [ticket=*ticket_id*] or Receive arbitrator node *node_id* [ticket=*ticket_id*] or Started arbitrator node *node_id* [ticket=*ticket_id*] or Lost	This is a report on the current state and progress of arbitration in the cluster. *node_id* is the node ID of the management node or SQL node selected as the arbitrator. *state_code* is an arbitration state code, as found in include/kernel/signaldata/ArbitSignalData.hpp. When an error	ArbitState	NodeRestart	6	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
arbitrator node *node_id* - process failure [state=*state_code*] or Lost arbitrator node *node_id* - process exit [state=*state_code*] or Lost arbitrator node *node_id* - *error_message* [state=*state_code*]	has occurred, an *error_message*, also defined in ArbitSignalData.hpp, is provided. *ticket_id* is a unique identifier handed out by the arbitrator when it is selected to all the nodes that participated in its selection; this is used to ensure that each node requesting arbitration was one of the nodes that took part in the selection process.				
Arbitration check lost - less than 1/2 nodes left or Arbitration check won - all node groups and more than 1/2 nodes left or Arbitration check won - node group majority or Arbitration check lost - missing node group or Network partitioning - arbitration required or Arbitration won - positive reply from node *node_id* or Arbitration lost - negative reply from node *node_id* or Network partitioning - no arbitrator available or Network partitioning -	This message reports on the result of arbitration. In the event of arbitration failure, an *error_message* and an arbitration *state_code* are provided; definitions for both of these are found in include/kernel/signaldata/ArbitSignalData.hpp.	ArbitResult	NodeRestart	2	ALERT

Log Message	Description	Event Name	Event Type	Priority	Severity
no arbitrator configured or Arbitration failure - *error_message* [state=*state_code*]					
Node *node_id*: GCP Take over started	This node is attempting to assume responsibility for the next global checkpoint (that is, it is becoming the master node)	GCP_TakeoverStarted	NodeRestart	7	INFO
Node *node_id*: GCP Take over completed	This node has become the master, and has assumed responsibility for the next global checkpoint	GCP_TakeoverCompleted	NodeRestart	7	INFO
Node *node_id*: LCP Take over started	This node is attempting to assume responsibility for the next set of local checkpoints (that is, it is becoming the master node)	LCP_TakeoverStarted	NodeRestart	7	INFO
Node *node_id*: LCP Take over completed	This node has become the master, and has assumed responsibility for the next set of local checkpoints	LCP_TakeoverCompleted	NodeRestart	7	INFO
Node *node_id*: Trans. Count = *transactions*, Commit Count = *commits*, Read Count = *reads*, Simple Read Count = *simple_reads*, Write Count = *writes*, AttrInfo Count = *AttrInfo_objects*, Concurrent Operations = *concurrent_operations*, Abort Count	This report of transaction activity is given approximately once every 10 seconds	TransReportCounters	Statistic	8	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
= *aborts*, Scans = *scans*, Range scans = *range_scans*					
Node *node_id*: Operations=*operations*	Number of operations performed by this node, provided approximately once every 10 seconds	OperationReportCounters	Statistic	8	INFO
Node *node_id*: Table with ID = *table_id* created	A table having the table ID shown has been created	TableCreated	Statistic	7	INFO
Node *node_id*: Mean loop Counter in doJob last 8192 times = *count*		JobStatistic	Statistic	9	INFO
Mean send size to Node = *node_id* last 4096 sends = *bytes* bytes	This node is sending an average of *bytes* bytes per send to node *node_id*	SendBytesStatistic	Statistic	9	INFO
Mean receive size to Node = *node_id* last 4096 sends = *bytes* bytes	This node is receiving an average of *bytes* of data each time it receives data from node *node_id*	ReceiveBytesStatistic	Statistic	9	INFO
Node *node_id*: Data usage is *data_memory_percentage*% (*data_pages_used* 32K pages of total *data_pages_total*) / Node *node_id*: Index usage is *index_memory_percentage*% (*index_pages_used* 8K pages of total *index_pages_total*)	This report is generated when a DUMP 1000 command is issued in the cluster management client; for more information, see DUMP 1000, in MySQL Cluster Internals	MemoryUsage	Statistic	5	INFO
Node *node1_id*: Transporter to node *node2_id* reported error *error_code*: *error_message*	A transporter error occurred while communicating with node *node2_id*; for a listing of transporter error codes and messages, see	TransporterError	Error	2	ERROR

Log Message	Description	Event Name	Event Type	Priority	Severity
	NDB Transporter Errors, in MySQL Cluster Internals				
Node *node1_id*: Transporter to node *node2_id* reported error *error_code*: *error_message*	A warning of a potential transporter problem while communicating with node *node2_id*; for a listing of transporter error codes and messages, see NDB Transporter Errors, for more information	TransporterWarning	Error	8	WARNING
Node *node1_id*: Node *node2_id* missed heartbeat *heartbeat_id*	This node missed a heartbeat from node *node2_id*	MissedHeartbeat	Error	8	WARNING
Node *node1_id*: Node *node2_id* declared dead due to missed heartbeat	This node has missed at least 3 heartbeats from node *node2_id*, and so has declared that node “dead”	DeadDueToHeartbeat	Error	8	ALERT
Node *node1_id*: Node Sent Heartbeat to node = *node2_id*	This node has sent a heartbeat to node *node2_id*	SentHeartbeat	Info	12	INFO
Node *node_id*: Event buffer status: used=*bytes_used* (*percent_used*%) alloc=*bytes_allocated* (*percent_available*%) max=*bytes_available* apply_gci=*latest_restorable_GCI* latest_gci=*latest_GCI*	This report is seen during heavy event buffer usage, for example, when many updates are being applied in a relatively short period of time; the report shows the number of bytes and the percentage of event buffer memory used, the bytes allocated and percentage still available, and the latest and latest restorable global checkpoints	EventBufferStatus	Info	7	INFO

Log Message	Description	Event Name	Event Type	Priority	Severity
`Node` *`node_id`*`: Entering single user mode`, `Node` *`node_id`*`: Entered single user mode Node` *`API_node_id`* `has exclusive access`, `Node` *`node_id`*`: Entering single user mode`	These reports are written to the cluster log when entering and exiting single user mode; *`API_node_id`* is the node ID of the API or SQL having exclusive access to the cluster (for more information, see Section 7.8, "MySQL Cluster Single User Mode"); the message `Unknown single user report` *`API_node_id`* indicates an error has taken place and should never be seen in normal operation	`SingleUser`	`Info`	7	`INFO`
`Node` *`node_id`*`: Backup` *`backup_id`* `started from node` *`mgm_node_id`*	A backup has been started using the management node having *`mgm_node_id`*; this message is also displayed in the cluster management client when the `START BACKUP` command is issued; for more information, see Section 7.3.2, "Using The MySQL Cluster Management Client to Create a Backup"	`BackupStarted`	`Backup`	7	`INFO`
`Node` *`node_id`*`: Backup` *`backup_id`* `started from node` *`mgm_node_id`* `completed. StartGCP:` *`start_gcp`* `StopGCP:` *`stop_gcp`*	The backup having the ID *`backup_id`* has been completed; for more information, see Section 7.3.2, "Using The MySQL Cluster Management Client to Create a Backup"	`BackupCompleted`	`Backup`	7	`INFO`

Log Message	Description	Event Name	Event Type	Priority	Severity
#Records: *records* #LogRecords: *log_records* Data: *data_bytes* bytes Log: *log_bytes* bytes					
Node *node_id*: Backup request from *mgm_node_id* failed to start. Error: *error_code*	The backup failed to start; for error codes, see MGM API Errors	BackupFailedToStart	Backup	7	ALERT
Node *node_id*: Backup *backup_id* started from *mgm_node_id* has been aborted. Error: *error_code*	The backup was terminated after starting, possibly due to user intervention	BackupAborted	Backup	7	ALERT

7.7.2 MySQL Cluster Log Startup Messages

Possible startup messages with descriptions are provided in the following list:

- `Initial start, waiting for %s to connect, nodes [ all: %s connected: %s no-wait: %s ]`
- `Waiting until nodes: %s connects, nodes [ all: %s connected: %s no-wait: %s ]`
- `Waiting %u sec for nodes %s to connect, nodes [ all: %s connected: %s no-wait: %s ]`
- `Waiting for non partitioned start, nodes [ all: %s connected: %s missing: %s no-wait: %s ]`
- `Waiting %u sec for non partitioned start, nodes [ all: %s connected: %s missing: %s no-wait: %s ]`
- `Initial start with nodes %s [ missing: %s no-wait: %s ]`
- `Start with all nodes %s`
- `Start with nodes %s [ missing: %s no-wait: %s ]`
- `Start potentially partitioned with nodes %s [ missing: %s no-wait: %s ]`
- `Unknown startreport: 0x%x [ %s %s %s %s ]`

7.7.3 MySQL Cluster: NDB Transporter Errors

This section lists error codes, names, and messages that are written to the cluster log in the event of transporter errors.

Error Code	Error Name	Error Text
0x00	TE_NO_ERROR	No error
0x01	TE_ERROR_CLOSING_SOCKET	Error found during closing of socket
0x02	TE_ERROR_IN_SELECT_BEFORE_ACCEPT	Error found before accept. The transporter will retry
0x03	TE_INVALID_MESSAGE_LENGTH	Error found in message (invalid message length)
0x04	TE_INVALID_CHECKSUM	Error found in message (checksum)
0x05	TE_COULD_NOT_CREATE_SOCKET	Error found while creating socket(can't create socket)
0x06	TE_COULD_NOT_BIND_SOCKET	Error found while binding server socket
0x07	TE_LISTEN_FAILED	Error found while listening to server socket
0x08	TE_ACCEPT_RETURN_ERROR	Error found during accept(accept return error)
0x0b	TE_SHM_DISCONNECT	The remote node has disconnected
0x0c	TE_SHM_IPC_STAT	Unable to check shm segment
0x0d	TE_SHM_UNABLE_TO_CREATE_SEGMENT	Unable to create shm segment
0x0e	TE_SHM_UNABLE_TO_ATTACH_SEGMENT	Unable to attach shm segment
0x0f	TE_SHM_UNABLE_TO_REMOVE_SEGMENT	Unable to remove shm segment
0x10	TE_TOO_SMALL_SIGID	Sig ID too small
0x11	TE_TOO_LARGE_SIGID	Sig ID too large
0x12	TE_WAIT_STACK_FULL	Wait stack was full
0x13	TE_RECEIVE_BUFFER_FULL	Receive buffer was full
0x14	TE_SIGNAL_LOST_SEND_BUFFER_FULL	Send buffer was full,and trying to force send fails
0x15	TE_SIGNAL_LOST	Send failed for unknown reason(signal lost)
0x16	TE_SEND_BUFFER_FULL	The send buffer was full, but sleeping for a while solved

Error Code	Error Name	Error Text
0x0017	**TE_SCI_LINK_ERROR**	There is no link from this node to the switch
0x18	**TE_SCI_UNABLE_TO_START_SEQUENCE**	Could not start a sequence, because system resources are exumed or no sequence has been created
0x19	**TE_SCI_UNABLE_TO_REMOVE_SEQUENCE**	Could not remove a sequence
0x1a	**TE_SCI_UNABLE_TO_CREATE_SEQUENCE**	Could not create a sequence, because system resources are exempted. Must reboot
0x1b	**TE_SCI_UNRECOVERABLE_DATA_TFX_ERROR**	Tried to send data on redundant link but failed
0x1c	**TE_SCI_CANNOT_INIT_LOCALSEGMENT**	Cannot initialize local segment
0x1d	**TE_SCI_CANNOT_MAP_REMOTESEGMENT**	Cannot map remote segment
0x1e	**TE_SCI_UNABLE_TO_UNMAP_SEGMENT**	Cannot free the resources used by this segment (step 1)
0x1f	**TE_SCI_UNABLE_TO_REMOVE_SEGMENT**	Cannot free the resources used by this segment (step 2)
0x20	**TE_SCI_UNABLE_TO_DISCONNECT_SEGMENT**	Cannot disconnect from a remote segment
0x21	**TE_SHM_IPC_PERMANENT**	Shm ipc Permanent error
0x22	**TE_SCI_UNABLE_TO_CLOSE_CHANNEL**	Unable to close the sci channel and the resources allocated

7.8 MySQL Cluster Single User Mode

Single user mode enables the database administrator to restrict access to the database system to a single API node, such as a MySQL server (SQL node) or an instance of `ndb_restore`. When entering single user mode, connections to all other API nodes are closed gracefully and all running transactions are aborted. No new transactions are permitted to start.

Once the cluster has entered single user mode, only the designated API node is granted access to the database.

You can use the `ALL STATUS` command in the `ndb_mgm` client to see when the cluster has entered single user mode. You can also check the `status` column of the `ndbinfo.nodes` table (see Section 7.10.18, “The ndbinfo nodes Table”, for more information).

Example:

```
ndb_mgm> ENTER SINGLE USER MODE 5
```

After this command has executed and the cluster has entered single user mode, the API node whose node ID is `5` becomes the cluster's only permitted user.

The node specified in the preceding command must be an API node; attempting to specify any other type of node will be rejected.

> **Note**
>
> When the preceding command is invoked, all transactions running on the designated node are aborted, the connection is closed, and the server must be restarted.

The command `EXIT SINGLE USER MODE` changes the state of the cluster's data nodes from single user mode to normal mode. API nodes—such as MySQL Servers—waiting for a connection (that is, waiting for the cluster to become ready and available), are again permitted to connect. The API node denoted as the single-user node continues to run (if still connected) during and after the state change.

Example:

```
ndb_mgm> EXIT SINGLE USER MODE
```

There are two recommended ways to handle a node failure when running in single user mode:

- Method 1:
  1. Finish all single user mode transactions
  2. Issue the `EXIT SINGLE USER MODE` command
  3. Restart the cluster's data nodes
- Method 2:

  Restart database nodes prior to entering single user mode.

7.9 Quick Reference: MySQL Cluster SQL Statements

This section discusses several SQL statements that can prove useful in managing and monitoring a MySQL server that is connected to a MySQL Cluster, and in some cases provide information about the cluster itself.

- `SHOW ENGINE NDB STATUS`, `SHOW ENGINE NDBCLUSTER STATUS`

  The output of this statement contains information about the server's connection to the cluster, creation and usage of MySQL Cluster objects, and binary logging for MySQL Cluster replication.

  See SHOW ENGINE Syntax, for a usage example and more detailed information.
- `SHOW ENGINES`

  This statement can be used to determine whether or not clustering support is enabled in the MySQL server, and if so, whether it is active.

  See SHOW ENGINES Syntax, for more detailed information.

  > **Note**
  >
  > This statement does not support a `LIKE` clause. However, you can use `LIKE` to filter queries against the `INFORMATION_SCHEMA.ENGINES` table, as discussed in the next item.
- `SELECT * FROM INFORMATION_SCHEMA.ENGINES [WHERE ENGINE LIKE 'NDB%']`

This is the equivalent of SHOW ENGINES, but uses the ENGINES table of the INFORMATION_SCHEMA database. Unlike the case with the SHOW ENGINES statement, it is possible to filter the results using a LIKE clause, and to select specific columns to obtain information that may be of use in scripts. For example, the following query shows whether the server was built with NDB support and, if so, whether it is enabled:

```
mysql> SELECT SUPPORT FROM INFORMATION_SCHEMA.ENGINES
    ->   WHERE ENGINE LIKE 'NDB%';
+---------+
| support |
+---------+
| ENABLED |
+---------+
```

See The INFORMATION_SCHEMA ENGINES Table, for more information.

- `SHOW VARIABLES LIKE 'NDB%'`

  This statement provides a list of most server system variables relating to the NDB storage engine, and their values, as shown here:

```
mysql> SHOW VARIABLES LIKE 'NDB%';
+-------------------------------------+-------+
| Variable_name                       | Value |
+-------------------------------------+-------+
| ndb_autoincrement_prefetch_sz       | 32    |
| ndb_cache_check_time                | 0     |
| ndb_extra_logging                   | 0     |
| ndb_force_send                      | ON    |
| ndb_index_stat_cache_entries        | 32    |
| ndb_index_stat_enable               | OFF   |
| ndb_index_stat_update_freq          | 20    |
| ndb_report_thresh_binlog_epoch_slip | 3     |
| ndb_report_thresh_binlog_mem_usage  | 10    |
| ndb_use_copying_alter_table         | OFF   |
| ndb_use_exact_count                 | ON    |
| ndb_use_transactions                | ON    |
+-------------------------------------+-------+
```

  See Server System Variables, for more information.

- `SELECT * FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES WHERE VARIABLE_NAME LIKE 'NDB%';`

  This statement is the equivalent of the SHOW command described in the previous item, and provides almost identical output, as shown here:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES
    ->   WHERE VARIABLE_NAME LIKE 'NDB%';
+-------------------------------------+----------------+
| VARIABLE_NAME                       | VARIABLE_VALUE |
+-------------------------------------+----------------+
| NDB_AUTOINCREMENT_PREFETCH_SZ       | 32             |
| NDB_CACHE_CHECK_TIME                | 0              |
| NDB_EXTRA_LOGGING                   | 0              |
| NDB_FORCE_SEND                      | ON             |
| NDB_INDEX_STAT_CACHE_ENTRIES        | 32             |
| NDB_INDEX_STAT_ENABLE               | OFF            |
| NDB_INDEX_STAT_UPDATE_FREQ          | 20             |
| NDB_REPORT_THRESH_BINLOG_EPOCH_SLIP | 3              |
| NDB_REPORT_THRESH_BINLOG_MEM_USAGE  | 10             |
| NDB_USE_COPYING_ALTER_TABLE         | OFF            |
| NDB_USE_EXACT_COUNT                 | ON             |
| NDB_USE_TRANSACTIONS                | ON             |
+-------------------------------------+----------------+
```

Unlike the case with the SHOW command, it is possible to select individual columns. For example:

```
mysql> SELECT VARIABLE_VALUE
    ->   FROM INFORMATION_SCHEMA.GLOBAL_VARIABLES
    ->   WHERE VARIABLE_NAME = 'ndb_force_send';
+----------------+
| VARIABLE_VALUE |
+----------------+
| ON             |
+----------------+
```

See The INFORMATION_SCHEMA GLOBAL_VARIABLES and SESSION_VARIABLES Tables, and Server System Variables, for more information.

- SHOW STATUS LIKE 'NDB%'

  This statement shows at a glance whether or not the MySQL server is acting as a cluster SQL node, and if so, it provides the MySQL server's cluster node ID, the host name and port for the cluster management server to which it is connected, and the number of data nodes in the cluster, as shown here:

```
mysql> SHOW STATUS LIKE 'NDB%';
+--------------------------+---------------+
| Variable_name            | Value         |
+--------------------------+---------------+
| Ndb_cluster_node_id      | 10            |
| Ndb_config_from_host     | 192.168.0.103 |
| Ndb_config_from_port     | 1186          |
| Ndb_number_of_data_nodes | 4             |
+--------------------------+---------------+
```

  If the MySQL server was built with clustering support, but it is not connected to a cluster, all rows in the output of this statement contain a zero or an empty string:

```
mysql> SHOW STATUS LIKE 'NDB%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Ndb_cluster_node_id      | 0     |
| Ndb_config_from_host     |       |
| Ndb_config_from_port     | 0     |
| Ndb_number_of_data_nodes | 0     |
+--------------------------+-------+
```

  See also SHOW STATUS Syntax.

- SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_NAME LIKE 'NDB%';

  This statement provides similar output to the SHOW command discussed in the previous item. However, unlike the case with SHOW STATUS, it is possible using the SELECT to extract values in SQL for use in scripts for monitoring and automation purposes.

  See The INFORMATION_SCHEMA GLOBAL_STATUS and SESSION_STATUS Tables, for more information.

You can also query the tables in the ndbinfo information database for real-time data about many MySQL Cluster operations. See Section 7.10, “The ndbinfo MySQL Cluster Information Database”.

7.10 The ndbinfo MySQL Cluster Information Database

ndbinfo is a database storing containing information specific to MySQL Cluster.

This database contains a number of tables, each providing a different sort of data about MySQL Cluster node status, resource usage, and operations. You can find more detailed information about each of these tables in the next several sections.

`ndbinfo` is included with MySQL Cluster support in the MySQL Server; no special compilation or configuration steps are required; the tables are created by the MySQL Server when it connects to the cluster. You can verify that `ndbinfo` support is active in a given MySQL Server instance using `SHOW PLUGINS`; if `ndbinfo` support is enabled, you should see a row containing `ndbinfo` in the `Name` column and `ACTIVE` in the `Status` column, as shown here (emphasized text):

```
mysql> SHOW PLUGINS;
+----------------------------------+--------+--------------------+---------+---------+
| Name                             | Status | Type               | Library | License |
+----------------------------------+--------+--------------------+---------+---------+
| binlog                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| mysql_native_password            | ACTIVE | AUTHENTICATION     | NULL    | GPL     |
| mysql_old_password               | ACTIVE | AUTHENTICATION     | NULL    | GPL     |
| CSV                              | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MEMORY                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MRG_MYISAM                       | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| MyISAM                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| PERFORMANCE_SCHEMA               | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| BLACKHOLE                        | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ARCHIVE                          | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndbcluster                       | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndbinfo                          | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| ndb_transid_mysql_connection_map | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| InnoDB                           | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
| INNODB_TRX                       | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCKS                     | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_LOCK_WAITS                | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP                       | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMP_RESET                 | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM                    | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| INNODB_CMPMEM_RESET              | ACTIVE | INFORMATION SCHEMA | NULL    | GPL     |
| partition                        | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
+----------------------------------+--------+--------------------+---------+---------+
22 rows in set (0.00 sec)
```

You can also do this by checking the output of `SHOW ENGINES` for a line including `ndbinfo` in the `Engine` column and `YES` in the `Support` column, as shown here (emphasized text):

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: ndbcluster
     Support: YES
     Comment: Clustered, fault-tolerant tables
Transactions: YES
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: ndbinfo
     Support: YES
     Comment: MySQL Cluster system information storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ***************************
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
```

```
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 6. row ***************************
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
*************************** 7. row ***************************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ***************************
      Engine: InnoDB
     Support: YES
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 9. row ***************************
      Engine: MyISAM
     Support: DEFAULT
     Comment: Default engine as of MySQL 3.23 with great performance
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 10. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
          XA: NO
  Savepoints: NO
10 rows in set (0.00 sec)
```

If `ndbinfo` support is enabled, then you can access `ndbinfo` using SQL statements in `mysql` or another MySQL client. For example, you can see `ndbinfo` listed in the output of `SHOW DATABASES`, as shown here:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| ndbinfo            |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

If the `mysqld` process was not started with the `--ndbcluster` option, `ndbinfo` is not available and is not displayed by `SHOW DATABASES`. If `mysqld` was formerly connected to a MySQL Cluster but the cluster becomes unavailable (due to events such as cluster shutdown, loss of network connectivity, and so forth), `ndbinfo` and its tables remain visible, but an attempt to access any tables (other than `blocks` or `config_params`) fails with `Got error 157 'Connection to NDB failed' from NDBINFO`.

With the exception of the `blocks` and `config_params` tables, what we refer to as `ndbinfo` "tables" are actually views generated from internal `NDB` tables not normally visible to the MySQL Server.

All `ndbinfo` tables are read-only, and are generated on demand when queried. Because many of them are generated in parallel by the data nodes while other are specific to a given SQL node, they are not guaranteed to provide a consistent snapshot.

In addition, pushing down of joins is not supported on `ndbinfo` tables; so joining large `ndbinfo` tables can require transfer of a large amount of data to the requesting API node, even when the query makes use of a `WHERE` clause.

`ndbinfo` tables are not included in the query cache. (Bug #59831)

You can select the `ndbinfo` database with a `USE` statement, and then issue a `SHOW TABLES` statement to obtain a list of tables, just as for any other database, like this:

```
mysql> USE ndbinfo;
Database changed
mysql> SHOW TABLES;
+---------------------------------+
| Tables_in_ndbinfo               |
+---------------------------------+
| arbitrator_validity_detail      |
| arbitrator_validity_summary     |
| blocks                          |
| cluster_operations              |
| cluster_transactions            |
| config_params                   |
| counters                        |
| dict_obj_types                  |
| disk_write_speed_aggregate      |
| disk_write_speed_aggregate_node |
| disk_write_speed_base           |
| diskpagebuffer                  |
| logbuffers                      |
| logspaces                       |
| membership                      |
| memory_per_fragment             |
| memoryusage                     |
| nodes                           |
| operations_per_fragment         |
| resources                       |
| restart_info                    |
| server_operations               |
| server_transactions             |
| threadblocks                    |
| threadstat                      |
| transporters                    |
+---------------------------------+
26 rows in set (0.00 sec)
```

The `dict_obj_types`, `disk_write_speed_aggregate`, `disk_write_speed_aggregate_node`, `disk_write_speed_base`, and `memory_per_fragment` tables were added in MySQL Cluster NDB 7.4.1. The `restart_info` table was added in MySQL Cluster NDB 7.4.2. The `operations_per_fragment` table was added in MySQL Cluster NDB 7.4.3.

You can execute `SELECT` statements against these tables, just as you would normally expect:

```
mysql> SELECT * FROM memoryusage;
+---------+---------------------+--------+------------+------------+-------------+
| node_id | memory_type         | used   | used_pages | total      | total_pages |
+---------+---------------------+--------+------------+------------+-------------+
|       5 | Data memory         | 753664 |         23 | 1073741824 |       32768 |
|       5 | Index memory        | 163840 |         20 | 1074003968 |      131104 |
|       5 | Long message buffer |   2304 |          9 |   67108864 |      262144 |
|       6 | Data memory         | 753664 |         23 | 1073741824 |       32768 |
|       6 | Index memory        | 163840 |         20 | 1074003968 |      131104 |
```

```
|       6 | Long message buffer |   2304 |          9 |   67108864 |      262144 |
+---------+---------------------+--------+------------+------------+-------------+
6 rows in set (0.02 sec)
```

More complex queries, such as the two following SELECT statements using the memoryusage table, are possible:

```
mysql> SELECT SUM(used) as 'Data Memory Used, All Nodes'
     >     FROM memoryusage
     >     WHERE memory_type = 'Data memory';
+-----------------------------+
| Data Memory Used, All Nodes |
+-----------------------------+
|                        6460 |
+-----------------------------+
1 row in set (0.37 sec)
mysql> SELECT SUM(max) as 'Total IndexMemory Available'
     >     FROM memoryusage
     >     WHERE memory_type = 'Index memory';
+-----------------------------+
| Total IndexMemory Available |
+-----------------------------+
|                       25664 |
+-----------------------------+
1 row in set (0.33 sec)
```

ndbinfo table and column names are case sensitive (as is the name of the ndbinfo database itself). These identifiers are in lowercase. Trying to use the wrong lettercase results in an error, as shown in this example:

```
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+
| node_id | uptime | status  | start_phase |
+---------+--------+---------+-------------+
|       1 |  13602 | STARTED |           0 |
|       2 |     16 | STARTED |           0 |
+---------+--------+---------+-------------+
2 rows in set (0.04 sec)
mysql> SELECT * FROM Nodes;
ERROR 1146 (42S02): Table 'ndbinfo.Nodes' doesn't exist
```

mysqldump ignores the ndbinfo database entirely, and excludes it from any output. This is true even when using the --databases or --all-databases option.

MySQL Cluster also maintains tables in the INFORMATION_SCHEMA information database, including the FILES table which contains information about files used for MySQL Cluster Disk Data storage, and the ndb_transid_mysql_connection_map table, which shows the relationships between transactions, transaction coordinators, and MySQL Cluster API nodes. For more information, see the descriptions of the tables or INFORMATION_SCHEMA Tables for MySQL Cluster.

7.10.1 The ndbinfo arbitrator_validity_detail Table

The arbitrator_validity_detail table shows the view that each data node in the cluster of the arbitrator. It is a subset of the membership table.

The following table provides information about the columns in the arbitrator_validity_detail table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	This node's node ID
arbitrator	integer	Node ID of arbitrator
arb_ticket	string	Internal identifier used to track arbitration

Column Name	Type	Description
arb_connected	Yes or No	Whether this node is connected to the arbitrator
arb_state	Enumeration (see text)	Arbitration state

The node ID is the same as that reported by ndb_mgm -e "SHOW".

All nodes should show the same arbitrator and arb_ticket values as well as the same arb_state value. Possible arb_state values are ARBIT_NULL, ARBIT_INIT, ARBIT_FIND, ARBIT_PREP1, ARBIT_PREP2, ARBIT_START, ARBIT_RUN, ARBIT_CHOOSE, ARBIT_CRASH, and UNKNOWN.

arb_connected shows whether the current node is connected to the arbitrator.

7.10.2 The ndbinfo arbitrator_validity_summary Table

The arbitrator_validity_summary table provides a composite view of the arbitrator with regard to the cluster's data nodes.

The following table provides information about the columns in the arbitrator_validity_summary table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
arbitrator	integer	Node ID of arbitrator
arb_ticket	string	Internal identifier used to track arbitration
arb_connected	Yes or No	Whether this arbitrator is connected to the cluster
consensus_count	integer	Number of data nodes that see this node as arbitrator

In normal operations, this table should have only 1 row for any appreciable length of time. If it has more than 1 row for longer than a few moments, then either not all nodes are connected to the arbitrator, or all nodes are connected, but do not agree on the same arbitrator.

The arbitrator column shows the arbitrator's node ID.

arb_ticket is the internal identifier used by this arbitrator.

arb_connected shows whether this node is connected to the cluster as an arbitrator.

7.10.3 The ndbinfo blocks Table

The blocks table is a static table which simply contains the names and internal IDs of all NDB kernel blocks (see NDB Kernel Blocks). It is for use by the other ndbinfo tables (most of which are actually views) in mapping block numbers to block names for producing human-readable output.

The following table provides information about the columns in the blocks table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
block_number	integer	Block number
block_name	string	Block name

To obtain a list of all block names, simply execute SELECT block_name FROM ndbinfo.blocks. Although this is a static table, its content can vary between different MySQL Cluster releases.

7.10.4 The ndbinfo cluster_operations Table

The `cluster_operations` table provides a per-operation (stateful primary key op) view of all activity in the MySQL Cluster from the point of view of the local data management (LQH) blocks (see The DBLQH Block).

The following table provides information about the columns in the `cluster_operations` table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
`node_id`	integer	Node ID of reporting LQH block
`block_instance`	integer	LQH block instance
`transid`	integer	Transaction ID
`operation_type`	string	Operation type (see text for possible values)
`state`	string	Operation state (see text for possible values)
`tableid`	integer	Table ID
`fragmentid`	integer	Fragment ID
`client_node_id`	integer	Client node ID
`client_block_ref`	integer	Client block reference
`tc_node_id`	integer	Transaction coordinator node ID
`tc_block_no`	integer	Transaction coordinator block number
`tc_block_instance`	integer	Transaction coordinator block instance

The transaction ID is a unique 64-bit number which can be obtained using the NDB API's `getTransactionId()` method. (Currently, the MySQL Server does not expose the NDB API transaction ID of an ongoing transaction.)

The `operation_type` column can take any one of the values `READ`, `READ-SH`, `READ-EX`, `INSERT`, `UPDATE`, `DELETE`, `WRITE`, `UNLOCK`, `REFRESH`, `SCAN`, `SCAN-SH`, `SCAN-EX`, or `<unknown>`.

The `state` column can have any one of the values `ABORT_QUEUED`, `ABORT_STOPPED`, `COMMITTED`, `COMMIT_QUEUED`, `COMMIT_STOPPED`, `COPY_CLOSE_STOPPED`, `COPY_FIRST_STOPPED`, `COPY_STOPPED`, `COPY_TUPKEY`, `IDLE`, `LOG_ABORT_QUEUED`, `LOG_COMMIT_QUEUED`, `LOG_COMMIT_QUEUED_WAIT_SIGNAL`, `LOG_COMMIT_WRITTEN`, `LOG_COMMIT_WRITTEN_WAIT_SIGNAL`, `LOG_QUEUED`, `PREPARED`, `PREPARED_RECEIVED_COMMIT`, `SCAN_CHECK_STOPPED`, `SCAN_CLOSE_STOPPED`, `SCAN_FIRST_STOPPED`, `SCAN_RELEASE_STOPPED`, `SCAN_STATE_USED`, `SCAN_STOPPED`, `SCAN_TUPKEY`, `STOPPED`, `TC_NOT_CONNECTED`, `WAIT_ACC`, `WAIT_ACC_ABORT`, `WAIT_AI_AFTER_ABORT`, `WAIT_ATTR`, `WAIT_SCAN_AI`, `WAIT_TUP`, `WAIT_TUPKEYINFO`, `WAIT_TUP_COMMIT`, or `WAIT_TUP_TO_ABORT`. (If the MySQL Server is running with `ndbinfo_show_hidden` enabled, you can view this list of states by selecting from the `ndb$dblqh_tcconnect_state` table, which is normally hidden.)

You can obtain the name of an `NDB` table from its table ID by checking the output of `ndb_show_tables`.

The `fragid` is the same as the partition number seen in the output of `ndb_desc --extra-partition-info` (short form `-p`).

In `client_node_id` and `client_block_ref`, `client` refers to a MySQL Cluster API or SQL node (that is, an NDB API client or a MySQL Server attached to the cluster).

7.10.5 The ndbinfo cluster_transactions Table

The cluster_transactions table shows information about all ongoing transactions in a MySQL Cluster.

The following table provides information about the columns in the cluster_transactions table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID of transaction coordinator
block_instance	integer	TC block instance
transid	integer	Transaction ID
state	string	Operation state (see text for possible values)
count_operations	integer	Number of stateful primary key operations in transaction (includes reads with locks, as well as DML operations)
outstanding_operations	integer	Operations still being executed in local data management blocks
inactive_seconds	integer	Time spent waiting for API
client_node_id	integer	Client node ID
client_block_ref	integer	Client block reference

The transaction ID is a unique 64-bit number which can be obtained using the NDB API's getTransactionId() method. (Currently, the MySQL Server does not expose the NDB API transaction ID of an ongoing transaction.)

The state column can have any one of the values CS_ABORTING, CS_COMMITTING, CS_COMMIT_SENT, CS_COMPLETE_SENT, CS_COMPLETING, CS_CONNECTED, CS_DISCONNECTED, CS_FAIL_ABORTED, CS_FAIL_ABORTING, CS_FAIL_COMMITTED, CS_FAIL_COMMITTING, CS_FAIL_COMPLETED, CS_FAIL_PREPARED, CS_PREPARE_TO_COMMIT, CS_RECEIVING, CS_REC_COMMITTING, CS_RESTART, CS_SEND_FIRE_TRIG_REQ, CS_STARTED, CS_START_COMMITTING, CS_START_SCAN, CS_WAIT_ABORT_CONF, CS_WAIT_COMMIT_CONF, CS_WAIT_COMPLETE_CONF, CS_WAIT_FIRE_TRIG_REQ. (If the MySQL Server is running with ndbinfo_show_hidden enabled, you can view this list of states by selecting from the ndb$dbtc_apiconnect_state table, which is normally hidden.)

In client_node_id and client_block_ref, client refers to a MySQL Cluster API or SQL node (that is, an NDB API client or a MySQL Server attached to the cluster).

7.10.6 The ndbinfo config_params Table

The config_params table is a static table which provides the names and internal ID numbers of all MySQL Cluster configuration parameters.

The following table provides information about the columns in the config_params table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
param_number	integer	The parameter's internal ID number
param_name	string	The name of the parameter

Although this is a static table, its content can vary between MySQL Cluster installations, since supported parameters can vary due to differences between software releases, cluster hardware configurations, and other factors.

7.10.7 The ndbinfo counters Table

The counters table provides running totals of events such as reads and writes for specific kernel blocks and data nodes. Counts are kept from the most recent node start or restart; a node start or restart resets all counters on that node. Not all kernel blocks have all types of counters.

The following table provides information about the columns in the counters table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	The data node ID
block_name	string	Name of the associated NDB kernel block (see NDB Kernel Blocks).
block_instance	integer	Block instance
counter_id	integer	The counter's internal ID number; normally an integer between 1 and 10, inclusive.
counter_name	string	The name of the counter. See text for names of individual counters and the NDB kernel block with which each counter is associated.
val	integer	The counter's value

Each counter is associated with a particular NDB kernel block.

The OPERATIONS counter is associated with the DBLQH (local query handler) kernel block (see The DBLQH Block). A primary-key read counts as one operation, as does a primary-key update. For reads, there is one operation in DBLQH per operation in DBTC. For writes, there is one operation counted per replica.

The ATTRINFO, TRANSACTIONS, COMMITS, READS, LOCAL_READS, SIMPLE_READS, WRITES, LOCAL_WRITES, ABORTS, TABLE_SCANS, and RANGE_SCANS counters are associated with the DBTC (transaction co-ordinator) kernel block (see The DBTC Block).

LOCAL_WRITES and LOCAL_READS are primary-key operations using a transaction coordinator in a node that also holds the primary replica of the record.

The READS counter includes all reads. LOCAL_READS includes only those reads of the primary replica on the same node as this transaction coordinator. SIMPLE_READS includes only those reads in which the read operation is the beginning and ending operation for a given transaction. Simple reads do not hold locks but are part of a transaction, in that they observe uncommitted changes made by the transaction containing them but not of any other uncommitted transactions. Such reads are “simple” from the point of view of the TC block; since they hold no locks they are not durable, and once DBTC has routed them to the relevant LQH block, it holds no state for them.

ATTRINFO keeps a count of the number of times an interpreted program is sent to the data node. See NDB Protocol Messages, for more information about ATTRINFO messages in the NDB kernel.

The LOCAL_TABLE_SCANS_SENT, READS_RECEIVED, PRUNED_RANGE_SCANS_RECEIVED, RANGE_SCANS_RECEIVED, LOCAL_READS_SENT, CONST_PRUNED_RANGE_SCANS_RECEIVED, LOCAL_RANGE_SCANS_SENT, REMOTE_READS_SENT, REMOTE_RANGE_SCANS_SENT, READS_NOT_FOUND, SCAN_BATCHES_RETURNED, TABLE_SCANS_RECEIVED, and SCAN_ROWS_RETURNED counters are associated with the DBSPJ (select push-down join) kernel block (see The DBSPJ Block).

A number of counters provide information about transporter overload and send buffer sizing when troubleshooting such issues. For each LQH instance, there is one instance of each counter in the following list:

- LQHKEY_OVERLOAD: Number of primary key requests rejected at the LQH block instance due to transporter overload
- LQHKEY_OVERLOAD_TC: Count of instances of LQHKEY_OVERLOAD where the TC node transporter was overloaded
- LQHKEY_OVERLOAD_READER: Count of instances of LQHKEY_OVERLOAD where the API reader (reads only) node was overloaded.
- LQHKEY_OVERLOAD_NODE_PEER: Count of instances of LQHKEY_OVERLOAD where the next backup data node (writes only) was overloaded
- LQHKEY_OVERLOAD_SUBSCRIBER: Count of instances of LQHKEY_OVERLOAD where a event subscriber (writes only) was overloaded.
- LQHSCAN_SLOWDOWNS: Count of instances where a fragment scan batch size was reduced due to scanning API transporter overload.

7.10.8 The ndbinfo dict_obj_types Table

The dict_obj_types table is a static table listing possible dictionary object types used in the NDB kernel. These are the same types defined by Object::Type in the NDB API.

The following table provides information about the columns in the dict_obj_types table. For each column, the table shows the name, data type, and a brief description.

Column Name	Type	Description
type_id	integer	The type ID for this type
type_name	string	The name of this type

This table was added in MySQL Cluster NDB 7.4.1.

7.10.9 The ndbinfo disk_write_speed_base Table

The disk_write_speed_base table provides base information about the speed of disk writes during LCP, backup, and restore operations.

The following table provides information about the columns in the disk_write_speed_base table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID of this node
thr_no	integer	Thread ID of this LDM thread
millis_ago	integer	Milliseconds since this reporting period ended
millis_passed	integer	Milliseconds elapsed in this reporting period
backup_lcp_bytes_written	integer	Number of bytes written to disk by local checkpoints and backup processes during this period
redo_bytes_written	integer	Number of bytes written to REDO log during this period
target_disk_write_speed	integer	Actual speed of disk writes per LDM thread (base data)

This table was added in MySQL Cluster NDB 7.4.1.

7.10.10 The ndbinfo disk_write_speed_aggregate Table

The `disk_write_speed_aggregate` table provides aggregated information about the speed of disk writes during LCP, backup, and restore operations.

The following table provides information about the columns in the `disk_write_speed_aggregate` table in MySQL Cluster NDB 7.4.3 and later; information about `disk_write_speed_aggregate` in MySQL Cluster NDB 7.4.2 and earlier can be found later in this section. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID of this node
thr_no	integer	Thread ID of this LDM thread
backup_lcp_speed_last_sec	integer	Number of bytes written to disk by backup and LCP processes in the last second
redo_speed_last_sec	integer	Number of bytes written to REDO log in the last second
backup_lcp_speed_last_10sec	integer	Number of bytes written to disk by backup and LCP processes per second, averaged over the last 10 seconds
redo_speed_last_10sec	integer	Number of bytes written to REDO log per second, averaged over the last 10 seconds
std_dev_backup_lcp_speed_last_10sec	integer	Standard deviation in number of bytes written to disk by backup and LCP processes per second, averaged over the last 10 seconds
std_dev_redo_speed_last_10sec	integer	Standard deviation in number of bytes written to REDO log per second, averaged over the last 10 seconds
backup_lcp_speed_last_60sec	integer	Number of bytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds
redo_speed_last_60sec	integer	Number of bytes written to REDO log per second, averaged over the last 10 seconds
std_dev_backup_lcp_speed_last_60sec	integer	Standard deviation in number of bytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds
std_dev_redo_speed_last_60sec	integer	Standard deviation in number of bytes written to REDO log per second, averaged over the last 60 seconds
slowdowns_due_to_io_lag	integer	Number of seconds that disk writes were slowed due to REDO log I/O lag
slowdowns_due_to_high_cpu	integer	Number of seconds that disk writes were slowed due to high CPU usage
disk_write_speed_set_to_min	integer	Number of seconds that disk write speed was set to minimum

Column Name	Type	Description
current_target_disk_write_speed	integer	Actual speed of disk writes per LDM thread (aggregated)

Prior to MySQL Cluster NDB 7.4.3, the standard deviation used in obtaining the values shown in the `std_dev_backup_lcp_speed_last_10sec`, `std_dev_redo_speed_last_10sec`, `std_dev_backup_lcp_speed_last_60sec`, and `std_dev_redo_speed_last_60sec` columns of this table was not calculated correctly. (Bug #74317, Bug #19795072)

Prior to MySQL Cluster NDB 7.4.3, some columns of this table showed values in kilobytes or other multiples of bytes. In MySQL Cluster NDB 7.4.3 and later, all such columns display values in bytes. (Bug #74317, Bug #19795072) The following table provides information about `disk_write_speed_aggregate` as implemented previously.

Column Name	Type	Description
node_id	integer	Node ID of this node
thr_no	integer	Thread ID of this LDM thread
backup_lcp_speed_last_sec	integer	Number of kilobytes written to disk by backup and LCP processes in the last second
redo_speed_last_sec	integer	Number of kilobytes written to REDO log in the last second
backup_lcp_speed_last_10sec	integer	Number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 10 seconds
redo_speed_last_10sec	integer	Number of kilobytes written to REDO log per second, averaged over the last 10 seconds
std_dev_backup_lcp_speed_last_10sec	integer	Standard deviation in number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 10 seconds
std_dev_redo_speed_last_10sec	integer	Standard deviation in number of kilobytes written to REDO log per second, averaged over the last 10 seconds
backup_lcp_speed_last_60sec	integer	Number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds
redo_speed_last_60sec	integer	Number of kilobytes written to REDO log per second, averaged over the last 10 seconds
std_dev_backup_lcp_speed_last_60sec	integer	Standard deviation in number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds
std_dev_redo_speed_last_60sec	integer	Standard deviation in number of kilobytes written to REDO log per second, averaged over the last 60 seconds
slowdowns_due_to_io_lag	integer	Number of seconds that disk writes were slowed due to REDO log I/O lag

Column Name	Type	Description
slowdowns_due_to_high_cpu	integer	Number of seconds that disk writes were slowed due to high CPU usage
disk_write_speed_set_to_min	integer	Number of seconds that disk write speed was set to minimum
current_target_disk_write_speed	integer	Actual speed of disk writes per LDM thread (aggregated)

The `disk_write_speed_aggregate` table was added in MySQL Cluster NDB 7.4.1.

7.10.11 The ndbinfo disk_write_speed_aggregate_node Table

The `disk_write_speed_aggregate_node` table provides aggregated information per node about the speed of disk writes during LCP, backup, and restore operations.

The following table provides information about the columns in the `disk_write_speed_aggregate_node` table as implemented in MySQL Cluster NDB 7.4.3 and later; information about `disk_write_speed_aggregate_node` in MySQL Cluster NDB 7.4.2 and earlier can be found later in this section. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID of this node
backup_lcp_speed_last_sec	numeric	Number of bytes written to disk by backup and LCP processes in the last second
redo_speed_last_sec	numeric	Number of bytes written to REDO log in the last second
backup_lcp_speed_last_10sec	numeric	Number of bytes written to disk by backup and LCP processes per second, averaged over the last 10 seconds
redo_speed_last_10sec	numeric	Number of bytes written to REDO log per second, averaged over the last 10 seconds
backup_lcp_speed_last_60sec	numeric	Number of bytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds
redo_speed_last_60sec	numeric	Number of bytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds

Prior to MySQL Cluster NDB 7.4.3, columns of this table showed values in kilobytes. In MySQL Cluster NDB 7.4.3 and later, the columns display all such values in bytes. (Bug #74317, Bug #19795072) The following table provides information about `disk_write_speed_aggregate` as implemented in MySQL Cluster NDB 7.4.2 and earlier.

Column Name	Type	Description
node_id	integer	Node ID of this node
backup_lcp_speed_last_sec	numeric	Number of kilobytes written to disk by backup and LCP processes in the last second
redo_speed_last_sec	numeric	Number of kilobytes written to REDO log in the last second

Column Name	Type	Description
backup_lcp_speed_last_10sec	numeric	Number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 10 seconds
redo_speed_last_10sec	numeric	Number of kilobytes written to REDO log per second, averaged over the last 10 seconds
backup_lcp_speed_last_60sec	numeric	Number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds
redo_speed_last_60sec	numeric	Number of kilobytes written to disk by backup and LCP processes per second, averaged over the last 60 seconds

The `disk_write_speed_aggregate_node` table was added in MySQL Cluster NDB 7.4.1.

7.10.12 The ndbinfo diskpagebuffer Table

The `diskpagebuffer` table provides statistics about disk page buffer usage by MySQL Cluster Disk Data tables.

The following table provides information about the columns in the `diskpagebuffer` table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
`node_id`	integer	The data node ID
`block_instance`	integer	Block instance
`pages_written`	integer	Number of pages written to disk.
`pages_written_lcp`	integer	Number of pages written by local checkpoints.
`pages_read`	integer	Number of pages read from disk
`log_waits`	integer	Number of page writes waiting for log to be written to disk
`page_requests_direct_return`	integer	Number of requests for pages that were available in buffer
`page_requests_wait_queue`	integer	Number of requests that had to wait for pages to become available in buffer
`page_requests_wait_io`	integer	Number of requests that had to be read from pages on disk (pages were unavailable in buffer)

You can use this table with MySQL Cluster Disk Data tables to determine whether `DiskPageBufferMemory` is sufficiently large to allow data to be read from the buffer rather from disk; minimizing disk seeks can help improve performance of such tables.

You can determine the proportion of reads from `DiskPageBufferMemory` to the total number of reads using a query such as this one, which obtains this ratio as a percentage:

```
SELECT
  node_id,
  100 * page_requests_direct_return /
    (page_requests_direct_return + page_requests_wait_io)
```

```
      AS hit_ratio
FROM ndbinfo.diskpagebuffer;
```

The result from this query should be similar to what is shown here, with one row for each data node in the cluster (in this example, the cluster has 4 data nodes):

```
+---------+-----------+
| node_id | hit_ratio |
+---------+-----------+
|       5 |   97.6744 |
|       6 |   97.6879 |
|       7 |   98.1776 |
|       8 |   98.1343 |
+---------+-----------+
4 rows in set (0.00 sec)
```

`hit_ratio` values approaching 100% indicate that only a very small number of reads are being made from disk rather than from the buffer, which means that Disk Data read performance is approaching an optimum level. If any of these values are less than 95%, this is a strong indicator that the setting for `DiskPageBufferMemory` needs to be increased in the `config.ini` file.

> **Note**
>
> A change in `DiskPageBufferMemory` requires a rolling restart of all of the cluster's data nodes before it takes effect.

7.10.13 The ndbinfo logbuffers Table

The `logbuffer` table provides information on MySQL Cluster log buffer usage.

The following table provides information about the columns in the `logbuffers` table. For each column, the table shows the name, data type, and a brief description.

Column Name	Type	Description
`node_id`	integer	The ID of this data node.
`log_type`	string	Type of log; one of: `REDO` or `DD-UNDO`.
`log_id`	integer	The log ID.
`log_part`	integer	The log part number.
`total`	integer	Total space available for this log.
`used`	integer	Space used by this log.

7.10.14 The ndbinfo logspaces Table

This table provides information about MySQL Cluster log space usage.

The following table provides information about the columns in the `logspaces` table. For each column, the table shows the name, data type, and a brief description.

Column Name	Type	Description
`node_id`	integer	The ID of this data node.
`log_type`	string	Type of log; one of: `REDO` or `DD-UNDO`.
`log_id`	integer	The log ID.
`log_part`	integer	The log part number.
`total`	integer	Total space available for this log.

Column Name	Type	Description
used	integer	Space used by this log.

7.10.15 The ndbinfo membership Table

The membership table describes the view that each data node has of all the others in the cluster, including node group membership, president node, arbitrator, arbitrator successor, arbitrator connection states, and other information.

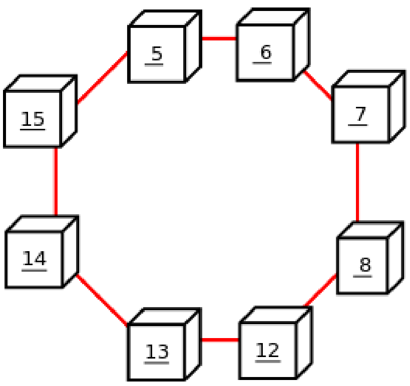
The following table provides information about the columns in the membership table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	This node's node ID
group_id	integer	Node group to which this node belongs
left node	integer	Node ID of the previous node
right_node	integer	Node ID of the next node
president	integer	President's node ID
successor	integer	Node ID of successor to president
succession_order	integer	Order in which this node succeeds to presidency
Conf_HB_order	integer	-
arbitrator	integer	Node ID of arbitrator
arb_ticket	string	Internal identifier used to track arbitration
arb_state	Enumeration (see text)	Arbitration state
arb_connected	Yes or No	Whether this node is connected to the arbitrator
connected_rank1_arbs	List of node IDs	Connected arbitrators of rank 1
connected_rank2_arbs	List of node IDs	Connected arbitrators of rank 1

The node ID and node group ID are the same as reported by `ndb_mgm -e "SHOW"`.

left_node and right_node are defined in terms of a model that connects all data nodes in a circle, in order of their node IDs, similar to the ordering of the numbers on a clock dial, as shown here:

Figure 7.2 Circular Arrangement of MySQL Cluster Nodes



In this example, we have 8 data nodes, numbered 5, 6, 7, 8, 12, 13, 14, and 15, ordered clockwise in a circle. We determine “left” and “right” from the interior of the circle. The node to the left of node 5 is node 15, and the node to the right of node 5 is node 6. You can see all these relationships by running the following query and observing the output:

```
mysql> SELECT node_id,left_node,right_node
    -> FROM ndbinfo.membership;
+---------+-----------+------------+
| node_id | left_node | right_node |
+---------+-----------+------------+
|       5 |        15 |          6 |
|       6 |         5 |          7 |
|       7 |         6 |          8 |
|       8 |         7 |         12 |
|      12 |         8 |         13 |
|      13 |        12 |         14 |
|      14 |        13 |         15 |
|      15 |        14 |          5 |
+---------+-----------+------------+
8 rows in set (0.00 sec)
```

The designations “left” and “right” are used in the event log in the same way.

The `president` node is the node viewed by the current node as responsible for setting an arbitrator (see MySQL Cluster Start Phases). If the president fails or becomes disconnected, the current node expects the node whose ID is shown in the `successor` column to become the new president. The `succession_order` column shows the place in the succession queue that the current node views itself as having.

In a normal MySQL Cluster, all data nodes should see the same node as `president`, and the same node (other than the president) as its `successor`. In addition, the current president should see itself as `1` in the order of succession, the `successor` node should see itself as `2`, and so on.

All nodes should show the same `arb_ticket` values as well as the same `arb_state` values. Possible `arb_state` values are `ARBIT_NULL`, `ARBIT_INIT`, `ARBIT_FIND`, `ARBIT_PREP1`, `ARBIT_PREP2`, `ARBIT_START`, `ARBIT_RUN`, `ARBIT_CHOOSE`, `ARBIT_CRASH`, and `UNKNOWN`.

`arb_connected` shows whether this node is connected to the node shown as this node's `arbitrator`.

The `connected_rank1_arbs` and `connected_rank2_arbs` columns each display a list of 0 or more arbitrators having an `ArbitrationRank` equal to 1, or to 2, respectively.

Note

Both management nodes and API nodes are eligible to become arbitrators.

7.10.16 The ndbinfo memoryusage Table

Querying this table provides information similar to that provided by the `ALL REPORT MemoryUsage` command in the `ndb_mgm` client, or logged by `ALL DUMP 1000`.

The following table provides information about the columns in the `memoryusage` table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
`node_id`	integer	The node ID of this data node.
`memory_type`	string	One of `Data memory` or `Index memory`, or (*MySQL Cluster NDB 7.3.5 and later*) `Long message buffer`.
`used`	integer	Number of bytes currently used for data memory or index memory by this data node.

Column Name	Type	Description
used_pages	integer	Number of pages currently used for data memory or index memory by this data node; see text.
total	integer	Total number of bytes of data memory or index memory available for this data node; see text.
total_pages	integer	Total number of memory pages available for data memory or index memory on this data node; see text.

The total column represents the total amount of memory in bytes available for the given resource (data memory or index memory) on a particular data node. This number should be approximately equal to the setting of the corresponding configuration parameter in the config.ini file.

Suppose that the cluster has 2 data nodes having node IDs 5 and 6, and the config.ini file contains the following:

```
[ndbd default]
DataMemory = 1G
IndexMemory = 1G
```

Suppose also that the value of the LongMessageBuffer configuration parameter is allowed to assume its default (64 MB in MySQL Cluster NDB 7.3.5 and later).

The following query shows approximately the same values:

```
mysql> SELECT node_id, memory_type, total
     > FROM ndbinfo.memoryusage;
+---------+---------------------+------------+
| node_id | memory_type         | total      |
+---------+---------------------+------------+
|       5 | Data memory         | 1073741824 |
|       5 | Index memory        | 1074003968 |
|       5 | Long message buffer |   67108864 |
|       6 | Data memory         | 1073741824 |
|       6 | Index memory        | 1074003968 |
|       6 | Long message buffer |   67108864 |
+---------+---------------------+------------+
6 rows in set (0.00 sec)
```

In this case, the total column values for index memory are slightly higher than the value set of IndexMemory due to internal rounding.

For the used_pages and total_pages columns, resources are measured in pages, which are 32K in size for DataMemory and 8K for IndexMemory. For long message buffer memory, the page size is 256 bytes.

Long message buffer information can be found in this table beginning with MySQL Cluster NDB 7.3.5; in earlier versions of MySQL Cluster, only data memory and index memory were included.

7.10.17 The ndbinfo memory_per_fragment Table

The memory_per_fragment table provides information about the usage of memory by indidivual fragments.

The following table provides information about the columns in the memory_per_fragment table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
fq_name	string	Name of this fragment
parent_fq_name	string	Name of this fragment's parent
type	string	Type of object; see text for possible values
table_id	integer	Table ID for this table
node_id	integer	Node ID for this node
block_instance	integer	Kernel block instance ID
fragment_num	integer	Fragment ID (number)
fixed_elem_alloc_bytes	integer	Number of bytes allocated for fixed-sized elements
fixed_elem_free_bytes	integer	Free bytes remaining in pages allocated to fixed-size elements
fixed_elem_size_bytes	integer	Length of each fixed-size element in bytes
fixed_elem_count	integer	Number of fixed-size elements
fixed_elem_free_rows	decimal	Number of free rows for fixed-size elements
var_elem_alloc_bytes	integer	Number of bytes allocated for variable-size elements
var_elem_free_bytes	integer	Free bytes remaining in pages allocated to variable-size elements
var_elem_count	integer	Number of variable-size elements
hash_index_alloc_bytes	integer	Number of bytes allocated to hash indexes

The `type` column from this table shows the dictionary object type used for this fragment (`Object::Type`, in the NDB API), and can take any one of the values shown in the following list:

- System table
- User table
- Unique hash index
- Hash index
- Unique ordered index
- Ordered index
- Hash index trigger
- Subscription trigger
- Read only constraint
- Index trigger
- Reorganize trigger
- Tablespace
- Log file group

- Data file
- Undo file
- Hash map
- Foreign key definition
- Foreign key parent trigger
- Foreign key child trigger
- Schema transaction

You can also obtain this list by executing `SELECT * FROM ndbinfo.dict_obj_types` in the `mysql` client.

This table was added in MySQL Cluster NDB 7.4.1.

7.10.18 The ndbinfo nodes Table

This table contains information on the status of data nodes. For each data node that is running in the cluster, a corresponding row in this table provides the node's node ID, status, and uptime. For nodes that are starting, it also shows the current start phase.

The following table provides information about the columns in the `nodes` table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
`node_id`	integer	The data node's unique node ID in the cluster.
`uptime`	integer	Time since the node was last started, in seconds.
`status`	string	Current status of the data node; see text for possible values.
`start_phase`	integer	If the data node is starting, the current start phase.
`config_generation`	integer	The version of the cluster configuration file in use on this data node.

The `uptime` column shows the time in seconds that this node has been running since it was last started or restarted. This is a `BIGINT` value. This figure includes the time actually needed to start the node; in other words, this counter starts running the moment that `ndbd` or `ndbmtd` is first invoked; thus, even for a node that has not yet finished starting, `uptime` may show a non-zero value.

The `status` column shows the node's current status. This is one of: `NOTHING`, `CMVMI`, `STARTING`, `STARTED`, `SINGLEUSER`, `STOPPING_1`, `STOPPING_2`, `STOPPING_3`, or `STOPPING_4`. When the status is `STARTING`, you can see the current start phase in the `start_phase` column (see later in this section). `SINGLEUSER` is displayed in the `status` column for all data nodes when the cluster is in single user mode (see Section 7.8, “MySQL Cluster Single User Mode”). Seeing one of the `STOPPING` states does not necessarily mean that the node is shutting down but can mean rather that it is entering a new state; for example, if you put the cluster in single user mode, you can sometimes see data nodes report their state briefly as `STOPPING_2` before the status changes to `SINGLEUSER`.

The `start_phase` column uses the same range of values as those used in the output of the `ndb_mgm` client *`node_id`* `STATUS` command (see Section 7.2, “Commands in the MySQL Cluster Management

Client”). If the node is not currently starting, then this column shows 0. For a listing of MySQL Cluster start phases with descriptions, see Section 7.1, “Summary of MySQL Cluster Start Phases”.

The config_generation column shows which version of the cluster configuration is in effect on each data node. This can be useful when performing a rolling restart of the cluster in order to make changes in configuration parameters. For example, from the output of the following SELECT statement, you can see that node 3 is not yet using the latest version of the cluster configuration (6) although nodes 1, 2, and 4 are doing so:

```
mysql> USE ndbinfo;
Database changed
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+-------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+-------------------+
|       1 |  10462 | STARTED |           0 |                 6 |
|       2 |  10460 | STARTED |           0 |                 6 |
|       3 |  10457 | STARTED |           0 |                 5 |
|       4 |  10455 | STARTED |           0 |                 6 |
+---------+--------+---------+-------------+-------------------+
2 rows in set (0.04 sec)
```

Therefore, for the case just shown, you should restart node 3 to complete the rolling restart of the cluster.

Nodes that are stopped are not accounted for in this table. Suppose that you have a MySQL Cluster with 4 data nodes (node IDs 1, 2, 3 and 4), and all nodes are running normally, then this table contains 4 rows, 1 for each data node:

```
mysql> USE ndbinfo;
Database changed
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+-------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+-------------------+
|       1 |  11776 | STARTED |           0 |                 6 |
|       2 |  11774 | STARTED |           0 |                 6 |
|       3 |  11771 | STARTED |           0 |                 6 |
|       4 |  11769 | STARTED |           0 |                 6 |
+---------+--------+---------+-------------+-------------------+
4 rows in set (0.04 sec)
```

If you shut down one of the nodes, only the nodes that are still running are represented in the output of this SELECT statement, as shown here:

```
ndb_mgm> 2 STOP
Node 2: Node shutdown initiated
Node 2: Node shutdown completed.
Node 2 has shutdown.
```

```
mysql> SELECT * FROM nodes;
+---------+--------+---------+-------------+-------------------+
| node_id | uptime | status  | start_phase | config_generation |
+---------+--------+---------+-------------+-------------------+
|       1 |  11807 | STARTED |           0 |                 6 |
|       3 |  11802 | STARTED |           0 |                 6 |
|       4 |  11800 | STARTED |           0 |                 6 |
+---------+--------+---------+-------------+-------------------+
3 rows in set (0.02 sec)
```

7.10.19 The ndbinfo operations_per_fragment Table

The operations_per_fragment table provides information about the operations performed on indidivual fragments and fragment replicas, as well as about some of the results from these operations.

The following table provides information about the columns in the `operations_per_fragment` table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
fq_name	string	Name of this fragment
parent_fq_name	string	Name of this fragment's parent
type	string	Type of object; see text for possible values
table_id	integer	Table ID for this table
node_id	integer	Node ID for this node
block_instance	integer	Kernel block instance ID
fragment_num	integer	Fragment ID (number)
tot_key_reads	integer	Total number of key reads for this fragment replica
tot_key_inserts	integer	Total number of key inserts for this fragment replica
tot_key_updates	integer	total number of key updates for this fragment replica
tot_key_writes	integer	Total number of key writes for this fragment replica
tot_key_deletes	integer	Total number of key deletes for this fragment replica
tot_key_refs	integer	Number of key operations refused
tot_key_attrinfo_bytes	integer	Total size of all `attrinfo` attributes
tot_key_keyinfo_bytes	integer	Total size of all `keyinfo` attributes
tot_key_prog_bytes	integer	Total size of all interpreted programs carried by `attrinfo` attributes
tot_key_inst_exec	integer	Total number of instructions executed by interpeted programs for key operations
tot_key_bytes_returned	integer	Total size of all data and metadata returned from key read operations
tot_frag_scans	integer	Total number of scans performed on this fragment replica
tot_scan_rows_examined	integer	Total number of rows examined by scans
tot_scan_rows_returned	integer	Total number of rows returned to client
tot_scan_bytes_returned	ineteger	Total size of data and metadata returned to the client
tot_scan_prog_bytes	integer	Total size of interpreted programs for scan operations
tot_scan_bound_bytes	integer	Total size of all bounds used in ordered index scans
tot_scan_inst_exec	integer	Total number of instructions executed for scans
tot_qd_frag_scans	integer	Number of times that scans of this fragment replica have been queued

Column Name	Type	Description
conc_frag_scans	integer	Number of scans currently active on this fragment replica (excluding queued scans)
conc_qd_frag_scans	integer	Number of scans currently queued for this fragment replica
tot_commits	integer	Total number of row changes committed to this fragment replica

The `fq_name` contains the fully qualified name of the schema object to which this fragment replica belongs. This currently has the following formats:

- Base table: - `DbName/def/TblName`
- `BLOB` table: - `DbName/def/NDB$BLOB_BaseTblId_ColNo`
- Ordered index: - `sys/def/BaseTblId/IndexName`
- Unique index: - `sys/def/BaseTblId/IndexName$unique`

The `$unique` suffix shown for unique indexes is added by `mysqld`; for an index created by a different NDB API client application, this may differ, or not be present.

The syntax just shown for fully qualified object names is an internal interface which is subject to change in future releases.

Consider a table `t1` created and modified by the following SQL statements:

```
CREATE DATABASE mydb;
USE mydb;
CREATE TABLE t1 (
  a INT NOT NULL,
  b INT NOT NULL,
  t TEXT NOT NULL,
  PRIMARY KEY (b)
) ENGINE=ndbcluster;
CREATE UNIQUE INDEX ix1 ON t1(b) USING HASH;
```

If `t1` is assigned table ID 11, this yields the `fq_name` values shown here:

- Base table: `mydb/def/t1`
- `BLOB` table: `mydb/def/NDB$BLOB_11_2`
- Ordered index (primary key): `sys/def/11/PRIMARY`
- Unique index: `sys/def/11/ix1$unique`

For indexes or `BLOB` tables, the `parent_fq_name` column contains the `fq_name` of the corresponding base table. For base tables, this column is always `NULL`.

The `type` column shows the schema object type used for this fragment, which can take any one of the values `System table`, `User table`, `Unique hash index`, or `Ordered index`. `BLOB` tables are shown as `User table`.

The `table_id` column value is unique at any given time, but can be reused if the corresponding object has been deleted. The same ID can be seen using the `ndb_show_tables` utility.

The `block_instance` column shows which LDM instance this fragment replica belongs to. The first such instance is always numbered 0.

Since there are typically two replicas, and assuming that this is so, each fragment_num value should appear twice in the table, on two different data nodes from the same node group.

Since NDB does not use single-key access for ordered indexes, the counts for tot_key_reads, tot_key_inserts, tot_key_updates, tot_key_writes, and tot_key_deletes are not incremented by ordered index operations.

> **Note**
>
> When using tot_key_writes, you should keep in mind that a write operation in this context updates the row if the key exists, and inserts a new row otherwise. (One use of this is in the NDB implementation of the REPLACE SQL statement.)

The tot_key_refs column shows the number of key operations refused by the LDM. Generally, such a refusal is due to duplicate keys (inserts), Key not found errors (updates, deletes, and reads), or the operation was rejected by an interpreted program used as a predicate on the row matching the key.

The attrinfo and keyinfo attributes counted by the tot_key_attrinfo_bytes and tot_key_keyinfo_bytes columns are attributes of an LQHKEYREQ signal (see The NDB Communication Protocol) used to initiate a key operation by the LDM. An attrinfo typically contains tuple field values (inserts and updates) or projection specifications (for reads); keyinfo contains the primary or unique key needed to locate a given tuple in this schema object.

The value shown by tot_frag_scans includes both full scans (that examine every row) and scans of subsets. Unique indexes and BLOB tables are never scanned, so this value, like other scan-related counts, is 0 for fragment replicas of these.

tot_scan_rows_examined may display less than the total number of rows in a given fragment replica, since ordered index scans can limited by bounds. In addition, a client may choose to end a scan before all potentially matching rows have been examined; this occurs when using an SQL statement containing a LIMIT or EXISTS clause, for example. tot_scan_rows_returned is always less than or equal to tot_scan_rows_examined.

tot_scan_bytes_returned includes, in the case of pushed joins, projections returned to the DBSPJ block in the NDB kernel.

tot_qd_frag_scans can be effected by the setting for the MaxParallelScansPerFragment data node configuration parameter, which limits the number of scans that may execute concurrently on a single fragment replica.

This table was added in MySQL Cluster NDB 7.4.3.

7.10.20 The ndbinfo resources Table

This table provides information about data node resource availability and usage.

These resources are sometimes known as *super-pools*.

The following table provides information about the columns in the resources table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	The unique node ID of this data node.
resource_name	string	Name of the resource; see text.
reserved	integer	The amount reserved for this resource.
used	integer	The amount actually used by this resource.

Column Name	Type	Description
max	integer	The maximum amount of this resource used, since the node was last started.

The resource_name can be one of RESERVED, DISK_OPERATIONS, DISK_RECORDS, DATA_MEMORY, JOBBUFFER, FILE_BUFFERS, or TRANSPORTER_BUFFERS.

7.10.21 The ndbinfo restart_info Table

The restart_info table contains information about node restart operations. Each entry in the table corresponds to a node restart status report in real time from a data node with the given node ID. Only the most recent report for any given node is shown.

The following table provides information about the columns in the restart_info table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID in the cluster
node_restart_status	VARCHAR(256)	Node status; see text for values. Each of these corresponds to a possible value of node_restart_status_int.
node_restart_status_int	integer	Node status code; see text for values.
secs_to_complete_node_failure	integer	Time in seconds to complete node failure handling
secs_to_allocate_node_id	integer	Time in seconds from node failure completion to allocation of node ID
secs_to_include_in_heartbeat_protocol	integer	Time in seconds from allocation of node ID to inclusion in heartbeat protocol
secs_until_wait_for_ndbcntr_master	integer	Time in seconds from being included in heartbeat protocol until waiting for NDBCNTR master began
secs_wait_for_ndbcntr_master	integer	Time in seconds spent waiting to be accepted by NDBCNTR master for starting
secs_to_get_start_permitted	integer	Time in seconds elapsed from receiving of permission for start from master until all nodes have accepted start of this node
secs_to_wait_for_lcp_for_copy_meta_data	integer	Time in seconds spent waiting for LCP completion before copying meta data
secs_to_copy_meta_data	integer	Time in seconds required to copy metadata from master to newly starting node
secs_to_include_node	integer	Time in seconds waited for GCP and inclusion of all nodes into protocols
secs_starting_node_to_request_local_recovery	integer	Time in seconds that the node just starting spent waiting to request local recovery
secs_for_local_recovery	integer	Time in seconds required for local recovery by node just starting
secs_restore_fragments	integer	Time in seconds required to restore fragments from LCP files
secs_undo_disk_data	integer	Time in seconds required to execute undo log on disk data part of records

Column Name	Type	Description
secs_exec_redo_log	integer	Time in seconds required to execute redo log on all restored fragments
secs_index_rebuild	integer	Time in seconds required to rebuild indexes on restored fragments
secs_to_synchronize_starting_node	integer	Time in seconds required to synchronize starting node from live nodes
secs_wait_lcp_for_restart	integer	Time in seconds required for LCP start and completion before restart was completed
secs_wait_subscription_handover	integer	Time in seconds spent waiting for handover of replication subscriptions
total_restart_secs	integer	Total number of seconds from node failure until node is started again

Defined values for node_restart_status_int and corresponding status names and messages (node_restart_status) are shown in the following table:

node_restart_status_int value	Status	Message (node_restart_status)
0	ALLOCATED_NODE_ID	Allocated node id
1	INCLUDED_IN_HB_PROTOCOL	Included in heartbeat protocol
2	NDBCNTR_START_WAIT	Wait for NDBCNTR master to permit us to start
3	NDBCNTR_STARTED	NDBCNTR master permitted us to start
4	START_PERMITTED	All nodes permitted us to start
5	WAIT_LCP_TO_COPY_DICT	Wait for LCP completion to start copying metadata
6	COPY_DICT_TO_STARTING_NODE	Copying metadata to starting node
7	INCLUDE_NODE_IN_LCP_AND_GCP	Include node in LCP and GCP protocols
8	LOCAL_RECOVERY_STARTED	Restore fragments ongoing
9	COPY_FRAGMENTS_STARTED	Synchronizing starting node with live nodes
10	WAIT_LCP_FOR_RESTART	Wait for LCP to ensure durability
11	WAIT_SUMA_HANDOVER	Wait for handover of subscriptions
12	RESTART_COMPLETED	Restart completed
13	NODE_FAILED	Node failed, failure handling in progress
14	NODE_FAILURE_COMPLETED	Node failure handling completed
15	NODE_GETTING_PERMIT	All nodes permitted us to start
16	NODE_GETTING_INCLUDED	Include node in LCP and GCP protocols
17	NODE_GETTING_SYNCHED	Synchronizing starting node with live nodes
18	NODE_GETTING_LCP_WAITED	[none]
19	NODE_ACTIVE	Restart completed

node_restart_status_int value	Status	Message (node_restart_status)
20	NOT_DEFINED_IN_CLUSTER	[none]
21	NODE_NOT_RESTARTED_YET	Initial state

Status numbers 0 through 12 apply on master nodes only; the remainder of those shown in the table apply to all restarting data nodes. Status numbers 13 and 14 define node failure states; 20 and 21 occur when no information about the restart of a given node is available.

The restart_info table was added in MySQL Cluster NDB 7.4.2.

See also Section 7.1, “Summary of MySQL Cluster Start Phases”.

7.10.22 The ndbinfo server_operations Table

The server_operations table contains entries for all ongoing NDB operations that the current SQL node (MySQL Server) is currently involved in. It effectively is a subset of the cluster_operations table, in which operations for other SQL and API nodes are not shown.

The following table provides information about the columns in the server_operations table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
mysql_connection_id	integer	MySQL Server connection ID
node_id	integer	Node ID
block_instance	integer	Block instance
transid	integer	Transaction ID
operation_type	string	Operation type (see text for possible values)
state	string	Operation state (see text for possible values)
tableid	integer	Table ID
fragmentid	integer	Fragment ID
client_node_id	integer	Client node ID
client_block_ref	integer	Client block reference
tc_node_id	integer	Transaction coordinator node ID
tc_block_no	integer	Transaction coordinator block number
tc_block_instance	integer	Transaction coordinator block instance

The mysql_connection_id is the same as the connection or session ID shown in the output of SHOW PROCESSLIST. It is obtained from the INFORMATION_SCHEMA table NDB_TRANSID_MYSQL_CONNECTION_MAP.

The transaction ID is a unique 64-bit number which can be obtained using the NDB API's getTransactionId() method. (Currently, the MySQL Server does not expose the NDB API transaction ID of an ongoing transaction.)

The operation_type column can take any one of the values READ, READ-SH, READ-EX, INSERT, UPDATE, DELETE, WRITE, UNLOCK, REFRESH, SCAN, SCAN-SH, SCAN-EX, or <unknown>.

The state column can have any one of the values ABORT_QUEUED, ABORT_STOPPED, COMMITTED, COMMIT_QUEUED, COMMIT_STOPPED, COPY_CLOSE_STOPPED, COPY_FIRST_STOPPED, COPY_STOPPED, COPY_TUPKEY, IDLE, LOG_ABORT_QUEUED, LOG_COMMIT_QUEUED, LOG_COMMIT_QUEUED_WAIT_SIGNAL, LOG_COMMIT_WRITTEN,

LOG_COMMIT_WRITTEN_WAIT_SIGNAL, LOG_QUEUED, PREPARED, PREPARED_RECEIVED_COMMIT, SCAN_CHECK_STOPPED, SCAN_CLOSE_STOPPED, SCAN_FIRST_STOPPED, SCAN_RELEASE_STOPPED, SCAN_STATE_USED, SCAN_STOPPED, SCAN_TUPKEY, STOPPED, TC_NOT_CONNECTED, WAIT_ACC, WAIT_ACC_ABORT, WAIT_AI_AFTER_ABORT, WAIT_ATTR, WAIT_SCAN_AI, WAIT_TUP, WAIT_TUPKEYINFO, WAIT_TUP_COMMIT, or WAIT_TUP_TO_ABORT. (If the MySQL Server is running with ndbinfo_show_hidden enabled, you can view this list of states by selecting from the ndb$dblqh_tcconnect_state table, which is normally hidden.)

You can obtain the name of an NDB table from its table ID by checking the output of ndb_show_tables.

The fragid is the same as the partition number seen in the output of ndb_desc --extra-partition-info (short form -p).

In client_node_id and client_block_ref, client refers to a MySQL Cluster API or SQL node (that is, an NDB API client or a MySQL Server attached to the cluster).

7.10.23 The ndbinfo server_transactions Table

The server_transactions table is subset of the cluster_transactions table, but includes only those transactions in which the current SQL node (MySQL Server) is a participant, while including the relevant connection IDs.

The following table provides information about the columns in the server_transactions table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
mysql_connection_id	integer	MySQL Server connection ID
node_id	integer	Transaction coordinator node ID
block_instance	integer	Transaction coordinator block instance
transid	integer	Transaction ID
state	string	Operation state (see text for possible values)
count_operations	integer	Number of stateful operations in the transaction
outstanding_operations	integer	Operations still being executed by local data management layer (LQH blocks)
inactive_seconds	integer	Time spent waiting for API
client_node_id	integer	Client node ID
client_block_ref	integer	Client block reference

The mysql_connection_id is the same as the connection or session ID shown in the output of SHOW PROCESSLIST. It is obtained from the INFORMATION_SCHEMA table NDB_TRANSID_MYSQL_CONNECTION_MAP.

The transaction ID is a unique 64-bit number which can be obtained using the NDB API's getTransactionId() method. (Currently, the MySQL Server does not expose the NDB API transaction ID of an ongoing transaction.)

The state column can have any one of the values CS_ABORTING, CS_COMMITTING, CS_COMMIT_SENT, CS_COMPLETE_SENT, CS_COMPLETING, CS_CONNECTED, CS_DISCONNECTED, CS_FAIL_ABORTED, CS_FAIL_ABORTING, CS_FAIL_COMMITTED, CS_FAIL_COMMITTING, CS_FAIL_COMPLETED, CS_FAIL_PREPARED, CS_PREPARE_TO_COMMIT, CS_RECEIVING, CS_REC_COMMITTING, CS_RESTART, CS_SEND_FIRE_TRIG_REQ, CS_STARTED, CS_START_COMMITTING, CS_START_SCAN, CS_WAIT_ABORT_CONF, CS_WAIT_COMMIT_CONF,

CS_WAIT_COMPLETE_CONF, CS_WAIT_FIRE_TRIG_REQ. (If the MySQL Server is running with ndbinfo_show_hidden enabled, you can view this list of states by selecting from the ndb $dbtc_apiconnect_state table, which is normally hidden.)

In client_node_id and client_block_ref, client refers to a MySQL Cluster API or SQL node (that is, an NDB API client or a MySQL Server attached to the cluster).

7.10.24 The ndbinfo threadblocks Table

The threadblocks table associates data nodes, threads, and instances of NDB kernel blocks.

The following table provides information about the columns in the threadblocks table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID
thr_no	integer	Thread ID
block_name	string	Block name
block_instance	integer	Block instance number

The block_name is one of the values found in the block_name column when selecting from the ndbinfo.blocks table. Although the list of possible values is static for a given MySQL Cluster release, the list may vary between releases.

7.10.25 The ndbinfo threadstat Table

The threadstat table provides a rough snapshot of statistics for threads running in the NDB kernel.

The following table provides information about the columns in the threadstat table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	Node ID
thr_no	integer	Thread ID
thr_nm	string	Thread name
c_loop	string	Number of loops in main loop
c_exec	string	Number of signals executed
c_wait	string	Number of times waiting for additional input
c_l_sent_prioa	integer	Number of priority A signals sent to own node
c_l_sent_priob	integer	Number of priority B signals sent to own node
c_r_sent_prioa	integer	Number of priority A signals sent to remote node
c_r_sent_priob	integer	Number of priority B signals sent to remote node
os_tid	integer	OS thread ID
os_now	integer	OS time (ms)
os_ru_utime	integer	OS user CPU time (µs)

Column Name	Type	Description
os_ru_stime	integer	OS system CPU time (µs)
os_ru_minflt	integer	OS page reclaims (soft page faults)
os_ru_majflt	integer	OS page faults (hard page faults)
os_ru_nvcsw	integer	OS voluntary context switches
os_ru_nivcsw	integer	OS involuntary context switches

`os_time` uses the system `gettimeofday()` call.

The values of the `os_ru_utime`, `os_ru_stime`, `os_ru_minflt`, `os_ru_majflt`, `os_ru_nvcsw`, and `os_ru_nivcsw` columns are obtained using the system `getrusage()` call, or the equivalent.

Since this table contains counts taken at a given point in time, for best results it is necessary to query this table periodically and store the results in an intermediate table or tables. The MySQL Server's Event Scheduler can be employed to automate such monitoring. For more information, see Using the Event Scheduler.

7.10.26 The ndbinfo transporters Table

This table contains information about NDB transporters.

The following table provides information about the columns in the `transporters` table. For each column, the table shows the name, data type, and a brief description. Additional information can be found in the notes following the table.

Column Name	Type	Description
node_id	integer	This data node's unique node ID in the cluster.
remote_node_id	integer	The remote data node's node ID.
status	string	Status of the connection.
remote_address	string	Name or IP address of the remote host
bytes_sent	integer	Number of bytes sent using this connection
bytes_received		Number of bytes received using this connection
connect_count		Number of times connection established on this transporter
overloaded		1 if this transporter is currently overloaded, otherwise 0
overload_count		Number of times this transporter has entered overload state since connecting
slowdown		1 if this transporter is in scan slowdown state, otherwise 0
slowdown_count		Number of times this transporter has entered scan slowdown state since connecting

For each running data node in the cluster, the `transporters` table displays a row showing the status of each of that node's connections with all nodes in the cluster, *including itself*. This information is shown in the table's *status* column, which can have any one of the following values: `CONNECTING`, `CONNECTED`, `DISCONNECTING`, or `DISCONNECTED`.

Connections to API and management nodes which are configured but not currently connected to the cluster are shown with status `DISCONNECTED`. Rows where the `node_id` is that of a data nodes which is not currently connected are not shown in this table. (This is similar omission of disconnected nodes in the ndbinfo.nodes table.

The remote_address is the host name or address for the node whose ID is shown in the remote_node_id column. The bytes_sent from this node and bytes_received by this node are the numbers, respectively, of bytes sent and received by the node using this connection since it was established. For nodes whose status is CONNECTING or DISCONNECTED, these columns always display 0.

The connect_count, overloaded, overload_count ,slowdown, and slowdown_count counters are reset on connection, and retain their values after the remote node disconnects. The bytes_send and bytes_received counters are also reset on connection, and so retain their values following disconnection (until the next connection resets them).

Assume you have a 5-node cluster consisting of 2 data nodes, 2 SQL nodes, and 1 management node, as shown in the output of the SHOW command in the ndb_mgm client:

```
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @10.100.10.1  (5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=2    @10.100.10.2  (5.6.27-ndb-7.4.9, Nodegroup: 0)
[ndb_mgmd(MGM)] 1 node(s)
id=10   @10.100.10.10  (5.6.27-ndb-7.4.9)
[mysqld(API)]   2 node(s)
id=20   @10.100.10.20  (5.6.27-ndb-7.4.9)
id=21   @10.100.10.21  (5.6.27-ndb-7.4.9)
```

There are 10 rows in the transporters table—5 for the first data node, and 5 for the second—assuming that all data nodes are running, as shown here:

```
mysql> SELECT node_id, remote_node_id, status
    ->   FROM ndbinfo.transporters;
+---------+----------------+---------------+
| node_id | remote_node_id | status        |
+---------+----------------+---------------+
|       1 |              1 | DISCONNECTED  |
|       1 |              2 | CONNECTED     |
|       1 |             10 | CONNECTED     |
|       1 |             20 | CONNECTED     |
|       1 |             21 | CONNECTED     |
|       2 |              1 | CONNECTED     |
|       2 |              2 | DISCONNECTED  |
|       2 |             10 | CONNECTED     |
|       2 |             20 | CONNECTED     |
|       2 |             21 | CONNECTED     |
+---------+----------------+---------------+
10 rows in set (0.04 sec)
```

If you shut down one of the data nodes in this cluster using the command 2 STOP in the ndb_mgm client, then repeat the previous query (again using the mysql client), this table now shows only 5 rows—1 row for each connection from the remaining management node to another node, including both itself and the data node that is currently offline—and displays CONNECTING for the status of each remaining connection to the data node that is currently offline, as shown here:

```
mysql> SELECT node_id, remote_node_id, status
    ->   FROM ndbinfo.transporters;
+---------+----------------+---------------+
| node_id | remote_node_id | status        |
+---------+----------------+---------------+
|       1 |              1 | DISCONNECTED  |
|       1 |              2 | CONNECTING    |
```

```
|        1 |               10 | CONNECTED      |
|        1 |               20 | CONNECTED      |
|        1 |               21 | CONNECTED      |
+---------+----------------+--------------+
5 rows in set (0.02 sec)
```

7.11 MySQL Cluster Security Issues

This section discusses security considerations to take into account when setting up and running MySQL Cluster.

Topics covered in this section include the following:

- MySQL Cluster and network security issues
- Configuration issues relating to running MySQL Cluster securely
- MySQL Cluster and the MySQL privilege system
- MySQL standard security procedures as applicable to MySQL Cluster

7.11.1 MySQL Cluster Security and Networking Issues

In this section, we discuss basic network security issues as they relate to MySQL Cluster. It is extremely important to remember that MySQL Cluster “out of the box” is not secure; you or your network administrator must take the proper steps to ensure that your cluster cannot be compromised over the network.

Cluster communication protocols are inherently insecure, and no encryption or similar security measures are used in communications between nodes in the cluster. Because network speed and latency have a direct impact on the cluster's efficiency, it is also not advisable to employ SSL or other encryption to network connections between nodes, as such schemes will effectively slow communications.

It is also true that no authentication is used for controlling API node access to a MySQL Cluster. As with encryption, the overhead of imposing authentication requirements would have an adverse impact on Cluster performance.

In addition, there is no checking of the source IP address for either of the following when accessing the cluster:

- SQL or API nodes using “free slots” created by empty `[mysqld]` or `[api]` sections in the `config.ini` file

  This means that, if there are any empty `[mysqld]` or `[api]` sections in the `config.ini` file, then any API nodes (including SQL nodes) that know the management server's host name (or IP address) and port can connect to the cluster and access its data without restriction. (See Section 7.11.2, “MySQL Cluster and MySQL Privileges”, for more information about this and related issues.)

> **Note**
>
> You can exercise some control over SQL and API node access to the cluster by specifying a `HostName` parameter for all `[mysqld]` and `[api]` sections in the `config.ini` file. However, this also means that, should you wish to connect an API node to the cluster from a previously unused host, you need to add an `[api]` section containing its host name to the `config.ini` file.
>
> More information is available elsewhere in this chapter about the `HostName` parameter. Also see Section 5.1, “Quick Test Setup of MySQL Cluster”, for configuration examples using `HostName` with API nodes.

- Any `ndb_mgm` client

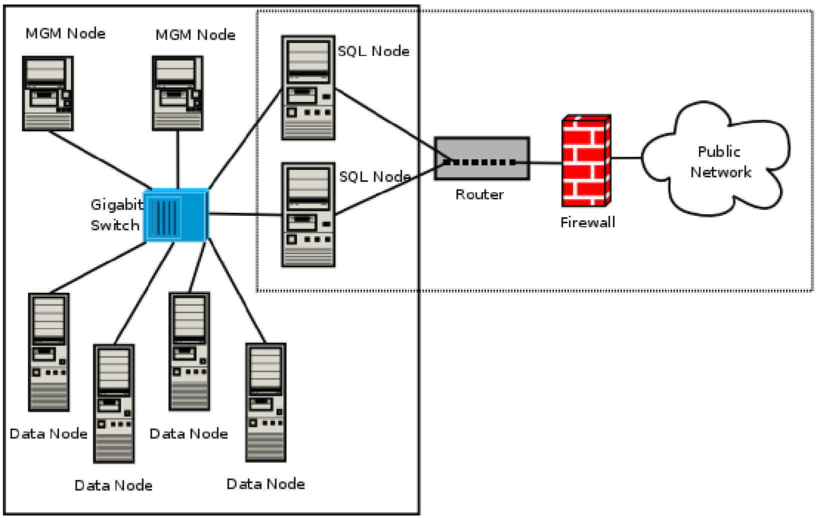
  This means that any cluster management client that is given the management server's host name (or IP address) and port (if not the standard port) can connect to the cluster and execute any management client command. This includes commands such as `ALL STOP` and `SHUTDOWN`.

For these reasons, it is necessary to protect the cluster on the network level. The safest network configuration for Cluster is one which isolates connections between Cluster nodes from any other network communications. This can be accomplished by any of the following methods:

1. Keeping Cluster nodes on a network that is physically separate from any public networks. This option is the most dependable; however, it is the most expensive to implement.

   We show an example of a MySQL Cluster setup using such a physically segregated network here:

   Figure 7.3 MySQL Cluster with Hardware Firewall

   This setup has two networks, one private (solid box) for the Cluster management servers and data nodes, and one public (dotted box) where the SQL nodes reside. (We show the management and data nodes connected using a gigabit switch since this provides the best performance.) Both networks are protected from the outside by a hardware firewall, sometimes also known as a *network-based firewall*.

   This network setup is safest because no packets can reach the cluster's management or data nodes from outside the network—and none of the cluster's internal communications can reach the outside—without going through the SQL nodes, as long as the SQL nodes do not permit any packets to be forwarded. This means, of course, that all SQL nodes must be secured against hacking attempts.

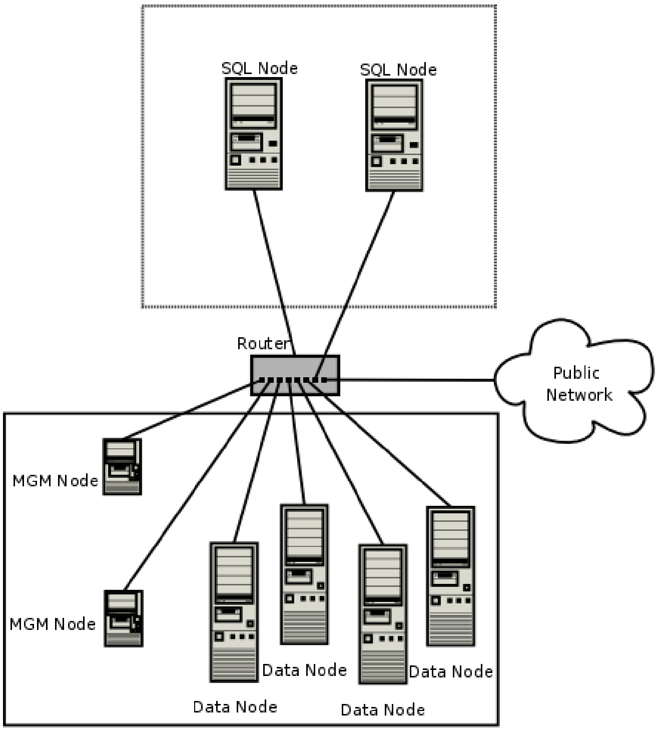
   > **Important**
   >
   > With regard to potential security vulnerabilities, an SQL node is no different from any other MySQL server. See Making MySQL Secure Against Attackers, for a description of techniques you can use to secure MySQL servers.

2. Using one or more software firewalls (also known as *host-based firewalls*) to control which packets pass through to the cluster from portions of the network that do not require access to it. In this type of setup, a software firewall must be installed on every host in the cluster which might otherwise be accessible from outside the local network.

   The host-based option is the least expensive to implement, but relies purely on software to provide protection and so is the most difficult to keep secure.

This type of network setup for MySQL Cluster is illustrated here:

Figure 7.4 MySQL Cluster with Software Firewalls



Using this type of network setup means that there are two zones of MySQL Cluster hosts. Each cluster host must be able to communicate with all of the other machines in the cluster, but only those hosting SQL nodes (dotted box) can be permitted to have any contact with the outside, while those in the zone containing the data nodes and management nodes (solid box) must be isolated from any machines that are not part of the cluster. Applications using the cluster and user of those applications must *not* be permitted to have direct access to the management and data node hosts.

To accomplish this, you must set up software firewalls that limit the traffic to the type or types shown in the following table, according to the type of node that is running on each cluster host computer:

Type of Node to be Accessed	Traffic to Permit
SQL or API node	• It originates from the IP address of a management or data node (using any TCP or UDP port). • It originates from within the network in which the cluster resides and is on the port that your application is using.
Data node or Management node	• It originates from the IP address of a management or data node (using any TCP or UDP port). • It originates from the IP address of an SQL or API node.

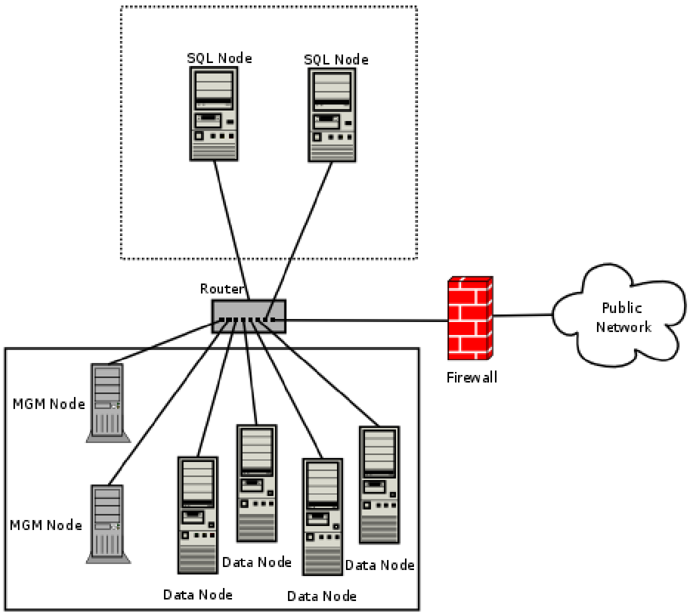
Any traffic other than that shown in the table for a given node type should be denied.

The specifics of configuring a firewall vary from firewall application to firewall application, and are beyond the scope of this Manual. `iptables` is a very common and reliable firewall application, which is often used with APF as a front end to make configuration easier. You can (and should) consult the documentation for the software firewall that you employ, should you choose to implement a MySQL Cluster network setup of this type, or of a “mixed” type as discussed under the next item.

3. It is also possible to employ a combination of the first two methods, using both hardware and software to secure the cluster—that is, using both network-based and host-based firewalls. This is between the first two schemes in terms of both security level and cost. This type of network setup keeps the cluster behind the hardware firewall, but permits incoming packets to travel beyond the router connecting all cluster hosts to reach the SQL nodes.

   One possible network deployment of a MySQL Cluster using hardware and software firewalls in combination is shown here:

   Figure 7.5 MySQL Cluster with a Combination of Hardware and Software Firewalls



   In this case, you can set the rules in the hardware firewall to deny any external traffic except to SQL nodes and API nodes, and then permit traffic to them only on the ports required by your application.

Whatever network configuration you use, remember that your objective from the viewpoint of keeping the cluster secure remains the same—to prevent any unessential traffic from reaching the cluster while ensuring the most efficient communication between the nodes in the cluster.

Because MySQL Cluster requires large numbers of ports to be open for communications between nodes, the recommended option is to use a segregated network. This represents the simplest way to prevent unwanted traffic from reaching the cluster.

> **Note**
>
> If you wish to administer a MySQL Cluster remotely (that is, from outside the local network), the recommended way to do this is to use `ssh` or another secure login shell to access an SQL node host. From this host, you can then run the management client to access the management server safely, from within the Cluster's own local network.
>
> Even though it is possible to do so in theory, it is *not* recommended to use `ndb_mgm` to manage a Cluster directly from outside the local network on which the Cluster is running. Since neither authentication nor encryption takes place between the management client and the management server, this represents an extremely insecure means of managing the cluster, and is almost certain to be compromised sooner or later.

7.11.2 MySQL Cluster and MySQL Privileges

In this section, we discuss how the MySQL privilege system works in relation to MySQL Cluster and the implications of this for keeping a MySQL Cluster secure.

Standard MySQL privileges apply to MySQL Cluster tables. This includes all MySQL privilege types (`SELECT` privilege, `UPDATE` privilege, `DELETE` privilege, and so on) granted on the database, table, and column level. As with any other MySQL Server, user and privilege information is stored in the `mysql` system database. The SQL statements used to grant and revoke privileges on `NDB` tables, databases containing such tables, and columns within such tables are identical in all respects with the `GRANT` and `REVOKE` statements used in connection with database objects involving any (other) MySQL storage engine. The same thing is true with respect to the `CREATE USER` and `DROP USER` statements.

It is important to keep in mind that, by default, the MySQL grant tables use the `MyISAM` storage engine. Because of this, those tables are not normally duplicated or shared among MySQL servers acting as SQL nodes in a MySQL Cluster. In other words, changes in users and their privileges do not automatically propagate between SQL nodes by default. If you wish, you can enable automatic distribution of MySQL users and privileges across MySQL Cluster SQL nodes; see Section 7.14, "Distributed MySQL Privileges for MySQL Cluster", for details.

Conversely, because there is no way in MySQL to deny privileges (privileges can either be revoked or not granted in the first place, but not denied as such), there is no special protection for `NDB` tables on one SQL node from users that have privileges on another SQL node; (This is true even if you are not using automatic distribution of user privileges. The definitive example of this is the MySQL `root` account, which can perform any action on any database object. In combination with empty `[mysqld]` or `[api]` sections of the `config.ini` file, this account can be especially dangerous. To understand why, consider the following scenario:

- The `config.ini` file contains at least one empty `[mysqld]` or `[api]` section. This means that the MySQL Cluster management server performs no checking of the host from which a MySQL Server (or other API node) accesses the MySQL Cluster.
- There is no firewall, or the firewall fails to protect against access to the MySQL Cluster from hosts external to the network.
- The host name or IP address of the MySQL Cluster's management server is known or can be determined from outside the network.

If these conditions are true, then anyone, anywhere can start a MySQL Server with `--ndbcluster --ndb-connectstring=management_host` and access this MySQL Cluster. Using the MySQL `root` account, this person can then perform the following actions:

- Execute metadata statements such as `SHOW DATABASES` statement (to obtain a list of all `NDB` databases on the server) or `SHOW TABLES FROM some_ndb_database` statement to obtain a list of all `NDB` tables in a given database
- Run any legal MySQL statements on any of the discovered tables, such as:
  - `SELECT * FROM some_table` to read all the data from any table
  - `DELETE FROM some_table` to delete all the data from a table
  - `DESCRIBE some_table` or `SHOW CREATE TABLE some_table` to determine the table schema
  - `UPDATE some_table SET column1 = some_value` to fill a table column with "garbage" data; this could actually cause much greater damage than simply deleting all the data

    More insidious variations might include statements like these:

```
UPDATE some_table SET an_int_column = an_int_column + 1
```

or

```
UPDATE some_table SET a_varchar_column = REVERSE(a_varchar_column)
```

Such malicious statements are limited only by the imagination of the attacker.

The only tables that would be safe from this sort of mayhem would be those tables that were created using storage engines other than `NDB`, and so not visible to a “rogue” SQL node.

A user who can log in as `root` can also access the `INFORMATION_SCHEMA` database and its tables, and so obtain information about databases, tables, stored routines, scheduled events, and any other database objects for which metadata is stored in `INFORMATION_SCHEMA`.

It is also a very good idea to use different passwords for the `root` accounts on different MySQL Cluster SQL nodes unless you are using distributed privileges.

In sum, you cannot have a safe MySQL Cluster if it is directly accessible from outside your local network.

> **Important**
>
> *Never leave the MySQL root account password empty*. This is just as true when running MySQL as a MySQL Cluster SQL node as it is when running it as a standalone (non-Cluster) MySQL Server, and should be done as part of the MySQL installation process before configuring the MySQL Server as an SQL node in a MySQL Cluster.

If you wish to employ MySQL Cluster's distributed privilege capabilities, you should not simply convert the system tables in the `mysql` database to use the `NDB` storage engine manually. Use the stored procedure provided for this purpose instead; see Section 7.14, “Distributed MySQL Privileges for MySQL Cluster”.

Otherwise, if you need to synchronize `mysql` system tables between SQL nodes, you can use standard MySQL replication to do so, or employ a script to copy table entries between the MySQL servers.

Summary. The most important points to remember regarding the MySQL privilege system with regard to MySQL Cluster are listed here:

1. Users and privileges established on one SQL node do not automatically exist or take effect on other SQL nodes in the cluster. Conversely, removing a user or privilege on one SQL node in the cluster does not remove the user or privilege from any other SQL nodes.

2. You can distribute MySQL users and privileges among SQL nodes using the SQL script, and the stored procedures it contains, that are supplied for this purpose in the MySQL Cluster distribution.

3. Once a MySQL user is granted privileges on an `NDB` table from one SQL node in a MySQL Cluster, that user can “see” any data in that table regardless of the SQL node from which the data originated, even if you are not using privilege distribution.

7.11.3 MySQL Cluster and MySQL Security Procedures

In this section, we discuss MySQL standard security procedures as they apply to running MySQL Cluster.

In general, any standard procedure for running MySQL securely also applies to running a MySQL Server as part of a MySQL Cluster. First and foremost, you should always run a MySQL Server as the `mysql` system user; this is no different from running MySQL in a standard (non-Cluster) environment.

The mysql system account should be uniquely and clearly defined. Fortunately, this is the default behavior for a new MySQL installation. You can verify that the mysqld process is running as the system user mysql by using the system command such as the one shown here:

```
shell> ps aux | grep mysql
root      10467  0.0  0.1   3616  1380 pts/3    S    11:53   0:00 \
  /bin/sh ./mysqld_safe --ndbcluster --ndb-connectstring=localhost:1186
mysql     10512  0.2  2.5  58528 26636 pts/3    Sl   11:53   0:00 \
  /usr/local/mysql/libexec/mysqld --basedir=/usr/local/mysql \
  --datadir=/usr/local/mysql/var --user=mysql --ndbcluster \
  --ndb-connectstring=localhost:1186 --pid-file=/usr/local/mysql/var/mothra.pid \
  --log-error=/usr/local/mysql/var/mothra.err
jon       10579  0.0  0.0   2736   688 pts/0    S+   11:54   0:00 grep mysql
```

If the mysqld process is running as any other user than mysql, you should immediately shut it down and restart it as the mysql user. If this user does not exist on the system, the mysql user account should be created, and this user should be part of the mysql user group; in this case, you should also make sure that the MySQL data directory on this system (as set using the --datadir option for mysqld) is owned by the mysql user, and that the SQL node's my.cnf file includes user=mysql in the [mysqld] section. Alternatively, you can start the MySQL server process with --user=mysql on the command line, but it is preferable to use the my.cnf option, since you might forget to use the command-line option and so have mysqld running as another user unintentionally. The mysqld_safe startup script forces MySQL to run as the mysql user.

> **Important**
>
> Never run mysqld as the system root user. Doing so means that potentially any file on the system can be read by MySQL, and thus—should MySQL be compromised—by an attacker.

As mentioned in the previous section (see Section 7.11.2, “MySQL Cluster and MySQL Privileges”), you should always set a root password for the MySQL Server as soon as you have it running. You should also delete the anonymous user account that is installed by default. You can accomplish these tasks using the following statements:

```
shell> mysql -u root
mysql> UPDATE mysql.user
    ->     SET Password=PASSWORD('secure_password')
    ->     WHERE User='root';
mysql> DELETE FROM mysql.user
    ->     WHERE User='';
mysql> FLUSH PRIVILEGES;
```

Be very careful when executing the DELETE statement not to omit the WHERE clause, or you risk deleting *all* MySQL users. Be sure to run the FLUSH PRIVILEGES statement as soon as you have modified the mysql.user table, so that the changes take immediate effect. Without FLUSH PRIVILEGES, the changes do not take effect until the next time that the server is restarted.

> **Note**
>
> Many of the MySQL Cluster utilities such as ndb_show_tables, ndb_desc, and ndb_select_all also work without authentication and can reveal table names, schemas, and data. By default these are installed on Unix-style systems with the permissions wxr-xr-x (755), which means they can be executed by any user that can access the mysql/bin directory.
>
> See Chapter 6, *MySQL Cluster Programs*, for more information about these utilities.

7.12 MySQL Cluster Disk Data Tables

It is possible to store the nonindexed columns of NDB tables on disk, rather than in RAM.

As part of implementing MySQL Cluster Disk Data work, a number of improvements were made in MySQL Cluster for the efficient handling of very large amounts (terabytes) of data during node recovery and restart. These include a “no-steal” algorithm for synchronizing a starting node with very large data sets. For more information, see the paper *Recovery Principles of MySQL Cluster 5.1*, by MySQL Cluster developers Mikael Ronström and Jonas Oreland.

MySQL Cluster Disk Data performance can be influenced by a number of configuration parameters. For information about these parameters and their effects, see *MySQL Cluster Disk Data configuration parameters* and *MySQL Cluster Disk Data storage and `GCP Stop` errors*

The performance of a MySQL Cluster that uses Disk Data storage can also be greatly improved by separating data node file systems from undo log files and tablespace data files, which can be done using symbolic links. For more information, see Section 7.12.2, “Using Symbolic Links with Disk Data Objects”.

7.12.1 MySQL Cluster Disk Data Objects

MySQL Cluster Disk Data storage is implemented using a number of *Disk Data objects*. These include the following:

- *Tablespaces* act as containers for other Disk Data objects.
- *Undo log files* undo information required for rolling back transactions.
- One or more undo log files are assigned to a *log file group*, which is then assigned to a tablespace.
- *Data files* store Disk Data table data. A data file is assigned directly to a tablespace.

Undo log files and data files are actual files in the file system of each data node; by default they are placed in `ndb_node_id_fs` in the `DataDir` specified in the MySQL Cluster `config.ini` file, and where `node_id` is the data node's node ID. It is possible to place these elsewhere by specifying either an absolute or relative path as part of the filename when creating the undo log or data file. Statements that create these files are shown later in this section.

MySQL Cluster tablespaces and log file groups are not implemented as files.

> **Important**
>
> Although not all Disk Data objects are implemented as files, they all share the same namespace. This means that *each Disk Data object* must be uniquely named (and not merely each Disk Data object of a given type). For example, you cannot have a tablespace and a log file group both named `dd1`.

Assuming that you have already set up a MySQL Cluster with all nodes (including management and SQL nodes), the basic steps for creating a MySQL Cluster table on disk are as follows:

1. Create a log file group, and assign one or more undo log files to it (an undo log file is also sometimes referred to as an *undofile*).

   > **Note**
   >
   > Undo log files are necessary only for Disk Data tables; they are not used for `NDBCLUSTER` tables that are stored only in memory.

2. Create a tablespace; assign the log file group, as well as one or more data files, to the tablespace.
3. Create a Disk Data table that uses this tablespace for data storage.

Each of these tasks can be accomplished using SQL statements in the `mysql` client or other MySQL client application, as shown in the example that follows.

1. We create a log file group named `lg_1` using `CREATE LOGFILE GROUP`. This log file group is to be made up of two undo log files, which we name `undo_1.log` and `undo_2.log`, whose initial sizes are 16 MB and 12 MB, respectively. (The default initial size for an undo log file is 128 MB.) Optionally, you can also specify a size for the log file group's undo buffer, or permit it to assume the default value of 8 MB. In this example, we set the UNDO buffer's size at 2 MB. A log file group must be created with an undo log file; so we add `undo_1.log` to `lg_1` in this `CREATE LOGFILE GROUP` statement:

```
CREATE LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_1.log'
    INITIAL_SIZE 16M
    UNDO_BUFFER_SIZE 2M
    ENGINE NDBCLUSTER;
```

   To add `undo_2.log` to the log file group, use the following `ALTER LOGFILE GROUP` statement:

```
ALTER LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_2.log'
    INITIAL_SIZE 12M
    ENGINE NDBCLUSTER;
```

   Some items of note:

   - The `.log` file extension used here is not required. We use it merely to make the log files easily recognisable.
   - Every `CREATE LOGFILE GROUP` and `ALTER LOGFILE GROUP` statement must include an `ENGINE` clause. In MySQL Cluster NDB 7.3 and later, the only permitted values for this clause are `NDBCLUSTER` and `NDB`.

     Important

     There can exist at most one log file group in the same MySQL Cluster at any given time.

   - When you add an undo log file to a log file group using `ADD UNDOFILE 'filename'`, a file with the name *`filename`* is created in the `ndb_node_id_fs` directory within the `DataDir` of each data node in the cluster, where *`node_id`* is the node ID of the data node. Each undo log file is of the size specified in the SQL statement. For example, if a MySQL Cluster has 4 data nodes, then the `ALTER LOGFILE GROUP` statement just shown creates 4 undo log files, 1 each on in the data directory of each of the 4 data nodes; each of these files is named `undo_2.log` and each file is 12 MB in size.
   - `UNDO_BUFFER_SIZE` is limited by the amount of system memory available.
   - For more information about the `CREATE LOGFILE GROUP` statement, see CREATE LOGFILE GROUP Syntax. For more information about `ALTER LOGFILE GROUP`, see ALTER LOGFILE GROUP Syntax.

2. Now we can create a tablespace, which contains files to be used by MySQL Cluster Disk Data tables for storing their data. A tablespace is also associated with a particular log file group. When creating a new tablespace, you must specify the log file group which it is to use for undo logging; you must also specify a data file. You can add more data files to the tablespace after the tablespace is created; it is also possible to drop data files from a tablespace (an example of dropping data files is provided later in this section).

   Assume that we wish to create a tablespace named `ts_1` which uses `lg_1` as its log file group. This tablespace is to contain two data files named `data_1.dat` and `data_2.dat`, whose initial sizes are 32 MB and 48 MB, respectively. (The default value for `INITIAL_SIZE` is 128 MB.) We can do this using two SQL statements, as shown here:

```
CREATE TABLESPACE ts_1
    ADD DATAFILE 'data_1.dat'
    USE LOGFILE GROUP lg_1
    INITIAL_SIZE 32M
    ENGINE NDBCLUSTER;
ALTER TABLESPACE ts_1
    ADD DATAFILE 'data_2.dat'
    INITIAL_SIZE 48M
    ENGINE NDBCLUSTER;
```

The `CREATE TABLESPACE` statement creates a tablespace `ts_1` with the data file `data_1.dat`, and associates `ts_1` with log file group `lg_1`. The `ALTER TABLESPACE` adds the second data file (`data_2.dat`).

Some items of note:

- As is the case with the `.log` file extension used in this example for undo log files, there is no special significance for the `.dat` file extension; it is used merely for easy recognition of data files.

- When you add a data file to a tablespace using `ADD DATAFILE '`*`filename`*`'`, a file with the name *`filename`* is created in the `ndb_`*`node_id`*`_fs` directory within the `DataDir` of each data node in the cluster, where *`node_id`* is the node ID of the data node. Each data file is of the size specified in the SQL statement. For example, if a MySQL Cluster has 4 data nodes, then the `ALTER TABLESPACE` statement just shown creates 4 data files, 1 each in the data directory of each of the 4 data nodes; each of these files is named `data_2.dat` and each file is 48 MB in size.

- All `CREATE TABLESPACE` and `ALTER TABLESPACE` statements must contain an `ENGINE` clause; only tables using the same storage engine as the tablespace can be created in the tablespace. In MySQL MySQL Cluster NDB 7.3, the only permitted values for this clause are `NDBCLUSTER` and `NDB`.

- For more information about the `CREATE TABLESPACE` and `ALTER TABLESPACE` statements, see CREATE TABLESPACE Syntax, and ALTER TABLESPACE Syntax.

3. Now it is possible to create a table whose nonindexed columns are stored on disk in the tablespace `ts_1`:

```
CREATE TABLE dt_1 (
    member_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    dob DATE NOT NULL,
    joined DATE NOT NULL,
    INDEX(last_name, first_name)
    )
    TABLESPACE ts_1 STORAGE DISK
    ENGINE NDBCLUSTER;
```

The `TABLESPACE ... STORAGE DISK` option tells the `NDBCLUSTER` storage engine to use tablespace `ts_1` for disk data storage.

> **Note**
>
> It is also possible to specify whether an individual column is stored on disk or in memory by using a `STORAGE` clause as part of the column's definition in a `CREATE TABLE` or `ALTER TABLE` statement. `STORAGE DISK` causes the column to be stored on disk, and `STORAGE MEMORY` causes in-memory storage to be used. See CREATE TABLE Syntax, for more information.

Once table ts_1 has been created as shown, you can perform INSERT, SELECT, UPDATE, and DELETE statements on it just as you would with any other MySQL table.

For table dt_1 as it has been defined here, only the dob and joined columns are stored on disk. This is because there are indexes on the id, last_name, and first_name columns, and so data belonging to these columns is stored in RAM. In MySQL Cluster NDB 7.3 and MySQL Cluster NDB 7.4, only nonindexed columns can be held on disk; indexes and indexed column data continue to be stored in memory. This tradeoff between the use of indexes and conservation of RAM is something you must keep in mind as you design Disk Data tables.

Performance note. The performance of a cluster using Disk Data storage is greatly improved if Disk Data files are kept on a separate physical disk from the data node file system. This must be done for each data node in the cluster to derive any noticeable benefit.

You may use absolute and relative file system paths with ADD UNDOFILE and ADD DATAFILE. Relative paths are calculated relative to the data node's data directory. You may also use symbolic links; see Section 7.12.2, “Using Symbolic Links with Disk Data Objects”, for more information and examples.

A log file group, a tablespace, and any Disk Data tables using these must be created in a particular order. The same is true for dropping any of these objects:

- A log file group cannot be dropped as long as any tablespaces are using it.
- A tablespace cannot be dropped as long as it contains any data files.
- You cannot drop any data files from a tablespace as long as there remain any tables which are using the tablespace.
- It is not possible to drop files created in association with a different tablespace than the one with which the files were created. (Bug #20053)

For example, to drop all the objects created so far in this section, you would use the following statements:

```
mysql> DROP TABLE dt_1;
mysql> ALTER TABLESPACE ts_1
    -> DROP DATAFILE 'data_2.dat'
    -> ENGINE NDBCLUSTER;
mysql> ALTER TABLESPACE ts_1
    -> DROP DATAFILE 'data_1.dat'
    -> ENGINE NDBCLUSTER;
mysql> DROP TABLESPACE ts_1
    -> ENGINE NDBCLUSTER;
mysql> DROP LOGFILE GROUP lg_1
    -> ENGINE NDBCLUSTER;
```

These statements must be performed in the order shown, except that the two ALTER TABLESPACE ... DROP DATAFILE statements may be executed in either order.

You can obtain information about data files used by Disk Data tables by querying the FILES table in the INFORMATION_SCHEMA database. An extra “NULL row” provides additional information about undo log files. For more information and examples, see The INFORMATION_SCHEMA FILES Table.

It is also possible to view information about allocated and free disk space for each Disk Data table or table partition using the ndb_desc utility. For more information, see Section 6.10, “ndb_desc — Describe NDB Tables”.

7.12.2 Using Symbolic Links with Disk Data Objects

The performance of a MySQL Cluster that uses Disk Data storage can be greatly improved by separating data node file systems from undo log files and tablespace data files and placing these on different disks. In early versions of MySQL Cluster, there was no direct support for this in MySQL

Cluster, but it was possible to achieve this separation using symbolic links as described in this section. MySQL Cluster supports the data node configuration parameters `FileSystemPathDD`, `FileSystemPathDataFiles`, and `FileSystemPathUndoFiles`, which make the use of symbolic links for this purpose unnecessary. For more information about these parameters, see Disk Data file system parameters.

Each data node in the cluster creates a file system in the directory named `ndb_node_id_fs` under the data node's `DataDir` as defined in the `config.ini` file. In this example, we assume that each data node host has 3 disks, aliased as `/data0`, `/data1`, and `/data2`, and that the cluster's `config.ini` includes the following:

```
[ndbd default]
DataDir= /data0
```

Our objective is to place all Disk Data log files in `/data1`, and all Disk Data data files in `/data2`, on each data node host.

> **Note**
>
> In this example, we assume that the cluster's data node hosts are all using Linux operating systems. For other platforms, you may need to substitute you operating system's commands for those shown here.

To accomplish this, perform the following steps:

- Under the data node file system create symbolic links pointing to the other drives:

  ```
  shell> cd /data0/ndb_2_fs
  shell> ls
  D1  D10  D11  D2  D8  D9  LCP
  shell> ln -s /data0 dnlogs
  shell> ln -s /data1 dndata
  ```

  You should now have two symbolic links:

  ```
  shell> ls -l --hide=D*
  lrwxrwxrwx 1 user group   30 2007-03-19 13:58 dndata -> /data1
  lrwxrwxrwx 1 user group   30 2007-03-19 13:59 dnlogs -> /data2
  ```

  We show this only for the data node with node ID 2; however, you must do this for *each* data node.

- Now, in the `mysql` client, create a log file group and tablespace using the symbolic links, as shown here:

  ```
  mysql> CREATE LOGFILE GROUP lg1
      ->    ADD UNDOFILE 'dnlogs/undo1.log'
      ->    INITIAL_SIZE 150M
      ->    UNDO_BUFFER_SIZE = 1M
      ->    ENGINE=NDBCLUSTER;
  mysql> CREATE TABLESPACE ts1
      ->    ADD DATAFILE 'dndata/data1.log'
      ->    USE LOGFILE GROUP lg1
      ->    INITIAL_SIZE 1G
      ->    ENGINE=NDBCLUSTER;
  ```

  Verify that the files were created and placed correctly as shown here:

  ```
  shell> cd /data1
  shell> ls -l
  total 2099304
  -rw-rw-r--  1 user group 157286400 2007-03-19 14:02 undo1.dat
  shell> cd /data2
  ```

```
shell> ls -l
total 2099304
-rw-rw-r--  1 user group 1073741824 2007-03-19 14:02 data1.dat
```

- If you are running multiple data nodes on one host, you must take care to avoid having them try to use the same space for Disk Data files. You can make this easier by creating a symbolic link in each data node file system. Suppose you are using `/data0` for both data node file systems, but you wish to have the Disk Data files for both nodes on `/data1`. In this case, you can do something similar to what is shown here:

```
shell> cd /data0
shell> ln -s /data1/dn2 ndb_2_fs/dd
shell> ln -s /data1/dn3 ndb_3_fs/dd
shell> ls -l --hide=D* ndb_2_fs
lrwxrwxrwx 1 user group   30 2007-03-19 14:22 dd -> /data1/dn2
shell> ls -l --hide=D* ndb_3_fs
lrwxrwxrwx 1 user group   30 2007-03-19 14:22 dd -> /data1/dn3
```

- Now you can create a logfile group and tablespace using the symbolic link, like this:

```
mysql> CREATE LOGFILE GROUP lg1
    ->    ADD UNDOFILE 'dd/undo1.log'
    ->    INITIAL_SIZE 150M
    ->    UNDO_BUFFER_SIZE = 1M
    ->    ENGINE=NDBCLUSTER;
mysql> CREATE TABLESPACE ts1
    ->    ADD DATAFILE 'dd/data1.log'
    ->    USE LOGFILE GROUP lg1
    ->    INITIAL_SIZE 1G
    ->    ENGINE=NDBCLUSTER;
```

  Verify that the files were created and placed correctly as shown here:

```
shell> cd /data1
shell> ls
dn2        dn3
shell> ls dn2
undo1.log        data1.log
shell> ls dn3
undo1.log        data1.log
```

7.12.3 MySQL Cluster Disk Data Storage Requirements

The following items apply to Disk Data storage requirements:

- Variable-length columns of Disk Data tables take up a fixed amount of space. For each row, this is equal to the space required to store the largest possible value for that column.

  For general information about calculating these values, see Data Type Storage Requirements.

  You can obtain an estimate the amount of space available in data files and undo log files by querying the `INFORMATION_SCHEMA.FILES` table. For more information and examples, see The INFORMATION_SCHEMA FILES Table.

  > **Note**
  >
  > The `OPTIMIZE TABLE` statement does not have any effect on Disk Data tables.

- In a Disk Data table, the first 256 bytes of a `TEXT` or `BLOB` column are stored in memory; only the remainder is stored on disk.

- Each row in a Disk Data table uses 8 bytes in memory to point to the data stored on disk. This means that, in some cases, converting an in-memory column to the disk-based format can actually result in greater memory usage. For example, converting a `CHAR(4)` column from memory-based to disk-based format increases the amount of `DataMemory` used per row from 4 to 8 bytes.

> **Important**
>
> Starting the cluster with the `--initial` option does *not* remove Disk Data files. You must remove these manually prior to performing an initial restart of the cluster.

Performance of Disk Data tables can be improved by minimizing the number of disk seeks by making sure that `DiskPageBufferMemory` is of sufficient size. You can query the `diskpagebuffer` table to help determine whether the value for this parameter needs to be increased.

7.13 Adding MySQL Cluster Data Nodes Online

This section describes how to add MySQL Cluster data nodes “online”—that is, without needing to shut down the cluster completely and restart it as part of the process.

> **Important**
>
> Currently, you must add new data nodes to a MySQL Cluster as part of a new node group. In addition, it is not possible to change the number of replicas (or the number of nodes per node group) online.

7.13.1 Adding MySQL Cluster Data Nodes Online: General Issues

This section provides general information about the behavior of and current limitations in adding MySQL Cluster nodes online.

Redistribution of Data. The ability to add new nodes online includes a means to reorganize `NDBCLUSTER` table data and indexes so that they are distributed across all data nodes, including the new ones, by means of the `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement. Table reorganization of both in-memory and Disk Data tables is supported. This redistribution does not currently include unique indexes (only ordered indexes are redistributed). Prior to MySQL Cluster NDB 7.3.3, `BLOB` table data is also not redistributed using this method (Bug #13714148).

The redistribution for `NDBCLUSTER` tables already existing before the new data nodes were added is not automatic, but can be accomplished using simple SQL statements in `mysql` or another MySQL client application. However, all data and indexes added to tables created after a new node group has been added are distributed automatically among all cluster data nodes, including those added as part of the new node group.

Partial starts. It is possible to add a new node group without all of the new data nodes being started. It is also possible to add a new node group to a degraded cluster—that is, a cluster that is only partially started, or where one or more data nodes are not running. In the latter case, the cluster must have enough nodes running to be viable before the new node group can be added.

Effects on ongoing operations. Normal DML operations using MySQL Cluster data are not prevented by the creation or addition of a new node group, or by table reorganization. However, it is not possible to perform DDL concurrently with table reorganization—that is, no other DDL statements can be issued while an `ALTER TABLE ... REORGANIZE PARTITION` statement is executing. In addition, during the execution of `ALTER TABLE ... REORGANIZE PARTITION` (or the execution of any other DDL statement), it is not possible to restart cluster data nodes.

Failure handling. Failures of data nodes during node group creation and table reorganization are handled as hown in the following table:

Failure occurs during:	Failure occurs in:		
	“Old” data nodes	**“New” data nodes**	**System**
Node group creation	• **If a node other than the master fails:**	• **If a node other than the master fails:**	• **If the execution of CREATE**

Failure occurs during:	Failure occurs in:		
	“Old” data nodes	“New” data nodes	System

	The creation of the node group is always rolled forward.
	• **If the master fails:**
	  • **If the internal commit point has been reached:** The creation of the node group is rolled forward.
	  • **If the internal commit point has not yet been reached.** The creation of the node group is rolled back
		The creation of the node group is always rolled forward.
		• **If the master fails:**
		  • **If the internal commit point has been reached:** The creation of the node group is rolled forward.
		  • **If the internal commit point has not yet been reached.** The creation of the node group is rolled back
			NODEGROUP has reached the internal commit point: When restarted, the cluster includes the new node group. Otherwise it without.
			• **If the execution of CREATE NODEGROUP has not yet reached the internal commit point:** When restarted, the cluster does not include the new node group.

Table reorganization	• **If a node other than the master fails:** The table reorganization is always rolled forward.
	• **If the master fails:**
	  • **If the internal commit point has been reached:** The table reorganization is rolled forward.
	  • **If the internal commit point has not yet been reached.** The table reorganization is rolled back.
		• **If a node other than the master fails:** The table reorganization is always rolled forward.
		• **If the master fails:**
		  • **If the internal commit point has been reached:** The table reorganization is rolled forward.
		  • **If the internal commit point has not yet been reached.** The table reorganization is rolled back.
			• **If the execution of an ALTER ONLINE TABLE table REORGANIZE PARTITION statement has reached the internal commit point:** When the cluster is restarted, the data and indexes belonging to *`table`* are distributed using the “new” data nodes.
			• **If the execution of an ALTER ONLINE TABLE table REORGANIZE PARTITION statement has not yet reached the internal commit point:** When the cluster is restarted, the data and indexes belonging to *`table`* are distributed using only the “old” data nodes.

Dropping node groups. The `ndb_mgm` client supports a `DROP NODEGROUP` command, but it is possible to drop a node group only when no data nodes in the node group contain any data. Since there is currently no way to “empty” a specific data node or node group, this command works only the following two cases:

1. After issuing `CREATE NODEGROUP` in the `ndb_mgm` client, but before issuing any `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statements in the `mysql` client.

2. After dropping all `NDBCLUSTER` tables using `DROP TABLE`.

   `TRUNCATE TABLE` does not work for this purpose because the data nodes continue to store the table definitions.

7.13.2 Adding MySQL Cluster Data Nodes Online: Basic procedure

In this section, we list the basic steps required to add new data nodes to a MySQL Cluster. This procedure applies whether you are using `ndbd` or `ndbmtd` binaries for the data node processes. For a more detailed example, see Section 7.13.3, “Adding MySQL Cluster Data Nodes Online: Detailed Example”.

Assuming that you already have a running MySQL Cluster, adding data nodes online requires the following steps:

1. Edit the cluster configuration `config.ini` file, adding new `[ndbd]` sections corresponding to the nodes to be added. In the case where the cluster uses multiple management servers, these changes need to be made to all `config.ini` files used by the management servers.

   You must be careful that node IDs for any new data nodes added in the `config.ini` file do not overlap node IDs used by existing nodes. In the event that you have API nodes using dynamically allocated node IDs and these IDs match node IDs that you want to use for new data nodes, it is possible to force any such API nodes to “migrate”, as described later in this procedure.

2. Perform a rolling restart of all MySQL Cluster management servers.

   > **Important**
   >
   > All management servers must be restarted with the `--reload` or `--initial` option to force the reading of the new configuration.

3. Perform a rolling restart of all existing MySQL Cluster data nodes. It is not necessary (or usually even desirable) to use `--initial` when restarting the existing data nodes.

   If you are using API nodes with dynamically allocated IDs matching any node IDs that you wish to assign to new data nodes, you must restart all API nodes (including SQL nodes) before restarting any of the data nodes processes in this step. This causes any API nodes with node IDs that were previously not explicitly assigned to relinquish those node IDs and acquire new ones.

4. Perform a rolling restart of any SQL or API nodes connected to the MySQL Cluster.

5. Start the new data nodes.

   The new data nodes may be started in any order. They can also be started concurrently, as long as they are started after the rolling restarts of all existing data nodes have been completed, and before proceeding to the next step.

6. Execute one or more `CREATE NODEGROUP` commands in the MySQL Cluster management client to create the new node group or node groups to which the new data nodes will belong.

7. Redistribute the cluster's data among all data nodes (including the new ones) by issuing an `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement in the `mysql` client for each `NDBCLUSTER` table.

   > **Note**
   >
   > This needs to be done only for tables already existing at the time the new node group is added. Data in tables created after the new node group is

added is distributed automatically; however, data added to any given table `tbl` that existed before the new nodes were added is not distributed using the new nodes until that table has been reorganized using `ALTER ONLINE TABLE tbl REORGANIZE PARTITION`.

8. Reclaim the space freed on the “old” nodes by issuing, for each `NDBCLUSTER` table, an `OPTIMIZE TABLE` statement in the `mysql` client.

You can add all the nodes desired, then issue several `CREATE NODEGROUP` commands in succession to add the new node groups to the cluster.

7.13.3 Adding MySQL Cluster Data Nodes Online: Detailed Example

In this section we provide a detailed example illustrating how to add new MySQL Cluster data nodes online, starting with a MySQL Cluster having 2 data nodes in a single node group and concluding with a cluster having 4 data nodes in 2 node groups.

Starting configuration. For purposes of illustration, we assume a minimal configuration, and that the cluster uses a `config.ini` file containing only the following information:

```
[ndbd default]
DataMemory = 100M
IndexMemory = 100M
NoOfReplicas = 2
DataDir = /usr/local/mysql/var/mysql-cluster
[ndbd]
Id = 1
HostName = 192.168.0.1
[ndbd]
Id = 2
HostName = 192.168.0.2
[mgm]
HostName = 192.168.0.10
Id = 10
[api]
Id=20
HostName = 192.168.0.20
[api]
Id=21
HostName = 192.168.0.21
```

Note

We have left a gap in the sequence between data node IDs and other nodes. This make it easier later to assign node IDs that are not already in use to data nodes which are newly added.

We also assume that you have already started the cluster using the appropriate command line or `my.cnf` options, and that running `SHOW` in the management client produces output similar to what is shown here:

```
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.27-ndb-7.4.9, Nodegroup: 0)
[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.27-ndb-7.4.9)
[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.27-ndb-7.4.9)
id=21   @192.168.0.21  (5.6.27-ndb-7.4.9)
```

Finally, we assume that the cluster contains a single NDBCLUSTER table created as shown here:

```
USE n;
CREATE TABLE ips (
    id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    country_code CHAR(2) NOT NULL,
    type CHAR(4) NOT NULL,
    ip_address varchar(15) NOT NULL,
    addresses BIGINT UNSIGNED DEFAULT NULL,
    date BIGINT UNSIGNED DEFAULT NULL
)   ENGINE NDBCLUSTER;
```

The memory usage and related information shown later in this section was generated after inserting approximately 50000 rows into this table.

> **Note**
>
> In this example, we show the single-threaded ndbd being used for the data node processes. However—beginning with MySQL Cluster NDB 7.0.4—you can also apply this example, if you are using the multi-threaded ndbmtd by substituting ndbmtd for ndbd wherever it appears in the steps that follow. (Bug #43108)

Step 1: Update configuration file. Open the cluster global configuration file in a text editor and add [ndbd] sections corresponding to the 2 new data nodes. (We give these data nodes IDs 3 and 4, and assume that they are to be run on host machines at addresses 192.168.0.3 and 192.168.0.4, respectively.) After you have added the new sections, the contents of the config.ini file should look like what is shown here, where the additions to the file are shown in bold type:

```
[ndbd default]
DataMemory = 100M
IndexMemory = 100M
NoOfReplicas = 2
DataDir = /usr/local/mysql/var/mysql-cluster
[ndbd]
Id = 1
HostName = 192.168.0.1
[ndbd]
Id = 2
HostName = 192.168.0.2
[ndbd]
Id = 3
HostName = 192.168.0.3
[ndbd]
Id = 4
HostName = 192.168.0.4
[mgm]
HostName = 192.168.0.10
Id = 10
[api]
Id=20
HostName = 192.168.0.20
[api]
Id=21
HostName = 192.168.0.21
```

Once you have made the necessary changes, save the file.

Step 2: Restart the management server. Restarting the cluster management server requires that you issue separate commands to stop the management server and then to start it again, as follows:

1. Stop the management server using the management client STOP command, as shown here:

```
ndb_mgm> 10 STOP
Node 10 has shut down.
```

```
Disconnecting to allow Management Server to shutdown
shell>
```

2. Because shutting down the management server causes the management client to terminate, you must start the management server from the system shell. For simplicity, we assume that `config.ini` is in the same directory as the management server binary, but in practice, you must supply the correct path to the configuration file. You must also supply the `--reload` or `--initial` option so that the management server reads the new configuration from the file rather than its configuration cache. If your shell's current directory is also the same as the directory where the management server binary is located, then you can invoke the management server as shown here:

```
shell> ndb_mgmd -f config.ini --reload
2008-12-08 17:29:23 [MgmSrvr] INFO     -- NDB Cluster Management Server. 5.6.27-ndb-7.4.9
2008-12-08 17:29:23 [MgmSrvr] INFO     -- Reading cluster configuration from 'config.ini'
```

If you check the output of `SHOW` in the management client after restarting the `ndb_mgm` process, you should now see something like this:

```
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.27-ndb-7.4.9, Nodegroup: 0)
id=3 (not connected, accepting connect from 192.168.0.3)
id=4 (not connected, accepting connect from 192.168.0.4)
[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.27-ndb-7.4.9)
[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.27-ndb-7.4.9)
id=21   @192.168.0.21  (5.6.27-ndb-7.4.9)
```

Step 3: Perform a rolling restart of the existing data nodes. This step can be accomplished entirely within the cluster management client using the `RESTART` command, as shown here:

```
ndb_mgm> 1 RESTART
Node 1: Node shutdown initiated
Node 1: Node shutdown completed, restarting, no start.
Node 1 is being restarted
ndb_mgm> Node 1: Start initiated (version 7.4.9)
Node 1: Started (version 7.1.36)
ndb_mgm> 2 RESTART
Node 2: Node shutdown initiated
Node 2: Node shutdown completed, restarting, no start.
Node 2 is being restarted
ndb_mgm> Node 2: Start initiated (version 7.4.9)
ndb_mgm> Node 2: Started (version 7.4.9)
```

Important

After issuing each `X RESTART` command, wait until the management client reports `Node X: Started (version ...)` *before* proceeding any further.

You can verify that all existing data nodes were restarted using the updated configuration by checking the `ndbinfo.nodes` table in the `mysql` client.

Step 4: Perform a rolling restart of all cluster API nodes. Shut down and restart each MySQL server acting as an SQL node in the cluster using `mysqladmin shutdown` followed by `mysqld_safe` (or another startup script). This should be similar to what is shown here, where *`password`* is the MySQL `root` password for a given MySQL server instance:

```
shell> mysqladmin -uroot -ppassword shutdown
```

```
081208 20:19:56 mysqld_safe mysqld from pid file
/usr/local/mysql/var/tonfisk.pid ended
shell> mysqld_safe --ndbcluster --ndb-connectstring=192.168.0.10 &
081208 20:20:06 mysqld_safe Logging to '/usr/local/mysql/var/tonfisk.err'.
081208 20:20:06 mysqld_safe Starting mysqld daemon with databases
from /usr/local/mysql/var
```

Of course, the exact input and output depend on how and where MySQL is installed on the system, as well as which options you choose to start it (and whether or not some or all of these options are specified in a `my.cnf` file).

Step 5: Perform an initial start of the new data nodes. From a system shell on each of the hosts for the new data nodes, start the data nodes as shown here, using the `--initial` option:

```
shell> ndbd -c 192.168.0.10 --initial
```

> **Note**
>
> Unlike the case with restarting the existing data nodes, you can start the new data nodes concurrently; you do not need to wait for one to finish starting before starting the other.

Wait until both of the new data nodes have started before proceeding with the next step. Once the new data nodes have started, you can see in the output of the management client `SHOW` command that they do not yet belong to any node group (as indicated with bold type here):

```
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.27-ndb-7.4.9, Nodegroup: 0)
id=3    @192.168.0.3  (5.6.27-ndb-7.4.9, no nodegroup)
id=4    @192.168.0.4  (5.6.27-ndb-7.4.9, no nodegroup)
[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.27-ndb-7.4.9)
[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.27-ndb-7.4.9)
id=21   @192.168.0.21  (5.6.27-ndb-7.4.9)
```

Step 6: Create a new node group. You can do this by issuing a `CREATE NODEGROUP` command in the cluster management client. This command takes as its argument a comma-separated list of the node IDs of the data nodes to be included in the new node group, as shown here:

```
ndb_mgm> CREATE NODEGROUP 3,4
Nodegroup 1 created
```

By issuing `SHOW` again, you can verify that data nodes 3 and 4 have joined the new node group (again indicated in bold type):

```
ndb_mgm> SHOW
Connected to Management Server at: 192.168.0.10:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=1    @192.168.0.1  (5.6.27-ndb-7.4.9, Nodegroup: 0, *)
id=2    @192.168.0.2  (5.6.27-ndb-7.4.9, Nodegroup: 0)
id=3    @192.168.0.3  (5.6.27-ndb-7.4.9, Nodegroup: 1)
id=4    @192.168.0.4  (5.6.27-ndb-7.4.9, Nodegroup: 1)
[ndb_mgmd(MGM)] 1 node(s)
id=10   @192.168.0.10  (5.6.27-ndb-7.4.9)
[mysqld(API)]   2 node(s)
id=20   @192.168.0.20  (5.6.27-ndb-7.4.9)
id=21   @192.168.0.21  (5.6.27-ndb-7.4.9)
```

Step 7: Redistribute cluster data. When a node group is created, existing data and indexes are not automatically distributed to the new node group's data nodes, as you can see by issuing the appropriate `REPORT` command in the management client:

```
ndb_mgm> ALL REPORT MEMORY
Node 1: Data usage is 5%(177 32K pages of total 3200)
Node 1: Index usage is 0%(108 8K pages of total 12832)
Node 2: Data usage is 5%(177 32K pages of total 3200)
Node 2: Index usage is 0%(108 8K pages of total 12832)
Node 3: Data usage is 0%(0 32K pages of total 3200)
Node 3: Index usage is 0%(0 8K pages of total 12832)
Node 4: Data usage is 0%(0 32K pages of total 3200)
Node 4: Index usage is 0%(0 8K pages of total 12832)
```

By using `ndb_desc` with the `-p` option, which causes the output to include partitioning information, you can see that the table still uses only 2 partitions (in the `Per partition info` section of the output, shown here in bold text):

```
shell> ndb_desc -c 192.168.0.10 -d n ips -p
-- ips --
Version: 1
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 6
Number of primary keys: 1
Length of frm data: 340
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 2
TableStatus: Retrieved
-- Attributes --
id Bigint PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
country_code Char(2;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
type Char(4;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
ip_address Varchar(15;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
addresses Bigunsigned NULL AT=FIXED ST=MEMORY
date Bigunsigned NULL AT=FIXED ST=MEMORY
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
-- Per partition info --
Partition    Row count   Commit count  Frag fixed memory   Frag varsized memory
0            26086       26086         1572864             557056
1            26329       26329         1605632             557056
NDBT_ProgramExit: 0 - OK
```

You can cause the data to be redistributed among all of the data nodes by performing, for each `NDBCLUSTER` table, an `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement in the `mysql` client. After issuing the statement `ALTER ONLINE TABLE ips REORGANIZE PARTITION`, you can see using `ndb_desc` that the data for this table is now stored using 4 partitions, as shown here (with the relevant portions of the output in bold type):

```
shell> ndb_desc -c 192.168.0.10 -d n ips -p
-- ips --
Version: 16777217
Fragment type: 9
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 6
Number of primary keys: 1
```

```
Length of frm data: 341
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
FragmentCount: 4
TableStatus: Retrieved
-- Attributes --
id Bigint PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
country_code Char(2;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
type Char(4;latin1_swedish_ci) NOT NULL AT=FIXED ST=MEMORY
ip_address Varchar(15;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
addresses Bigunsigned NULL AT=FIXED ST=MEMORY
date Bigunsigned NULL AT=FIXED ST=MEMORY
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
-- Per partition info --
Partition   Row count   Commit count  Frag fixed memory   Frag varsized memory
0           12981       52296         1572864             557056
1           13236       52515         1605632             557056
2           13105       13105         819200              294912
3           13093       13093         819200              294912
NDBT_ProgramExit: 0 - OK
```

> **Note**
>
> Normally, `ALTER [ONLINE] TABLE table_name REORGANIZE PARTITION` is used with a list of partition identifiers and a set of partition definitions to create a new partitioning scheme for a table that has already been explicitly partitioned. Its use here to redistribute data onto a new MySQL Cluster node group is an exception in this regard; when used in this way, only the name of the table is used following the `TABLE` keyword, and no other keywords or identifiers follow `REORGANIZE PARTITION`.
>
> For more information, see ALTER TABLE Syntax.

In addition, for each table, the `ALTER ONLINE TABLE` statement should be followed by an `OPTIMIZE TABLE` to reclaim wasted space. You can obtain a list of all `NDBCLUSTER` tables using the following query against the `INFORMATION_SCHEMA.TABLES` table:

```
SELECT TABLE_SCHEMA, TABLE_NAME
    FROM INFORMATION_SCHEMA.TABLES
    WHERE ENGINE = 'NDBCLUSTER';
```

> **Note**
>
> The `INFORMATION_SCHEMA.TABLES.ENGINE` value for a MySQL Cluster table is always `NDBCLUSTER`, regardless of whether the `CREATE TABLE` statement used to create the table (or `ALTER TABLE` statement used to convert an existing table from a different storage engine) used `NDB` or `NDBCLUSTER` in its `ENGINE` option.

You can see after performing these statements in the output of `ALL REPORT MEMORY` that the data and indexes are now redistributed between all cluster data nodes, as shown here:

```
ndb_mgm> ALL REPORT MEMORY
Node 1: Data usage is 5%(176 32K pages of total 3200)
Node 1: Index usage is 0%(76 8K pages of total 12832)
Node 2: Data usage is 5%(176 32K pages of total 3200)
Node 2: Index usage is 0%(76 8K pages of total 12832)
Node 3: Data usage is 2%(80 32K pages of total 3200)
Node 3: Index usage is 0%(51 8K pages of total 12832)
Node 4: Data usage is 2%(80 32K pages of total 3200)
Node 4: Index usage is 0%(50 8K pages of total 12832)
```

Note

Since only one DDL operation on `NDBCLUSTER` tables can be executed at a time, you must wait for each `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement to finish before issuing the next one.

It is not necessary to issue `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statements for `NDBCLUSTER` tables created *after* the new data nodes have been added; data added to such tables is distributed among all data nodes automatically. However, in `NDBCLUSTER` tables that existed *prior to* the addition of the new nodes, neither existing nor new data is distributed using the new nodes until these tables have been reorganized using `ALTER ONLINE TABLE ... REORGANIZE PARTITION`.

Alternative procedure, without rolling restart. It is possible to avoid the need for a rolling restart by configuring the extra data nodes, but not starting them, when first starting the cluster. We assume, as before, that you wish to start with two data nodes—nodes 1 and 2—in one node group and later to expand the cluster to four data nodes, by adding a second node group consisting of nodes 3 and 4:

```
[ndbd default]
DataMemory = 100M
IndexMemory = 100M
NoOfReplicas = 2
DataDir = /usr/local/mysql/var/mysql-cluster
[ndbd]
Id = 1
HostName = 192.168.0.1
[ndbd]
Id = 2
HostName = 192.168.0.2
[ndbd]
Id = 3
HostName = 192.168.0.3
Nodegroup = 65536
[ndbd]
Id = 4
HostName = 192.168.0.4
Nodegroup = 65536
[mgm]
HostName = 192.168.0.10
Id = 10
[api]
Id=20
HostName = 192.168.0.20
[api]
Id=21
HostName = 192.168.0.21
```

The data nodes to be brought online at a later time (nodes 3 and 4) can be configured with `NodeGroup = 65536`, in which case nodes 1 and 2 can each be started as shown here:

```
shell> ndbd -c 192.168.0.10 --initial
```

The data nodes configured with `NodeGroup = 65536` are treated by the management server as though you had started nodes 1 and 2 using `--nowait-nodes=3,4` after waiting for a period of time determined by the setting for the `StartNoNodeGroupTimeout` data node configuration parameter. By default, this is 15 seconds (15000 milliseconds).

Note

`StartNoNodegroupTimeout` must be the same for all data nodes in the cluster; for this reason, you should always set it in the `[ndbd default]` section of the `config.ini` file, rather than for individual data nodes.

When you are ready to add the second node group, you need only perform the following additional steps:

1. Start data nodes 3 and 4, invoking the data node process once for each new node:

```
shell> ndbd -c 192.168.0.10 --initial
```

2. Issue the appropriate CREATE NODEGROUP command in the management client:

```
ndb_mgm> CREATE NODEGROUP 3,4
```

3. In the mysql client, issue ALTER ONLINE TABLE ... REORGANIZE PARTITION and OPTIMIZE TABLE statements for each existing NDBCLUSTER table. (As noted elsewhere in this section, existing MySQL Cluster tables cannot use the new nodes for data distribution until this has been done.)

7.14 Distributed MySQL Privileges for MySQL Cluster

MySQL Cluster supports distribution of MySQL users and privileges across all SQL nodes in a MySQL Cluster. This support is not enabled by default; you should follow the procedure outlined in this section in order to do so.

Normally, each MySQL server's user privilege tables in the mysql database must use the MyISAM storage engine, which means that a user account and its associated privileges created on one SQL node are not available on the cluster's other SQL nodes. An SQL file ndb_dist_priv.sql is provided with MySQL Cluster NDB 7.3 and later distributions. This file can be found in the share directory in the MySQL installation directory.

The first step in enabling distributed privileges is to load this script into a MySQL Server that functions as an SQL node (which we refer to after this as the *target* SQL node or MySQL Server). You can do this by executing the following command from the system shell on the target SQL node after changing to its MySQL installation directory (where *options* stands for any additional options needed to connect to this SQL node):

```
shell> mysql options -uroot < share/ndb_dist_priv.sql
```

Importing ndb_dist_priv.sql creates a number of stored routines (six stored procedures and one stored function) in the mysql database on the target SQL node. After connecting to the SQL node in the mysql client (as the MySQL root user), you can verify that these were created as shown here:

```
mysql> SELECT ROUTINE_NAME, ROUTINE_SCHEMA, ROUTINE_TYPE
    ->     FROM INFORMATION_SCHEMA.ROUTINES
    ->     WHERE ROUTINE_NAME LIKE 'mysql_cluster%'
    ->     ORDER BY ROUTINE_TYPE;
+---------------------------------------------+----------------+--------------+
| ROUTINE_NAME                                | ROUTINE_SCHEMA | ROUTINE_TYPE |
+---------------------------------------------+----------------+--------------+
| mysql_cluster_privileges_are_distributed    | mysql          | FUNCTION     |
| mysql_cluster_backup_privileges             | mysql          | PROCEDURE    |
| mysql_cluster_move_grant_tables             | mysql          | PROCEDURE    |
| mysql_cluster_move_privileges               | mysql          | PROCEDURE    |
| mysql_cluster_restore_local_privileges      | mysql          | PROCEDURE    |
| mysql_cluster_restore_privileges            | mysql          | PROCEDURE    |
| mysql_cluster_restore_privileges_from_local | mysql          | PROCEDURE    |
+---------------------------------------------+----------------+--------------+
7 rows in set (0.01 sec)
```

The stored procedure named mysql_cluster_move_privileges creates backup copies of the existing privilege tables, then converts them to NDB.

mysql_cluster_move_privileges performs the backup and conversion in two steps. The first step is to call mysql_cluster_backup_privileges, which creates two sets of copies in the mysql database:

- A set of local copies that use the MyISAM storage engine. Their names are generated by adding the suffix _backup to the original privilege table names.
- A set of distributed copies that use the NDBCLUSTER storage engine. These tables are named by prefixing ndb_ and appending _backup to the names of the original tables.

After the copies are created, mysql_cluster_move_privileges invokes mysql_cluster_move_grant_tables, which contains the ALTER TABLE ... ENGINE = NDB statements that convert the mysql system tables to NDB.

Normally, you should not invoke either mysql_cluster_backup_privileges or mysql_cluster_move_grant_tables manually; these stored procedures are intended only for use by mysql_cluster_move_privileges.

Although the original privilege tables are backed up automatically, it is always a good idea to create backups manually of the existing privilege tables on all affected SQL nodes before proceeding. You can do this using mysqldump in a manner similar to what is shown here:

```
shell> mysqldump options -uroot \
    mysql user db tables_priv columns_priv procs_priv proxies_priv > backup_file
```

To perform the conversion, you must be connected to the target SQL node using the mysql client (again, as the MySQL root user). Invoke the stored procedure like this:

```
mysql> CALL mysql.mysql_cluster_move_privileges();
Query OK, 0 rows affected (22.32 sec)
```

Depending on the number of rows in the privilege tables, this procedure may take some time to execute. If some of the privilege tables are empty, you may see one or more No data - zero rows fetched, selected, or processed warnings when mysql_cluster_move_privileges returns. In such cases, the warnings may be safely ignored. To verify that the conversion was successful, you can use the stored function mysql_cluster_privileges_are_distributed as shown here:

```
mysql> SELECT CONCAT(
    ->    'Conversion ',
    ->    IF(mysql.mysql_cluster_privileges_are_distributed(), 'succeeded', 'failed'),
    ->    '.')
    ->    AS Result;
+-----------------------+
| Result                |
+-----------------------+
| Conversion succeeded. |
+-----------------------+
1 row in set (0.00 sec)
```

mysql_cluster_privileges_are_distributed checks for the existence of the distributed privilege tables and returns 1 if all of the privilege tables are distributed; otherwise, it returns 0.

You can verify that the backups have been created using a query such as this one:

```
mysql> SELECT TABLE_NAME, ENGINE FROM INFORMATION_SCHEMA.TABLES
    ->     WHERE TABLE_SCHEMA = 'mysql' AND TABLE_NAME LIKE '%backup'
    ->     ORDER BY ENGINE;
+-------------------------+------------+
| TABLE_NAME              | ENGINE     |
+-------------------------+------------+
| host_backup             | MyISAM     |
| db_backup               | MyISAM     |
| columns_priv_backup     | MyISAM     |
| user_backup             | MyISAM     |
```

```
| tables_priv_backup      | MyISAM     |
| proxies_priv_backup     | MyISAM     |
| procs_priv_backup       | MyISAM     |
| ndb_user_backup         | ndbcluster |
| ndb_tables_priv_backup  | ndbcluster |
| ndb_proxies_priv_backup | ndbcluster |
| ndb_procs_priv_backup   | ndbcluster |
| ndb_host_backup         | ndbcluster |
| ndb_db_backup           | ndbcluster |
| ndb_columns_priv_backup | ndbcluster |
+-------------------------+------------+
14 rows in set (0.00 sec)
```

Once the conversion to distributed privileges has been made, any time a MySQL user account is created, dropped, or has its privileges updated on any SQL node, the changes take effect immediately on all other MySQL servers attached to the cluster. Once privileges are distributed, any new MySQL Servers that connect to the cluster automatically participate in the distribution.

> **Note**
>
> For clients connected to SQL nodes at the time that `mysql_cluster_move_privileges` is executed, you may need to execute `FLUSH PRIVILEGES` on those SQL nodes, or to disconnect and then reconnect the clients, in order for those clients to be able to see the changes in privileges.

All MySQL user privileges are distributed across all connected MySQL Servers. This includes any privileges associated with views and stored routines, even though distribution of views and stored routines themselves is not currently supported.

In the event that an SQL node becomes disconnected from the cluster while `mysql_cluster_move_privileges` is running, you must drop its privilege tables after reconnecting to the cluster, using a statement such as `DROP TABLE IF EXISTS mysql.user mysql.db mysql.tables_priv mysql.columns_priv mysql.procs_priv`. This causes the SQL node to use the shared privilege tables rather than its own local versions of them. This is not needed when connecting a new SQL node to the cluster for the first time.

In the event of an initial restart of the entire cluster (all data nodes shut down, then started again with `--initial`), the shared privilege tables are lost. If this happens, you can restore them using the original target SQL node either from the backups made by `mysql_cluster_move_privileges` or from a dump file created with `mysqldump`. If you need to use a new MySQL Server to perform the restoration, you should start it with `--skip-grant-tables` when connecting to the cluster for the first time; after this, you can restore the privilege tables locally, then distribute them again using `mysql_cluster_move_privileges`. After restoring and distributing the tables, you should restart this MySQL Server without the `--skip-grant-tables` option.

You can also restore the distributed tables using `ndb_restore --restore-privilege-tables` [308] from a backup made using `START BACKUP` in the `ndb_mgm` client. (The `MyISAM` tables created by `mysql_cluster_move_privileges` are not backed up by the `START BACKUP` command.) `ndb_restore` does not restore the privilege tables by default; the `--restore-privilege-tables` [308] option causes it to do so.

You can restore the SQL node's local privileges using either of two procedures. `mysql_cluster_restore_privileges` works as follows:

1. If copies of the `mysql.ndb_*_backup` tables are available, attempt to restore the system tables from these.

2. Otherwise, attempt to restore the system tables from the local backups named `*_backup` (without the `ndb_` prefix).

The other procedure, named `mysql_cluster_restore_local_privileges`, restores the system tables from the local backups only, without checking the `ndb_*` backups.

The system tables re-created by `mysql_cluster_restore_privileges` or `mysql_cluster_restore_local_privileges` use the MySQL server default storage engine; they are not shared or distributed in any way, and do not use MySQL Cluster's `NDB` storage engine.

The additional stored procedure `mysql_cluster_restore_privileges_from_local` is intended for the use of `mysql_cluster_restore_privileges` and `mysql_cluster_restore_local_privileges`. It should not be invoked directly.

> **Important**
>
> Applications that access MySQL Cluster data directly, including NDB API and ClusterJ applications, are not subject to the MySQL privilege system. This means that, once you have distributed the grant tables, they can be freely accessed by such applications, just as they can any other `NDB` tables. In particular, you should keep in mind that *NDB API and ClusterJ applications can read and write user names, host names, password hashes, and any other contents of the distributed grant tables without any restrictions*.

7.15 NDB API Statistics Counters and Variables

A number of types of statistical counters relating to actions performed by or affecting `Ndb` objects are available. Such actions include starting and closing (or aborting) transactions; primary key and unique key operations; table, range, and pruned scans; threads blocked while waiting for the completion of various operations; and data and events sent and received by `NDBCLUSTER`. The counters are incremented inside the NDB kernel whenever NDB API calls are made or data is sent to or received by the data nodes. `mysqld` exposes these counters as system status variables; their values can be read in the output of `SHOW STATUS`, or by querying the `INFORMATION_SCHEMA.SESSION_STATUS` or `INFORMATION_SCHEMA.GLOBAL_STATUS` table. By comparing the values before and after statements operating on `NDB` tables, you can observe the corresponding actions taken on the API level, and thus the cost of performing the statement.

You can list all of these status variables using the following `SHOW STATUS` statement:

```
mysql> SHOW STATUS LIKE 'ndb_api%';
+--------------------------------------------+----------+
| Variable_name                              | Value    |
+--------------------------------------------+----------+
| Ndb_api_wait_exec_complete_count_session   | 0        |
| Ndb_api_wait_scan_result_count_session     | 0        |
| Ndb_api_wait_meta_request_count_session    | 0        |
| Ndb_api_wait_nanos_count_session           | 0        |
| Ndb_api_bytes_sent_count_session           | 0        |
| Ndb_api_bytes_received_count_session       | 0        |
| Ndb_api_trans_start_count_session          | 0        |
| Ndb_api_trans_commit_count_session         | 0        |
| Ndb_api_trans_abort_count_session          | 0        |
| Ndb_api_trans_close_count_session          | 0        |
| Ndb_api_pk_op_count_session                | 0        |
| Ndb_api_uk_op_count_session                | 0        |
| Ndb_api_table_scan_count_session           | 0        |
| Ndb_api_range_scan_count_session           | 0        |
| Ndb_api_pruned_scan_count_session          | 0        |
| Ndb_api_scan_batch_count_session           | 0        |
| Ndb_api_read_row_count_session             | 0        |
| Ndb_api_trans_local_read_row_count_session | 0        |
| Ndb_api_event_data_count_injector          | 0        |
| Ndb_api_event_nondata_count_injector       | 0        |
| Ndb_api_event_bytes_count_injector         | 0        |
| Ndb_api_wait_exec_complete_count_slave     | 0        |
| Ndb_api_wait_scan_result_count_slave       | 0        |
| Ndb_api_wait_meta_request_count_slave      | 0        |
| Ndb_api_wait_nanos_count_slave             | 0        |
| Ndb_api_bytes_sent_count_slave             | 0        |
| Ndb_api_bytes_received_count_slave         | 0        |
| Ndb_api_trans_start_count_slave            | 0        |
```

```
| Ndb_api_trans_commit_count_slave            | 0        |
| Ndb_api_trans_abort_count_slave             | 0        |
| Ndb_api_trans_close_count_slave             | 0        |
| Ndb_api_pk_op_count_slave                   | 0        |
| Ndb_api_uk_op_count_slave                   | 0        |
| Ndb_api_table_scan_count_slave              | 0        |
| Ndb_api_range_scan_count_slave              | 0        |
| Ndb_api_pruned_scan_count_slave             | 0        |
| Ndb_api_scan_batch_count_slave              | 0        |
| Ndb_api_read_row_count_slave                | 0        |
| Ndb_api_trans_local_read_row_count_slave    | 0        |
| Ndb_api_wait_exec_complete_count            | 2        |
| Ndb_api_wait_scan_result_count              | 3        |
| Ndb_api_wait_meta_request_count             | 27       |
| Ndb_api_wait_nanos_count                    | 45612023 |
| Ndb_api_bytes_sent_count                    | 992      |
| Ndb_api_bytes_received_count                | 9640     |
| Ndb_api_trans_start_count                   | 2        |
| Ndb_api_trans_commit_count                  | 1        |
| Ndb_api_trans_abort_count                   | 0        |
| Ndb_api_trans_close_count                   | 2        |
| Ndb_api_pk_op_count                         | 1        |
| Ndb_api_uk_op_count                         | 0        |
| Ndb_api_table_scan_count                    | 1        |
| Ndb_api_range_scan_count                    | 0        |
| Ndb_api_pruned_scan_count                   | 0        |
| Ndb_api_scan_batch_count                    | 0        |
| Ndb_api_read_row_count                      | 1        |
| Ndb_api_trans_local_read_row_count          | 1        |
| Ndb_api_event_data_count                    | 0        |
| Ndb_api_event_nondata_count                 | 0        |
| Ndb_api_event_bytes_count                   | 0        |
+---------------------------------------------+----------+
60 rows in set (0.02 sec)
```

These status variables are also available from the `SESSION_STATUS` and `GLOBAL_STATUS` tables of the `INFORMATION_SCHEMA` database, as shown here:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS
    ->   WHERE VARIABLE_NAME LIKE 'ndb_api%';
+---------------------------------------------+----------------+
| VARIABLE_NAME                               | VARIABLE_VALUE |
+---------------------------------------------+----------------+
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SESSION    | 2              |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SESSION      | 0              |
| NDB_API_WAIT_META_REQUEST_COUNT_SESSION     | 1              |
| NDB_API_WAIT_NANOS_COUNT_SESSION            | 8144375        |
| NDB_API_BYTES_SENT_COUNT_SESSION            | 68             |
| NDB_API_BYTES_RECEIVED_COUNT_SESSION        | 84             |
| NDB_API_TRANS_START_COUNT_SESSION           | 1              |
| NDB_API_TRANS_COMMIT_COUNT_SESSION          | 1              |
| NDB_API_TRANS_ABORT_COUNT_SESSION           | 0              |
| NDB_API_TRANS_CLOSE_COUNT_SESSION           | 1              |
| NDB_API_PK_OP_COUNT_SESSION                 | 1              |
| NDB_API_UK_OP_COUNT_SESSION                 | 0              |
| NDB_API_TABLE_SCAN_COUNT_SESSION            | 0              |
| NDB_API_RANGE_SCAN_COUNT_SESSION            | 0              |
| NDB_API_PRUNED_SCAN_COUNT_SESSION           | 0              |
| NDB_API_SCAN_BATCH_COUNT_SESSION            | 0              |
| NDB_API_READ_ROW_COUNT_SESSION              | 1              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SESSION  | 1              |
| NDB_API_EVENT_DATA_COUNT_INJECTOR           | 0              |
| NDB_API_EVENT_NONDATA_COUNT_INJECTOR        | 0              |
| NDB_API_EVENT_BYTES_COUNT_INJECTOR          | 0              |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SLAVE      | 0              |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SLAVE        | 0              |
| NDB_API_WAIT_META_REQUEST_COUNT_SLAVE       | 0              |
| NDB_API_WAIT_NANOS_COUNT_SLAVE              | 0              |
| NDB_API_BYTES_SENT_COUNT_SLAVE              | 0              |
| NDB_API_BYTES_RECEIVED_COUNT_SLAVE          | 0              |
| NDB_API_TRANS_START_COUNT_SLAVE             | 0              |
```

```
| NDB_API_TRANS_COMMIT_COUNT_SLAVE             | 0              |
| NDB_API_TRANS_ABORT_COUNT_SLAVE              | 0              |
| NDB_API_TRANS_CLOSE_COUNT_SLAVE              | 0              |
| NDB_API_PK_OP_COUNT_SLAVE                    | 0              |
| NDB_API_UK_OP_COUNT_SLAVE                    | 0              |
| NDB_API_TABLE_SCAN_COUNT_SLAVE               | 0              |
| NDB_API_RANGE_SCAN_COUNT_SLAVE               | 0              |
| NDB_API_PRUNED_SCAN_COUNT_SLAVE              | 0              |
| NDB_API_SCAN_BATCH_COUNT_SLAVE               | 0              |
| NDB_API_READ_ROW_COUNT_SLAVE                 | 0              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SLAVE     | 0              |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT             | 4              |
| NDB_API_WAIT_SCAN_RESULT_COUNT               | 3              |
| NDB_API_WAIT_META_REQUEST_COUNT              | 28             |
| NDB_API_WAIT_NANOS_COUNT                     | 53756398       |
| NDB_API_BYTES_SENT_COUNT                     | 1060           |
| NDB_API_BYTES_RECEIVED_COUNT                 | 9724           |
| NDB_API_TRANS_START_COUNT                    | 3              |
| NDB_API_TRANS_COMMIT_COUNT                   | 2              |
| NDB_API_TRANS_ABORT_COUNT                    | 0              |
| NDB_API_TRANS_CLOSE_COUNT                    | 3              |
| NDB_API_PK_OP_COUNT                          | 2              |
| NDB_API_UK_OP_COUNT                          | 0              |
| NDB_API_TABLE_SCAN_COUNT                     | 1              |
| NDB_API_RANGE_SCAN_COUNT                     | 0              |
| NDB_API_PRUNED_SCAN_COUNT                    | 0              |
| NDB_API_SCAN_BATCH_COUNT                     | 0              |
| NDB_API_READ_ROW_COUNT                       | 2              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT           | 2              |
| NDB_API_EVENT_DATA_COUNT                     | 0              |
| NDB_API_EVENT_NONDATA_COUNT                  | 0              |
| NDB_API_EVENT_BYTES_COUNT                    | 0              |
+----------------------------------------------+----------------+
60 rows in set (0.00 sec)
mysql> SELECT * FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    ->     WHERE VARIABLE_NAME LIKE 'ndb_api%';
+----------------------------------------------+----------------+
| VARIABLE_NAME                                | VARIABLE_VALUE |
+----------------------------------------------+----------------+
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SESSION     | 2              |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SESSION       | 0              |
| NDB_API_WAIT_META_REQUEST_COUNT_SESSION      | 1              |
| NDB_API_WAIT_NANOS_COUNT_SESSION             | 8144375        |
| NDB_API_BYTES_SENT_COUNT_SESSION             | 68             |
| NDB_API_BYTES_RECEIVED_COUNT_SESSION         | 84             |
| NDB_API_TRANS_START_COUNT_SESSION            | 1              |
| NDB_API_TRANS_COMMIT_COUNT_SESSION           | 1              |
| NDB_API_TRANS_ABORT_COUNT_SESSION            | 0              |
| NDB_API_TRANS_CLOSE_COUNT_SESSION            | 1              |
| NDB_API_PK_OP_COUNT_SESSION                  | 1              |
| NDB_API_UK_OP_COUNT_SESSION                  | 0              |
| NDB_API_TABLE_SCAN_COUNT_SESSION             | 0              |
| NDB_API_RANGE_SCAN_COUNT_SESSION             | 0              |
| NDB_API_PRUNED_SCAN_COUNT_SESSION            | 0              |
| NDB_API_SCAN_BATCH_COUNT_SESSION             | 0              |
| NDB_API_READ_ROW_COUNT_SESSION               | 1              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SESSION   | 1              |
| NDB_API_EVENT_DATA_COUNT_INJECTOR            | 0              |
| NDB_API_EVENT_NONDATA_COUNT_INJECTOR         | 0              |
| NDB_API_EVENT_BYTES_COUNT_INJECTOR           | 0              |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT_SLAVE       | 0              |
| NDB_API_WAIT_SCAN_RESULT_COUNT_SLAVE         | 0              |
| NDB_API_WAIT_META_REQUEST_COUNT_SLAVE        | 0              |
| NDB_API_WAIT_NANOS_COUNT_SLAVE               | 0              |
| NDB_API_BYTES_SENT_COUNT_SLAVE               | 0              |
| NDB_API_BYTES_RECEIVED_COUNT_SLAVE           | 0              |
| NDB_API_TRANS_START_COUNT_SLAVE              | 0              |
| NDB_API_TRANS_COMMIT_COUNT_SLAVE             | 0              |
| NDB_API_TRANS_ABORT_COUNT_SLAVE              | 0              |
| NDB_API_TRANS_CLOSE_COUNT_SLAVE              | 0              |
| NDB_API_PK_OP_COUNT_SLAVE                    | 0              |
| NDB_API_UK_OP_COUNT_SLAVE                    | 0              |
```

```
| NDB_API_TABLE_SCAN_COUNT_SLAVE             | 0              |
| NDB_API_RANGE_SCAN_COUNT_SLAVE             | 0              |
| NDB_API_PRUNED_SCAN_COUNT_SLAVE            | 0              |
| NDB_API_SCAN_BATCH_COUNT_SLAVE             | 0              |
| NDB_API_READ_ROW_COUNT_SLAVE               | 0              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT_SLAVE   | 0              |
| NDB_API_WAIT_EXEC_COMPLETE_COUNT           | 4              |
| NDB_API_WAIT_SCAN_RESULT_COUNT             | 3              |
| NDB_API_WAIT_META_REQUEST_COUNT            | 28             |
| NDB_API_WAIT_NANOS_COUNT                   | 53756398       |
| NDB_API_BYTES_SENT_COUNT                   | 1060           |
| NDB_API_BYTES_RECEIVED_COUNT               | 9724           |
| NDB_API_TRANS_START_COUNT                  | 3              |
| NDB_API_TRANS_COMMIT_COUNT                 | 2              |
| NDB_API_TRANS_ABORT_COUNT                  | 0              |
| NDB_API_TRANS_CLOSE_COUNT                  | 3              |
| NDB_API_PK_OP_COUNT                        | 2              |
| NDB_API_UK_OP_COUNT                        | 0              |
| NDB_API_TABLE_SCAN_COUNT                   | 1              |
| NDB_API_RANGE_SCAN_COUNT                   | 0              |
| NDB_API_PRUNED_SCAN_COUNT                  | 0              |
| NDB_API_SCAN_BATCH_COUNT                   | 0              |
| NDB_API_READ_ROW_COUNT                     | 2              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT         | 2              |
| NDB_API_EVENT_DATA_COUNT                   | 0              |
| NDB_API_EVENT_NONDATA_COUNT                | 0              |
| NDB_API_EVENT_BYTES_COUNT                  | 0              |
+--------------------------------------------+----------------+
60 rows in set (0.00 sec)
```

Each `Ndb` object has its own counters. NDB API applications can read the values of the counters for use in optimization or monitoring. For multi-threaded clients which use more than one `Ndb` object concurrently, it is also possible to obtain a summed view of counters from all `Ndb` objects belonging to a given `Ndb_cluster_connection`.

Four sets of these counters are exposed. One set applies to the current session only; the other 3 are global. *This is in spite of the fact that their values can be obtained as either session or global status variables in the `mysql` client*. This means that specifying the `SESSION` or `GLOBAL` keyword with `SHOW STATUS` has no effect on the values reported for NDB API statistics status variables, and the value for each of these variables is the same whether the value is obtained from the equivalent column of the `SESSION_STATUS` or the `GLOBAL_STATUS` table.

- *Session counters (session specific)*

  Session counters relate to the `Ndb` objects in use by (only) the current session. Use of such objects by other MySQL clients does not influence these counts.

  In order to minimize confusion with standard MySQL session variables, we refer to the variables that correspond to these NDB API session counters as "`_session` variables", with a leading underscore.

- *Slave counters (global)*

  This set of counters relates to the `Ndb` objects used by the replication slave SQL thread, if any. If this `mysqld` does not act as a replication slave, or does not use `NDB` tables, then all of these counts are 0.

  We refer to the related status variables as "`_slave` variables" (with a leading underscore).

- *Injector counters (global)*

  Injector counters relate to the `Ndb` object used to listen to cluster events by the binary log injector thread. Even when not writing a binary log, `mysqld` processes attached to a MySQL Cluster continue to listen for some events, such as schema changes.

  We refer to the status variables that correspond to NDB API injector counters as "`_injector` variables" (with a leading underscore).

- *Server (Global) counters (global)*

  This set of counters relates to all `Ndb` objects currently used by this `mysqld`. This includes all MySQL client applications, the slave SQL thread (if any), the binlog injector, and the `NDB` utility thread.

  We refer to the status variables that correspond to these counters as “global variables” or “`mysqld`-level variables”.

You can obtain values for a particular set of variables by additionally filtering for the substring `session`, `slave`, or `injector` in the variable name (along with the common prefix `Ndb_api`). For `_session` variables, this can be done as shown here:

```
mysql> SHOW STATUS LIKE 'ndb_api%session';
+--------------------------------------------+---------+
| Variable_name                              | Value   |
+--------------------------------------------+---------+
| Ndb_api_wait_exec_complete_count_session   | 2       |
| Ndb_api_wait_scan_result_count_session     | 0       |
| Ndb_api_wait_meta_request_count_session    | 1       |
| Ndb_api_wait_nanos_count_session           | 8144375 |
| Ndb_api_bytes_sent_count_session           | 68      |
| Ndb_api_bytes_received_count_session       | 84      |
| Ndb_api_trans_start_count_session          | 1       |
| Ndb_api_trans_commit_count_session         | 1       |
| Ndb_api_trans_abort_count_session          | 0       |
| Ndb_api_trans_close_count_session          | 1       |
| Ndb_api_pk_op_count_session                | 1       |
| Ndb_api_uk_op_count_session                | 0       |
| Ndb_api_table_scan_count_session           | 0       |
| Ndb_api_range_scan_count_session           | 0       |
| Ndb_api_pruned_scan_count_session          | 0       |
| Ndb_api_scan_batch_count_session           | 0       |
| Ndb_api_read_row_count_session             | 1       |
| Ndb_api_trans_local_read_row_count_session | 1       |
+--------------------------------------------+---------+
18 rows in set (0.50 sec)
```

To obtain a listing of the NDB API `mysqld`-level status variables, filter for variable names beginning with `ndb_api` and ending in `_count`, like this:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS
    ->     WHERE VARIABLE_NAME LIKE 'ndb_api%count';
+------------------------------------+----------------+
| VARIABLE_NAME                      | VARIABLE_VALUE |
+------------------------------------+----------------+
| NDB_API_WAIT_EXEC_COMPLETE_COUNT   | 4              |
| NDB_API_WAIT_SCAN_RESULT_COUNT     | 3              |
| NDB_API_WAIT_META_REQUEST_COUNT    | 28             |
| NDB_API_WAIT_NANOS_COUNT           | 53756398       |
| NDB_API_BYTES_SENT_COUNT           | 1060           |
| NDB_API_BYTES_RECEIVED_COUNT       | 9724           |
| NDB_API_TRANS_START_COUNT          | 3              |
| NDB_API_TRANS_COMMIT_COUNT         | 2              |
| NDB_API_TRANS_ABORT_COUNT          | 0              |
| NDB_API_TRANS_CLOSE_COUNT          | 3              |
| NDB_API_PK_OP_COUNT                | 2              |
| NDB_API_UK_OP_COUNT                | 0              |
| NDB_API_TABLE_SCAN_COUNT           | 1              |
| NDB_API_RANGE_SCAN_COUNT           | 0              |
| NDB_API_PRUNED_SCAN_COUNT          | 0              |
| NDB_API_SCAN_BATCH_COUNT           | 0              |
| NDB_API_READ_ROW_COUNT             | 2              |
| NDB_API_TRANS_LOCAL_READ_ROW_COUNT | 2              |
| NDB_API_EVENT_DATA_COUNT           | 0              |
| NDB_API_EVENT_NONDATA_COUNT        | 0              |
| NDB_API_EVENT_BYTES_COUNT          | 0              |
+------------------------------------+----------------+
```

```
21 rows in set (0.09 sec)
```

Not all counters are reflected in all 4 sets of status variables. For the event counters `DataEventsRecvdCount`, `NondataEventsRecvdCount`, and `EventBytesRecvdCount`, only `_injector` and `mysqld`-level NDB API status variables are available:

```
mysql> SHOW STATUS LIKE 'ndb_api%event%';
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| Ndb_api_event_data_count_injector    | 0     |
| Ndb_api_event_nondata_count_injector | 0     |
| Ndb_api_event_bytes_count_injector   | 0     |
| Ndb_api_event_data_count             | 0     |
| Ndb_api_event_nondata_count          | 0     |
| Ndb_api_event_bytes_count            | 0     |
+--------------------------------------+-------+
6 rows in set (0.00 sec)
```

`_injector` status variables are not implemented for any other NDB API counters, as shown here:

```
mysql> SHOW STATUS LIKE 'ndb_api%injector%';
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| Ndb_api_event_data_count_injector    | 0     |
| Ndb_api_event_nondata_count_injector | 0     |
| Ndb_api_event_bytes_count_injector   | 0     |
+--------------------------------------+-------+
3 rows in set (0.00 sec)
```

The names of the status variables can easily be associated with the names of the corresponding counters. Each NDB API statistics counter is listed in the following table with a description as well as the names of any MySQL server status variables corresponding to this counter.

Counter Name	Description / Status Variables (by statistic type): Session; Slave; Injector; Server
`WaitExecCompleteCount`	Number of times thread has been blocked while waiting for execution of an operation to complete. Includes all `execute()` calls as well as implicit executes for blob operations and auto-increment not visible to clients. • `Ndb_api_wait_exec_complete_count_session` • `Ndb_api_wait_exec_complete_count_slave` • [none] • `Ndb_api_wait_exec_complete_count`
`WaitScanResultCount`	Number of times thread has been blocked while waiting for a scan-based signal, such waiting for additional results, or for a scan to close. • `Ndb_api_wait_scan_result_count_session`

Counter Name	Description
	Status Variables (by statistic type): • **Session** • **Slave** • **Injector** • **Server**
	• Ndb_api_wait_scan_result_count_slave • [none] • Ndb_api_wait_scan_result_count
WaitMetaRequestCount	Number of times thread has been blocked waiting for a metadata-based signal; this can occur when waiting for a DDL operation or for an epoch to be started (or ended).
	• Ndb_api_wait_meta_request_count_session • Ndb_api_wait_meta_request_count_slave • [none] • Ndb_api_wait_meta_request_count
WaitNanosCount	Total time (in nanoseconds) spent waiting for some type of signal from the data nodes.
	• Ndb_api_wait_nanos_count_session • Ndb_api_wait_nanos_count_slave • [none] • Ndb_api_wait_nanos_count
BytesSentCount	Amount of data (in bytes) sent to the data nodes
	• Ndb_api_bytes_sent_count_session • Ndb_api_bytes_sent_count_slave • [none] • Ndb_api_bytes_sent_count
BytesRecvdCount	Amount of data (in bytes) received from the data nodes
	• Ndb_api_bytes_received_count_session • Ndb_api_bytes_received_count_slave • [none] • Ndb_api_bytes_received_count
TransStartCount	Number of transactions started.
	• Ndb_api_trans_start_count_session • Ndb_api_trans_start_count_slave • [none]

Counter Name	Description **Status Variables (by statistic type):** • **Session** • **Slave** • **Injector** • **Server**
	• Ndb_api_trans_start_count
TransCommitCount	Number of transactions committed. • Ndb_api_trans_commit_count_session • Ndb_api_trans_commit_count_slave • [none] • Ndb_api_trans_commit_count
TransAbortCount	Number of transactions aborted. • Ndb_api_trans_abort_count_session • Ndb_api_trans_abort_count_slave • [none] • Ndb_api_trans_abort_count
TransCloseCount	Number of transactions aborted. (This value may be greater than the sum of TransCommitCount and TransAbortCount.) • Ndb_api_trans_close_count_session • Ndb_api_trans_close_count_slave • [none] • Ndb_api_trans_close_count
PkOpCount	Number of operations based on or using primary keys. This count includes blob-part table operations, implicit unlocking operations, and auto-increment operations, as well as primary key operations normally visible to MySQL clients. • Ndb_api_pk_op_count_session • Ndb_api_pk_op_count_slave • [none] • Ndb_api_pk_op_count
UkOpCount	Number of operations based on or using unique keys. • Ndb_api_uk_op_count_session • Ndb_api_uk_op_count_slave • [none] • Ndb_api_uk_op_count

Counter Name	Description
	Status Variables (by statistic type): • **Session** • **Slave** • **Injector** • **Server**
TableScanCount	Number of table scans that have been started. This includes scans of internal tables.
	• Ndb_api_table_scan_count_session • Ndb_api_table_scan_count_slave • [none] • Ndb_api_table_scan_count
RangeScanCount	Number of range scans that have been started.
	• Ndb_api_range_scan_count_session • Ndb_api_range_scan_count_slave • [none] • Ndb_api_range_scan_count
PrunedScanCount	Number of scans that have been pruned to a single partition.
	• Ndb_api_pruned_scan_count_session • Ndb_api_pruned_scan_count_slave • [none] • Ndb_api_pruned_scan_count
ScanBatchCount	Number of batches of rows received. (A *batch* in this context is a set of scan results from a single fragment.)
	• Ndb_api_scan_batch_count_session • Ndb_api_scan_batch_count_slave • [none] • Ndb_api_scan_batch_count
ReadRowCount	Total number of rows that have been read. Includes rows read using primary key, unique key, and scan operations.
	• Ndb_api_read_row_count_session • Ndb_api_read_row_count_slave • [none] • Ndb_api_read_row_count
TransLocalReadRowCount	Number of rows read from the data same node on which the transaction was being run.
	• Ndb_api_trans_local_read_row_count_session

Counter Name	Description **Status Variables (by statistic type):** • **Session** • **Slave** • **Injector** • **Server**
	• Ndb_api_trans_local_read_row_count_slave • [none] • Ndb_api_trans_local_read_row_count
DataEventsRecvdCount	Number of row change events received. • [none] • [none] • Ndb_api_event_data_count_injector • Ndb_api_event_data_count
NondataEventsRecvdCount	Number of events received, other than row change events. • [none] • [none] • Ndb_api_event_nondata_count_injector • Ndb_api_event_nondata_count
EventBytesRecvdCount	Number of bytes of events received. • [none] • [none] • Ndb_api_event_bytes_count_injector • Ndb_api_event_bytes_count

To see all counts of committed transactions—that is, all `TransCommitCount` counter status variables—you can filter the results of `SHOW STATUS` for the substring `trans_commit_count`, like this:

```
mysql> SHOW STATUS LIKE '%trans_commit_count%';
+------------------------------------+-------+
| Variable_name                      | Value |
+------------------------------------+-------+
| Ndb_api_trans_commit_count_session | 1     |
| Ndb_api_trans_commit_count_slave   | 0     |
| Ndb_api_trans_commit_count         | 2     |
+------------------------------------+-------+
3 rows in set (0.00 sec)
```

From this you can determine that 1 transaction has been committed in the current `mysql` client session, and 2 transactions have been committed on this `mysqld` since it was last restarted.

You can see how various NDB API counters are incremented by a given SQL statement by comparing the values of the corresponding `_session` status variables immediately before and after performing the statement. In this example, after getting the initial values from `SHOW STATUS`, we create in the `test` database an `NDB` table, named `t`, that has a single column:

```
mysql> SHOW STATUS LIKE 'ndb_api%session%';
+--------------------------------------------+--------+
| Variable_name                              | Value  |
+--------------------------------------------+--------+
| Ndb_api_wait_exec_complete_count_session   | 2      |
| Ndb_api_wait_scan_result_count_session     | 0      |
| Ndb_api_wait_meta_request_count_session    | 3      |
| Ndb_api_wait_nanos_count_session           | 820705 |
| Ndb_api_bytes_sent_count_session           | 132    |
| Ndb_api_bytes_received_count_session       | 372    |
| Ndb_api_trans_start_count_session          | 1      |
| Ndb_api_trans_commit_count_session         | 1      |
| Ndb_api_trans_abort_count_session          | 0      |
| Ndb_api_trans_close_count_session          | 1      |
| Ndb_api_pk_op_count_session                | 1      |
| Ndb_api_uk_op_count_session                | 0      |
| Ndb_api_table_scan_count_session           | 0      |
| Ndb_api_range_scan_count_session           | 0      |
| Ndb_api_pruned_scan_count_session          | 0      |
| Ndb_api_scan_batch_count_session           | 0      |
| Ndb_api_read_row_count_session             | 1      |
| Ndb_api_trans_local_read_row_count_session | 1      |
+--------------------------------------------+--------+
18 rows in set (0.00 sec)
mysql> USE test;
Database changed
mysql> CREATE TABLE t (c INT) ENGINE NDBCLUSTER;
Query OK, 0 rows affected (0.85 sec)
```

Now you can execute a new `SHOW STATUS` statement and observe the changes, as shown here (with the changed rows highlighted in the output):

```
mysql> SHOW STATUS LIKE 'ndb_api%session%';
+--------------------------------------------+-----------+
| Variable_name                              | Value     |
+--------------------------------------------+-----------+
| Ndb_api_wait_exec_complete_count_session   | 8         |
| Ndb_api_wait_scan_result_count_session     | 0         |
| Ndb_api_wait_meta_request_count_session    | 17        |
| Ndb_api_wait_nanos_count_session           | 706871709 |
| Ndb_api_bytes_sent_count_session           | 2376      |
| Ndb_api_bytes_received_count_session       | 3844      |
| Ndb_api_trans_start_count_session          | 4         |
| Ndb_api_trans_commit_count_session         | 4         |
| Ndb_api_trans_abort_count_session          | 0         |
| Ndb_api_trans_close_count_session          | 4         |
| Ndb_api_pk_op_count_session                | 6         |
| Ndb_api_uk_op_count_session                | 0         |
| Ndb_api_table_scan_count_session           | 0         |
| Ndb_api_range_scan_count_session           | 0         |
| Ndb_api_pruned_scan_count_session          | 0         |
| Ndb_api_scan_batch_count_session           | 0         |
| Ndb_api_read_row_count_session             | 2         |
| Ndb_api_trans_local_read_row_count_session | 1         |
+--------------------------------------------+-----------+
18 rows in set (0.00 sec)
```

Similarly, you can see the changes in the NDB API statistics counters caused by inserting a row into `t`: Insert the row, then run the same `SHOW STATUS` statement used in the previous example, as shown here:

```
mysql> INSERT INTO t VALUES (100);
Query OK, 1 row affected (0.00 sec)
mysql> SHOW STATUS LIKE 'ndb_api%session%';
+--------------------------------------------+-----------+
```

```
| Variable_name                              | Value     |
+--------------------------------------------+-----------+
| Ndb_api_wait_exec_complete_count_session   | 11        |
| Ndb_api_wait_scan_result_count_session     | 6         |
| Ndb_api_wait_meta_request_count_session    | 20        |
| Ndb_api_wait_nanos_count_session           | 707370418 |
| Ndb_api_bytes_sent_count_session           | 2724      |
| Ndb_api_bytes_received_count_session       | 4116      |
| Ndb_api_trans_start_count_session          | 7         |
| Ndb_api_trans_commit_count_session         | 6         |
| Ndb_api_trans_abort_count_session          | 0         |
| Ndb_api_trans_close_count_session          | 7         |
| Ndb_api_pk_op_count_session                | 8         |
| Ndb_api_uk_op_count_session                | 0         |
| Ndb_api_table_scan_count_session           | 1         |
| Ndb_api_range_scan_count_session           | 0         |
| Ndb_api_pruned_scan_count_session          | 0         |
| Ndb_api_scan_batch_count_session           | 0         |
| Ndb_api_read_row_count_session             | 3         |
| Ndb_api_trans_local_read_row_count_session | 2         |
+--------------------------------------------+-----------+
18 rows in set (0.00 sec)
```

We can make a number of observations from these results:

- Although we created `t` with no explicit primary key, 5 primary key operations were performed in doing so (the difference in the “before” and “after” values of `Ndb_api_pk_op_count_session`, or 6 minus 1). This reflects the creation of the hidden primary key that is a feature of all tables using the `NDB` storage engine.

- By comparing successive values for `Ndb_api_wait_nanos_count_session`, we can see that the NDB API operations implementing the `CREATE TABLE` statement waited much longer (706871709 - 820705 = 706051004 nanoseconds, or approximately 0.7 second) for responses from the data nodes than those executed by the `INSERT` (707370418 - 706871709 = 498709 ns or roughly .0005 second). The execution times reported for these statements in the `mysql` client correlate roughly with these figures.

  On platforms with without sufficient (nanosecond) time resolution, small changes in the value of the `WaitNanosCount` NDB API counter due to SQL statements that execute very quickly may not always be visible in the values of `Ndb_api_wait_nanos_count_session`, `Ndb_api_wait_nanos_count_slave`, or `Ndb_api_wait_nanos_count`.

- The `INSERT` statement incremented both the `ReadRowCount` and `TransLocalReadRowCount` NDB API statistics counters, as reflected by the increased values of `Ndb_api_read_row_count_session` and `Ndb_api_trans_local_read_row_count_session`.

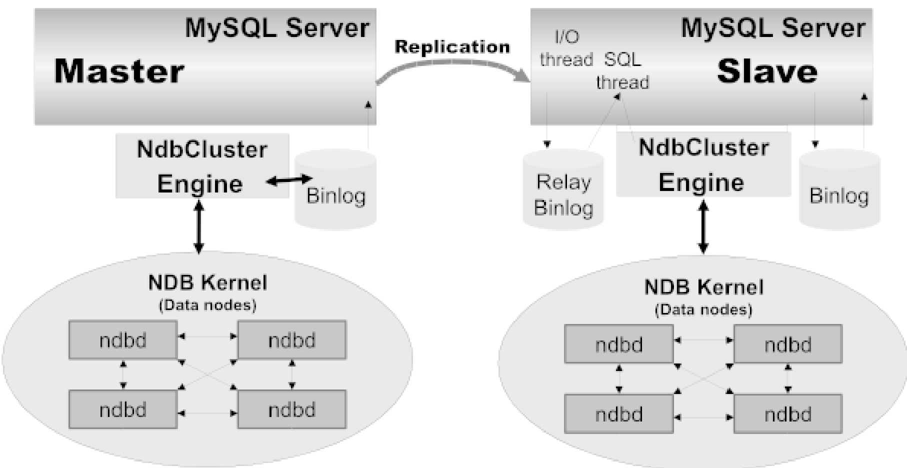
Chapter 8 MySQL Cluster Replication

Table of Contents

MySQL Cluster supports *asynchronous replication*, more usually referred to simply as “replication”. This section explains how to set up and manage a configuration in which one group of computers operating as a MySQL Cluster replicates to a second computer or group of computers. We assume some familiarity on the part of the reader with standard MySQL replication as discussed elsewhere in this Manual. (See Replication).

Normal (non-clustered) replication involves a “master” server and a “slave” server, the master being the source of the operations and data to be replicated and the slave being the recipient of these. In MySQL Cluster, replication is conceptually very similar but can be more complex in practice, as it may be extended to cover a number of different configurations including replicating between two complete clusters. Although a MySQL Cluster itself depends on the `NDB` storage engine for clustering functionality, it is not necessary to use `NDB` as the storage engine for the slave's copies of the replicated tables (see Replication from NDB to other storage engines). However, for maximum availability, it is possible (and preferable) to replicate from one MySQL Cluster to another, and it is this scenario that we discuss, as shown in the following figure:

Figure 8.1 MySQL Cluster-to-Cluster Replication Layout



In this scenario, the replication process is one in which successive states of a master cluster are logged and saved to a slave cluster. This process is accomplished by a special thread known as the NDB binary log injector thread, which runs on each MySQL server and produces a binary log (`binlog`). This thread ensures that all changes in the cluster producing the binary log—and not just those changes that are effected through the MySQL Server—are inserted into the binary log with the correct serialization order. We refer to the MySQL replication master and replication slave servers as replication servers or replication nodes, and the data flow or line of communication between them as a *replication channel*.

For information about performing point-in-time recovery with MySQL Cluster and MySQL Cluster Replication, see Section 8.9.2, “Point-In-Time Recovery Using MySQL Cluster Replication”.

NDB API _slave status variables. NDB API counters can provide enhanced monitoring capabilities on MySQL Cluster replication slaves. These are implemented as NDB statistics `_slave` status variables, as seen in the output of `SHOW STATUS`, or in the results of queries against the `SESSION_STATUS` or `GLOBAL_STATUS` table in a `mysql` client session connected to a MySQL Server that is acting as a slave in MySQL Cluster Replication. By comparing the values of these status variables before and after the execution of statements affecting replicated `NDB` tables, you can observe the corresponding actions taken on the NDB API level by the slave, which can be useful when monitoring or troubleshooting MySQL Cluster Replication. Section 7.15, “NDB API Statistics Counters and Variables”, provides additional information.

Replication from NDB to non-NDB tables. It is possible to replicate `NDB` tables from a MySQL Cluster acting as the master to tables using other MySQL storage engines such as `InnoDB` or `MyISAM` on a slave `mysqld`. This is subject to a number of conditions; see Replication from NDB to other storage engines, and Replication from NDB to a nontransactional storage engine, for more information.

8.1 MySQL Cluster Replication: Abbreviations and Symbols

Throughout this section, we use the following abbreviations or symbols for referring to the master and slave clusters, and to processes and commands run on the clusters or cluster nodes:

Symbol or Abbreviation	Description (Refers to...)
`M`	The cluster serving as the (primary) replication master
`S`	The cluster acting as the (primary) replication slave
`shellM>`	Shell command to be issued on the master cluster
`mysqlM>`	MySQL client command issued on a single MySQL server running as an SQL node on the master cluster
`mysqlM*>`	MySQL client command to be issued on all SQL nodes participating in the replication master cluster
`shellS>`	Shell command to be issued on the slave cluster
`mysqlS>`	MySQL client command issued on a single MySQL server running as an SQL node on the slave cluster
`mysqlS*>`	MySQL client command to be issued on all SQL nodes participating in the replication slave cluster
`C`	Primary replication channel
`C'`	Secondary replication channel
`M'`	Secondary replication master
`S'`	Secondary replication slave

8.2 General Requirements for MySQL Cluster Replication

A replication channel requires two MySQL servers acting as replication servers (one each for the master and slave). For example, this means that in the case of a replication setup with two replication channels (to provide an extra channel for redundancy), there will be a total of four replication nodes, two per cluster.

Replication of a MySQL Cluster as described in this section and those following is dependent on row-based replication. This means that the replication master MySQL server must be running with `--binlog-format=ROW` or `--binlog-format=MIXED`, as described in Section 8.6, "Starting MySQL Cluster Replication (Single Replication Channel)". For general information about row-based replication, see Replication Formats.

> **Important**
>
> If you attempt to use MySQL Cluster Replication with `--binlog-format=STATEMENT`, replication fails to work properly because the `ndb_binlog_index` table on the master and the `epoch` column of the `ndb_apply_status` table on the slave are not updated (see Section 8.4, "MySQL Cluster Replication Schema and Tables"). Instead, only updates on the MySQL server acting as the replication master propagate to the slave, and no updates from any other SQL nodes on the master cluster are replicated.
>
> The default value for the `--binlog-format` option in MySQL Cluster NDB 7.3 is `MIXED`.

Each MySQL server used for replication in either cluster must be uniquely identified among all the MySQL replication servers participating in either cluster (you cannot have replication servers on both the master and slave clusters sharing the same ID). This can be done by starting each SQL node using the `--server-id=`*`id`* option, where *`id`* is a unique integer. Although it is not strictly necessary, we will assume for purposes of this discussion that all MySQL Cluster binaries are of the same release version.

It is generally true in MySQL Replication that both MySQL servers (`mysqld` processes) involved must be compatible with one another with respect to both the version of the replication protocol used and the SQL feature sets which they support (see Replication Compatibility Between MySQL Versions). It is due to such differences between the binaries in the MySQL Cluster and MySQL Server 5.6 distributions that MySQL Cluster Replication has the additional requirement that both `mysqld` binaries come from a MySQL Cluster distribution. The simplest and easiest way to assure that the `mysqld` servers are compatible is to use the same MySQL Cluster distribution for all master and slave `mysqld` binaries.

We assume that the slave server or cluster is dedicated to replication of the master, and that no other data is being stored on it.

All `NDB` tables being replicated must be created using a MySQL server and client. Tables and other database objects created using the NDB API (with, for example, `Dictionary::createTable()`) are not visible to a MySQL server and so are not replicated. Updates by NDB API applications to existing tables that were created using a MySQL server can be replicated.

> **Note**
>
> It is possible to replicate a MySQL Cluster using statement-based replication. However, in this case, the following restrictions apply:
>
> - All updates to data rows on the cluster acting as the master must be directed to a single MySQL server.
> - It is not possible to replicate a cluster using multiple simultaneous MySQL replication processes.
> - Only changes made at the SQL level are replicated.

> These are in addition to the other limitations of statement-based replication as opposed to row-based replication; see Advantages and Disadvantages of Statement-Based and Row-Based Replication, for more specific information concerning the differences between the two replication formats.

8.3 Known Issues in MySQL Cluster Replication

This section discusses known problems or issues when using replication with MySQL Cluster NDB 7.3.

Loss of master-slave connection. A loss of connection can occur either between the replication master SQL node and the replication slave SQL node, or between the replication master SQL node and the data nodes in the master cluster. In the latter case, this can occur not only as a result of loss of physical connection (for example, a broken network cable), but due to the overflow of data node event buffers; if the SQL node is too slow to respond, it may be dropped by the cluster (this is controllable to some degree by adjusting the `MaxBufferedEpochs` and `TimeBetweenEpochs` configuration parameters). If this occurs, *it is entirely possible for new data to be inserted into the master cluster without being recorded in the replication master's binary log*. For this reason, to guarantee high availability, it is extremely important to maintain a backup replication channel, to monitor the primary channel, and to fail over to the secondary replication channel when necessary to keep the slave cluster synchronized with the master. MySQL Cluster is not designed to perform such monitoring on its own; for this, an external application is required.

The replication master issues a “gap” event when connecting or reconnecting to the master cluster. (A gap event is a type of “incident event,” which indicates an incident that occurs that affects the contents of the database but that cannot easily be represented as a set of changes. Examples of incidents are server crashes, database resynchronization, (some) software updates, and (some) hardware changes.) When the slave encounters a gap in the replication log, it stops with an error message. This message is available in the output of `SHOW SLAVE STATUS`, and indicates that the SQL thread has stopped due to an incident registered in the replication stream, and that manual intervention is required. See Section 8.8, “Implementing Failover with MySQL Cluster Replication”, for more information about what to do in such circumstances.

> **Important**
>
> Because MySQL Cluster is not designed on its own to monitor replication status or provide failover, if high availability is a requirement for the slave server or cluster, then you must set up multiple replication lines, monitor the master `mysqld` on the primary replication line, and be prepared fail over to a secondary line if and as necessary. This must be done manually, or possibly by means of a third-party application. For information about implementing this type of setup, see Section 8.7, “Using Two Replication Channels for MySQL Cluster Replication”, and Section 8.8, “Implementing Failover with MySQL Cluster Replication”.
>
> However, if you are replicating from a standalone MySQL server to a MySQL Cluster, one channel is usually sufficient.

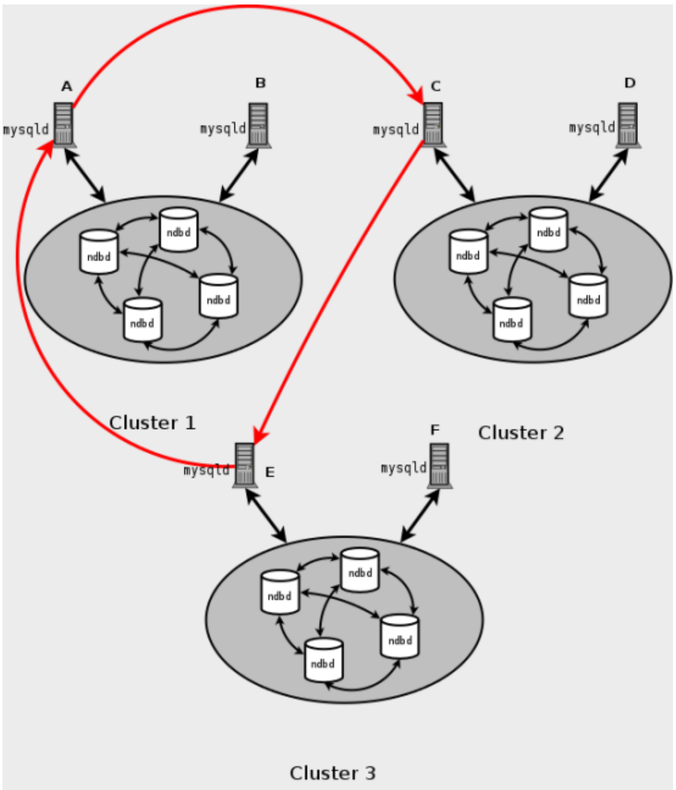
Circular replication. MySQL Cluster Replication supports circular replication, as shown in the next example. The replication setup involves three MySQL Clusters numbered 1, 2, and 3, in which Cluster 1 acts as the replication master for Cluster 2, Cluster 2 acts as the master for Cluster 3, and Cluster 3 acts as the master for Cluster 1, thus completing the circle. Each MySQL Cluster has two SQL nodes, with SQL nodes A and B belonging to Cluster 1, SQL nodes C and D belonging to Cluster 2, and SQL nodes E and F belonging to Cluster 3.

Circular replication using these clusters is supported as long as the following conditions are met:

- The SQL nodes on all masters and slaves are the same
- All SQL nodes acting as replication masters and slaves are started using the `--log-slave-updates` option

This type of circular replication setup is shown in the following diagram:

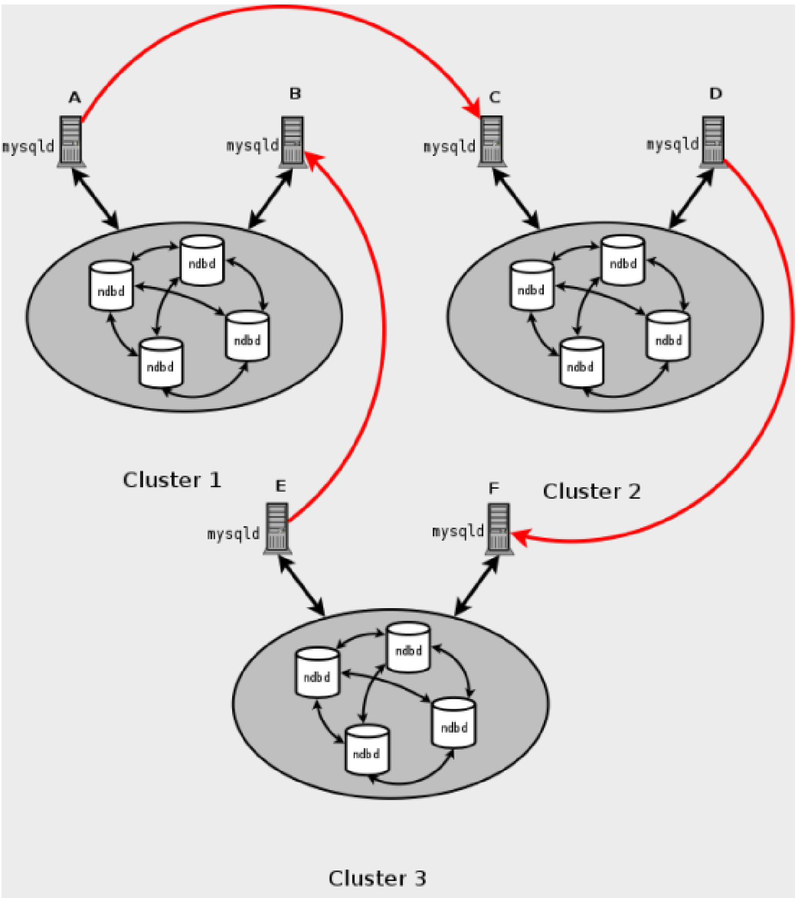
Figure 8.2 MySQL Cluster Circular Replication With All Masters As Slaves



In this scenario, SQL node A in Cluster 1 replicates to SQL node C in Cluster 2; SQL node C replicates to SQL node E in Cluster 3; SQL node E replicates to SQL node A. In other words, the replication line (indicated by the red arrows in the diagram) directly connects all SQL nodes used as replication masters and slaves.

It should also be possible to set up circular replication in which not all master SQL nodes are also slaves, as shown here:

Figure 8.3 MySQL Cluster Circular Replication Where Not All Masters Are Slaves



In this case, different SQL nodes in each cluster are used as replication masters and slaves. However, you must *not* start any of the SQL nodes using `--log-slave-updates`. This type of circular replication scheme for MySQL Cluster, in which the line of replication (again indicated by the red arrows in the diagram) is discontinuous, should be possible, but it should be noted that it has not yet been thoroughly tested and must therefore still be considered experimental.

> **Note**
>
> The `NDB` storage engine uses *idempotent execution mode*, which suppresses duplicate-key and other errors that otherwise break circular replication of MySQL Cluster. This is equivalent to setting the global `slave_exec_mode` system variable to `IDEMPOTENT`, although this is not necessary in MySQL Cluster replication, since MySQL Cluster sets this variable automatically and ignores any attempts to set it explicitly.

MySQL Cluster replication and primary keys. In the event of a node failure, errors in replication of `NDB` tables without primary keys can still occur, due to the possibility of duplicate rows being inserted in such cases. For this reason, it is highly recommended that all `NDB` tables being replicated have primary keys.

MySQL Cluster Replication and Unique Keys. In older versions of MySQL Cluster, operations that updated values of unique key columns of `NDB` tables could result in duplicate-key errors when replicated. This issue is solved for replication between `NDB` tables by deferring unique key checks until after all table row updates have been performed.

Deferring constraints in this way is currently supported only by `NDB`. Thus, updates of unique keys when replicating from `NDB` to a different storage engine such as `MyISAM` or `InnoDB` are still not supported.

The problem encountered when replicating without deferred checking of unique key updates can be illustrated using NDB table such as t, is created and populated on the master (and replicated to a slave that does not support deferred unique key updates) as shown here:

```
CREATE TABLE t (
    p INT PRIMARY KEY,
    c INT,
    UNIQUE KEY u (c)
)   ENGINE NDB;
INSERT INTO t
    VALUES (1,1), (2,2), (3,3), (4,4), (5,5);
```

The following UPDATE statement on t succeeded on the master, since the rows affected are processed in the order determined by the ORDER BY option, performed over the entire table:

```
UPDATE t SET c = c - 1 ORDER BY p;
```

However, the same statement failed with a duplicate key error or other constraint violation on the slave, because the ordering of the row updates was done for one partition at a time, rather than for the table as a whole.

> **Note**
>
> Every NDB table is implicitly partitioned by key when it is created. See KEY Partitioning, for more information.

GTIDs not supported. Replication using global transaction IDs is not compatible with the NDB storage engine, and is not supported. Enabling GTIDs is likely to cause MySQL Cluster Replication to fail.

Multi-threaded slaves not supported. MySQL Cluster does not support multi-threaded slaves, and setting related system variables such as slave_parallel_workers, slave_checkpoint_group, and slave_checkpoint_group (or the equivalent mysqld startup options) has no effect.

This is because the slave may not be able to separate transactions occurring in one database from those in another if they are written within the same epoch. In addition, every transaction handled by the NDB storage engine involves at least two databases—the target database and the mysql system database—due to the requirement for updating the mysql.ndb_apply_status table (see Section 8.4, “MySQL Cluster Replication Schema and Tables”). This in turn breaks the requirement for multi-threading that the transaction is specific to a given database.

Restarting with --initial. Restarting the cluster with the --initial option causes the sequence of GCI and epoch numbers to start over from 0. (This is generally true of MySQL Cluster and not limited to replication scenarios involving Cluster.) The MySQL servers involved in replication should in this case be restarted. After this, you should use the RESET MASTER and RESET SLAVE statements to clear the invalid ndb_binlog_index and ndb_apply_status tables, respectively.

Replication from NDB to other storage engines. It is possible to replicate an NDB table on the master to a table using a different storage engine on the slave, taking into account the restrictions listed here:

- Multi-master and circular replication are not supported (tables on both the master and the slave must use the NDB storage engine for this to work).
- Using a storage engine which does not perform binary logging for slave tables requires special handling.
- Use of a nontransactional storage engine for slave tables also requires special handling.
- The master mysqld must be started with --ndb-log-update-as-write=0 [474] or --ndb-log-update-as-write=OFF.

The next few paragraphs provide additional information about each of the issues just described.

Multiple masters not supported when replicating NDB to other storage engines. For replication from `NDB` to a different storage engine, the relationship between the two databases must be a simple master-slave one. This means that circular or master-master replication is not supported between MySQL Cluster and other storage engines.

In addition, it is not possible to configure more than one replication channel when replicating between `NDB` and a different storage engine. (However, a MySQL Cluster database *can* simultaneously replicate to multiple slave MySQL Cluster databases.) If the master uses `NDB` tables, it is still possible to have more than one MySQL Server maintain a binary log of all changes; however, for the slave to change masters (fail over), the new master-slave relationship must be explicitly defined on the slave.

Replicating NDB to a slave storage engine that does not perform binary logging. If you attempt to replicate from a MySQL Cluster to a slave that uses a storage engine that does not handle its own binary logging, the replication process aborts with the error `Binary logging not possible ... Statement cannot be written atomically since more than one engine involved and at least one engine is self-logging` (Error `1595`). It is possible to work around this issue in one of the following ways:

- **Turn off binary logging on the slave.** This can be accomplished by setting `sql_log_bin = 0`.
- **Change the storage engine used for the mysql.ndb_apply_status table.** Causing this table to use an engine that does not handle its own binary logging can also eliminate the conflict. This can be done by issuing a statement such as `ALTER TABLE mysql.ndb_apply_status ENGINE=MyISAM` on the slave. It is safe to do this when using a non-`NDB` storage engine on the slave, since you do not then need to worry about keeping multiple slave SQL nodes synchronized.
- **Filter out changes to the mysql.ndb_apply_status table on the slave.** This can be done by starting the slave SQL node with `--replicate-ignore-table=mysql.ndb_apply_status`. If you need for other tables to be ignored by replication, you might wish to use an appropriate `--replicate-wild-ignore-table` option instead.

> **Important**
>
> You should *not* disable replication or binary logging of `mysql.ndb_apply_status` or change the storage engine used for this table when replicating from one MySQL Cluster to another. See Replication and binary log filtering rules with replication between MySQL Clusters, for details.

Replication from NDB to a nontransactional storage engine. When replicating from `NDB` to a nontransactional storage engine such as `MyISAM`, you may encounter unnecessary duplicate key errors when replicating `INSERT ... ON DUPLICATE KEY UPDATE` statements. You can suppress these by using `--ndb-log-update-as-write=0` [474], which forces updates to be logged as writes (rather than as updates).

Replication and binary log filtering rules with replication between MySQL Clusters. If you are using any of the options `--replicate-do-*`, `--replicate-ignore-*`, `--binlog-do-db`, or `--binlog-ignore-db` to filter databases or tables being replicated, care must be taken not to block replication or binary logging of the `mysql.ndb_apply_status`, which is required for replication between MySQL Clusters to operate properly. In particular, you must keep in mind the following:

1. Using `--replicate-do-db=db_name` (and no other `--replicate-do-*` or `--replicate-ignore-*` options) means that *only* tables in database *db_name* are replicated. In this case, you should also use `--replicate-do-db=mysql`, `--binlog-do-db=mysql`, or `--replicate-do-table=mysql.ndb_apply_status` to ensure that `mysql.ndb_apply_status` is populated on slaves.

   Using `--binlog-do-db=db_name` (and no other `--binlog-do-db` options) means that changes *only* to tables in database *db_name* are written to the binary log. In this case, you should also use `--replicate-do-db=mysql`, `--binlog-do-db=mysql`, or `--replicate-do-table=mysql.ndb_apply_status` to ensure that `mysql.ndb_apply_status` is populated on slaves.

2. Using `--replicate-ignore-db=mysql` means that no tables in the `mysql` database are replicated. In this case, you should also use `--replicate-do-table=mysql.ndb_apply_status` to ensure that `mysql.ndb_apply_status` is replicated.

   Using `--binlog-ignore-db=mysql` means that no changes to tables in the `mysql` database are written to the binary log. In this case, you should also use `--replicate-do-table=mysql.ndb_apply_status` to ensure that `mysql.ndb_apply_status` is replicated.

You should also remember that each replication rule requires the following:

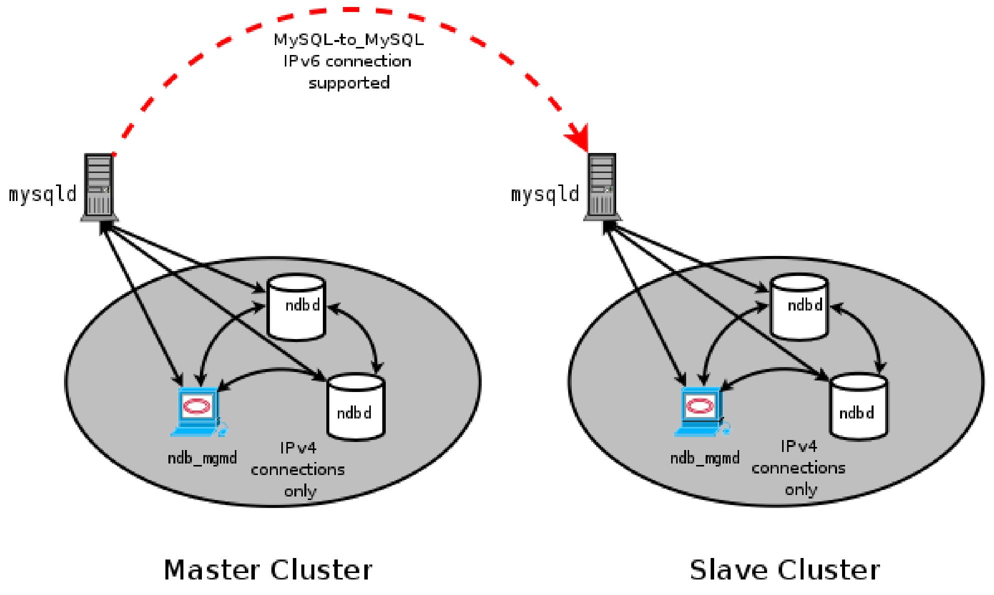
1. Its own `--replicate-do-*` or `--replicate-ignore-*` option, and that multiple rules cannot be expressed in a single replication filtering option. For information about these rules, see Replication and Binary Logging Options and Variables.

2. Its own `--binlog-do-db` or `--binlog-ignore-db` option, and that multiple rules cannot be expressed in a single binary log filtering option. For information about these rules, see The Binary Log.

If you are replicating a MySQL Cluster to a slave that uses a storage engine other than `NDB`, the considerations just given previously may not apply, as discussed elsewhere in this section.

MySQL Cluster Replication and IPv6. Currently, the NDB API and MGM API do not support IPv6. However, MySQL Servers—including those acting as SQL nodes in a MySQL Cluster—can use IPv6 to contact other MySQL Servers. This means that you can replicate between MySQL Clusters using IPv6 to connect the master and slave SQL nodes as shown by the dotted arrow in the following diagram:

Figure 8.4 Replication Between SQL Nodes Connected Using IPv6



However, all connections originating *within* the MySQL Cluster—represented in the preceding diagram by solid arrows—must use IPv4. In other words, all MySQL Cluster data nodes, management servers, and management clients must be accessible from one another using IPv4. In addition, SQL nodes must use IPv4 to communicate with the cluster.

Since there is currently no support in the NDB and MGM APIs for IPv6, any applications written using these APIs must also make all connections using IPv4.

Attribute promotion and demotion. MySQL Cluster Replication includes support for attribute promotion and demotion. The implementation of the latter distinguishes between lossy and non-lossy type conversions, and their use on the slave can be controlled by setting the `slave_type_conversions` global server system variable.

For more information about attribute promotion and demotion in MySQL Cluster, see Row-based replication: attribute promotion and demotion.

8.4 MySQL Cluster Replication Schema and Tables

Replication in MySQL Cluster makes use of a number of dedicated tables in the `mysql` database on each MySQL Server instance acting as an SQL node in both the cluster being replicated and the replication slave (whether the slave is a single server or a cluster). These tables are created during the MySQL installation process by the `mysql_install_db` script, and include a table for storing the binary log's indexing data. Since the `ndb_binlog_index` table is local to each MySQL server and does not participate in clustering, it uses the `MyISAM` storage engine. This means that it must be created separately on each `mysqld` participating in the master cluster. (However, the binary log itself contains updates from all MySQL servers in the cluster to be replicated.) This table is defined as follows:

```
CREATE TABLE `ndb_binlog_index` (
    `Position` BIGINT(20) UNSIGNED NOT NULL,
    `File` VARCHAR(255) NOT NULL,
    `epoch` BIGINT(20) UNSIGNED NOT NULL,
    `inserts` INT(10) UNSIGNED NOT NULL,
    `updates` INT(10) UNSIGNED NOT NULL,
    `deletes` INT(10) UNSIGNED NOT NULL,
    `schemaops` INT(10) UNSIGNED NOT NULL,
    `orig_server_id` INT(10) UNSIGNED NOT NULL,
    `orig_epoch` BIGINT(20) UNSIGNED NOT NULL,
    `gci` INT(10) UNSIGNED NOT NULL,
    `next_position` bigint(20) unsigned NOT NULL,
    `next_file` varchar(255) NOT NULL,
    PRIMARY KEY (`epoch`,`orig_server_id`,`orig_epoch`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

The size of this table is dependent on the number of epochs per binary log file and the number of binary log files. The number of epochs per binary log file normally depends on the amount of binary log generated per epoch and the size of the binary log file, with smaller epochs resulting in more epochs per file. It should be noted that empty epochs produce result in inserts to the `ndb_binlog_index` table, even when the `--ndb-log-empty-epochs` option is `OFF`, meaning that the number of entries per file depends on the length of time that the file is in use; that is,

```
[number of epochs per file] = [time spent per file] / TimeBetweenEpochs
```

A busy MySQL Cluster writes to the binary log regularly and presumably rotates binary log files more quickly than a quiet one. This means that a “quiet” MySQL Cluster with `--ndb-log-empty-epochs=ON` can actually have a much higher number of `ndb_binlog_index` rows per file than one with a great deal of activity.

When `mysqld` is started with the `--ndb-log-orig` option, the `orig_server_id` and `orig_epoch` columns store, respectively, the ID of the server on which the event originated and the epoch in which the event took place on the originating server, which is useful in MySQL Cluster replication setups employing multiple masters. The `SELECT` statement used to find the closest binary log position to the highest applied epoch on the slave in a multi-master setup (see Section 8.10, “MySQL Cluster Replication: Multi-Master and Circular Replication”) employs these two columns, which are not indexed. This can lead to performance issues when trying to fail over, since the query must perform a table scan, especially when the master has been running with `--ndb-log-empty-epochs=ON`. You can improve multi-master failover times by adding an index to these columns, as shown here:

```
ALTER TABLE mysql.ndb_binlog_index
```

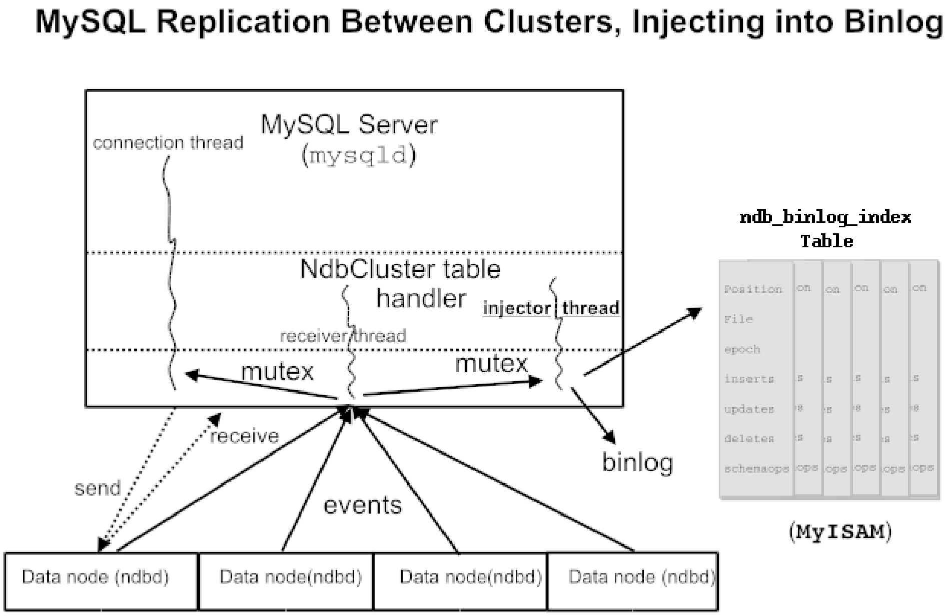
```
    ADD INDEX orig_lookup USING BTREE (orig_server_id, orig_epoch);
```

Adding this index provides no benefit when replicating from a single master to a single slave, since the query used to get the binary log position in such cases makes no use of `orig_server_id` or `orig_epoch`.

See Section 8.8, "Implementing Failover with MySQL Cluster Replication", for more information about using the `next_position` and `next_file` columns.

The following figure shows the relationship of the MySQL Cluster replication master server, its binary log injector thread, and the `mysql.ndb_binlog_index` table.

Figure 8.5 The Replication Master Cluster



An additional table, named `ndb_apply_status`, is used to keep a record of the operations that have been replicated from the master to the slave. Unlike the case with `ndb_binlog_index`, the data in this table is not specific to any one SQL node in the (slave) cluster, and so `ndb_apply_status` can use the `NDBCLUSTER` storage engine, as shown here:

```
CREATE TABLE `ndb_apply_status` (
    `server_id`   INT(10) UNSIGNED NOT NULL,
    `epoch`       BIGINT(20) UNSIGNED NOT NULL,
    `log_name`    VARCHAR(255) CHARACTER SET latin1 COLLATE latin1_bin NOT NULL,
    `start_pos`   BIGINT(20) UNSIGNED NOT NULL,
    `end_pos`     BIGINT(20) UNSIGNED NOT NULL,
    PRIMARY KEY (`server_id`) USING HASH
) ENGINE=NDBCLUSTER   DEFAULT CHARSET=latin1;
```

The `ndb_apply_status` table is populated only on slaves, which means that, on the master, this table never contains any rows; thus, there is no need to allow for `DataMemory` or `IndexMemory` to be allotted to `ndb_apply_status` there.

Because this table is populated from data originating on the master, it should be allowed to replicate; any replication filtering or binary log filtering rules that inadvertently prevent the slave from updating `ndb_apply_status` or the master from writing into the binary log may prevent replication between clusters from operating properly. For more information about potential problems arising from such filtering rules, see Replication and binary log filtering rules with replication between MySQL Clusters.

The ndb_binlog_index and ndb_apply_status tables are created in the mysql database because they should not be explicitly replicated by the user. User intervention is normally not required to create or maintain either of these tables, since both ndb_binlog_index and the ndb_apply_status are maintained by the NDB binary log (binlog) injector thread. This keeps the master mysqld process updated to changes performed by the NDB storage engine. The NDB *binlog injector thread* receives events directly from the NDB storage engine. The NDB injector is responsible for capturing all the data events within the cluster, and ensures that all events which change, insert, or delete data are recorded in the ndb_binlog_index table. The slave I/O thread transfers the events from the master's binary log to the slave's relay log.

However, it is advisable to check for the existence and integrity of these tables as an initial step in preparing a MySQL Cluster for replication. It is possible to view event data recorded in the binary log by querying the mysql.ndb_binlog_index table directly on the master. This can be also be accomplished using the SHOW BINLOG EVENTS statement on either the replication master or slave MySQL servers. (See SHOW BINLOG EVENTS Syntax.)

You can also obtain useful information from the output of SHOW ENGINE NDB STATUS.

The ndb_schema table is used to track schema changes made to NDB tables. It is defined as shown here:

```
CREATE TABLE ndb_schema (
    `db` VARBINARY(63) NOT NULL,
    `name` VARBINARY(63) NOT NULL,
    `slock` BINARY(32) NOT NULL,
    `query` BLOB NOT NULL,
    `node_id` INT UNSIGNED NOT NULL,
    `epoch` BIGINT UNSIGNED NOT NULL,
    `id` INT UNSIGNED NOT NULL,
    `version` INT UNSIGNED NOT NULL,
    `type` INT UNSIGNED NOT NULL,
    PRIMARY KEY USING HASH (db,name)
) ENGINE=NDB   DEFAULT CHARSET=latin1;
```

Unlike the two tables previously mentioned in this section, the ndb_schema table is not visible either to MySQL SHOW statements, or in any INFORMATION_SCHEMA tables; however, it can be seen in the output of ndb_show_tables, as shown here:

```
shell> ndb_show_tables -t 2
id    type              state    logging database    schema   name
4     UserTable         Online   Yes     mysql       def      ndb_apply_status
5     UserTable         Online   Yes     ndbworld    def      City
6     UserTable         Online   Yes     ndbworld    def      Country
3     UserTable         Online   Yes     mysql       def      NDB$BLOB_2_3
7     UserTable         Online   Yes     ndbworld    def      CountryLanguage
2     UserTable         Online   Yes     mysql       def      ndb_schema
NDBT_ProgramExit: 0 - OK
```

It is also possible to SELECT from this table in mysql and other MySQL client applications, as shown here:

```
mysql> SELECT * FROM mysql.ndb_schema WHERE name='City' \G
*************************** 1. row ***************************
     db: ndbworld
   name: City
  slock:
  query: alter table City engine=ndb
node_id: 4
  epoch: 0
     id: 0
version: 0
   type: 7
1 row in set (0.00 sec)
```

This can sometimes be useful when debugging applications.

> **Note**
>
> When performing schema changes on NDB tables, applications should wait until the ALTER TABLE statement has returned in the MySQL client connection that issued the statement before attempting to use the updated definition of the table.

If the ndb_apply_status table or the ndb_schema table does not exist on the slave, ndb_restore re-creates the missing table or tables (Bug #14612).

Conflict resolution for MySQL Cluster Replication requires the presence of an additional mysql.ndb_replication table. Currently, this table must be created manually. For information about how to do this, see Section 8.11, "MySQL Cluster Replication Conflict Resolution".

8.5 Preparing the MySQL Cluster for Replication

Preparing the MySQL Cluster for replication consists of the following steps:

1. Check all MySQL servers for version compatibility (see Section 8.2, "General Requirements for MySQL Cluster Replication").

2. Create a slave account on the master Cluster with the appropriate privileges:

   ```
   mysqlM> GRANT REPLICATION SLAVE
        -> ON *.* TO 'slave_user'@'slave_host'
        -> IDENTIFIED BY 'slave_password';
   ```

   In the previous statement, *slave_user* is the slave account user name, *slave_host* is the host name or IP address of the replication slave, and *slave_password* is the password to assign to this account.

   For example, to create a slave user account with the name "myslave," logging in from the host named "rep-slave," and using the password "53cr37," use the following GRANT statement:

   ```
   mysqlM> GRANT REPLICATION SLAVE
        -> ON *.* TO 'myslave'@'rep-slave'
        -> IDENTIFIED BY '53cr37';
   ```

   For security reasons, it is preferable to use a unique user account—not employed for any other purpose—for the replication slave account.

3. Configure the slave to use the master. Using the MySQL Monitor, this can be accomplished with the CHANGE MASTER TO statement:

   ```
   mysqlS> CHANGE MASTER TO
        -> MASTER_HOST='master_host',
        -> MASTER_PORT=master_port,
        -> MASTER_USER='slave_user',
        -> MASTER_PASSWORD='slave_password';
   ```

   In the previous statement, *master_host* is the host name or IP address of the replication master, *master_port* is the port for the slave to use for connecting to the master, *slave_user* is the user name set up for the slave on the master, and *slave_password* is the password set for that user account in the previous step.

   For example, to tell the slave to replicate from the MySQL server whose host name is "rep-master," using the replication slave account created in the previous step, use the following statement:

   ```
   mysqlS> CHANGE MASTER TO
   ```

```
    -> MASTER_HOST='rep-master',
    -> MASTER_PORT=3306,
    -> MASTER_USER='myslave',
    -> MASTER_PASSWORD='53cr37';
```

For a complete list of options that can be used with this statement, see CHANGE MASTER TO Syntax.

To provide replication backup capability, you also need to add an `--ndb-connectstring` option to the slave's `my.cnf` file prior to starting the replication process. See Section 8.9, "MySQL Cluster Backups With MySQL Cluster Replication", for details.

For additional options that can be set in `my.cnf` for replication slaves, see Replication and Binary Logging Options and Variables.

4. If the master cluster is already in use, you can create a backup of the master and load this onto the slave to cut down on the amount of time required for the slave to synchronize itself with the master. If the slave is also running MySQL Cluster, this can be accomplished using the backup and restore procedure described in Section 8.9, "MySQL Cluster Backups With MySQL Cluster Replication".

```
ndb-connectstring=management_host[:port]
```

In the event that you are *not* using MySQL Cluster on the replication slave, you can create a backup with this command on the replication master:

```
shellM> mysqldump --master-data=1
```

Then import the resulting data dump onto the slave by copying the dump file over to the slave. After this, you can use the `mysql` client to import the data from the dumpfile into the slave database as shown here, where *`dump_file`* is the name of the file that was generated using `mysqldump` on the master, and *`db_name`* is the name of the database to be replicated:

```
shellS> mysql -u root -p db_name < dump_file
```

For a complete list of options to use with `mysqldump`, see `mysqldump` — A Database Backup Program.

> **Note**
>
> If you copy the data to the slave in this fashion, you should make sure that the slave is started with the `--skip-slave-start` option on the command line, or else include `skip-slave-start` in the slave's `my.cnf` file to keep it from trying to connect to the master to begin replicating before all the data has been loaded. Once the data loading has completed, follow the additional steps outlined in the next two sections.

5. Ensure that each MySQL server acting as a replication master is configured with a unique server ID, and with binary logging enabled, using the row format. (See Replication Formats.) These options can be set either in the master server's `my.cnf` file, or on the command line when starting the master `mysqld` process. See Section 8.6, "Starting MySQL Cluster Replication (Single Replication Channel)", for information regarding the latter option.

8.6 Starting MySQL Cluster Replication (Single Replication Channel)

This section outlines the procedure for starting MySQL Cluster replication using a single replication channel.

1. Start the MySQL replication master server by issuing this command:

```
shellM> mysqld --ndbcluster --server-id=id \
        --log-bin &
```

In the previous statement, *id* is this server's unique ID (see Section 8.2, "General Requirements for MySQL Cluster Replication"). This starts the server's `mysqld` process with binary logging enabled using the proper logging format.

> **Note**
>
> You can also start the master with `--binlog-format=MIXED`, in which case row-based replication is used automatically when replicating between clusters. `STATEMENT` based binary logging is not supported for MySQL Cluster Replication (see Section 8.2, "General Requirements for MySQL Cluster Replication").

2. Start the MySQL replication slave server as shown here:

```
shellS> mysqld --ndbcluster --server-id=id &
```

In the command just shown, *id* is the slave server's unique ID. It is not necessary to enable logging on the replication slave.

> **Note**
>
> You should use the `--skip-slave-start` option with this command or else you should include `skip-slave-start` in the slave server's `my.cnf` file, unless you want replication to begin immediately. With the use of this option, the start of replication is delayed until the appropriate `START SLAVE` statement has been issued, as explained in Step 4 below.

3. It is necessary to synchronize the slave server with the master server's replication binary log. If binary logging has not previously been running on the master, run the following statement on the slave:

```
mysqlS> CHANGE MASTER TO
     -> MASTER_LOG_FILE='',
     -> MASTER_LOG_POS=4;
```

This instructs the slave to begin reading the master's binary log from the log's starting point. Otherwise—that is, if you are loading data from the master using a backup—see Section 8.8, "Implementing Failover with MySQL Cluster Replication", for information on how to obtain the correct values to use for `MASTER_LOG_FILE` and `MASTER_LOG_POS` in such cases.

4. Finally, you must instruct the slave to begin applying replication by issuing this command from the `mysql` client on the replication slave:

```
mysqlS> START SLAVE;
```

This also initiates the transmission of replication data from the master to the slave.

It is also possible to use two replication channels, in a manner similar to the procedure described in the next section; the differences between this and using a single replication channel are covered in Section 8.7, "Using Two Replication Channels for MySQL Cluster Replication".

It is also possible to improve cluster replication performance by enabling *batched updates*. This can be accomplished by setting the `slave_allow_batching` system variable on the slave `mysqld` processes. Normally, updates are applied as soon as they are received. However, the use of batching causes updates to be applied in 32 KB batches, which can result in higher throughput and less CPU usage, particularly where individual updates are relatively small.

> **Note**
>
> Slave batching works on a per-epoch basis; updates belonging to more than one transaction can be sent as part of the same batch.
>
> All outstanding updates are applied when the end of an epoch is reached, even if the updates total less than 32 KB.

Batching can be turned on and off at runtime. To activate it at runtime, you can use either of these two statements:

```
SET GLOBAL slave_allow_batching = 1;
SET GLOBAL slave_allow_batching = ON;
```

If a particular batch causes problems (such as a statement whose effects do not appear to be replicated correctly), slave batching can be deactivated using either of the following statements:

```
SET GLOBAL slave_allow_batching = 0;
SET GLOBAL slave_allow_batching = OFF;
```

You can check whether slave batching is currently being used by means of an appropriate `SHOW VARIABLES` statement, like this one:

```
mysql> SHOW VARIABLES LIKE 'slave%';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| slave_allow_batching      | ON    |
| slave_compressed_protocol | OFF   |
| slave_load_tmpdir         | /tmp  |
| slave_net_timeout         | 3600  |
| slave_skip_errors         | OFF   |
| slave_transaction_retries | 10    |
+---------------------------+-------+
6 rows in set (0.00 sec)
```

8.7 Using Two Replication Channels for MySQL Cluster Replication

In a more complete example scenario, we envision two replication channels to provide redundancy and thereby guard against possible failure of a single replication channel. This requires a total of four replication servers, two masters for the master cluster and two slave servers for the slave cluster. For purposes of the discussion that follows, we assume that unique identifiers are assigned as shown here:

Server ID	Description
1	Master - primary replication channel (*M*)
2	Master - secondary replication channel (*M'*)
3	Slave - primary replication channel (*S*)
4	Slave - secondary replication channel (*S'*)

Setting up replication with two channels is not radically different from setting up a single replication channel. First, the `mysqld` processes for the primary and secondary replication masters must be started, followed by those for the primary and secondary slaves. Then the replication processes may be initiated by issuing the `START SLAVE` statement on each of the slaves. The commands and the order in which they need to be issued are shown here:

1. Start the primary replication master:

```
shellM> mysqld --ndbcluster --server-id=1 \
               --log-bin &
```

2. Start the secondary replication master:

```
shellM'> mysqld --ndbcluster --server-id=2 \
               --log-bin &
```

3. Start the primary replication slave server:

```
shellS> mysqld --ndbcluster --server-id=3 \
               --skip-slave-start &
```

4. Start the secondary replication slave:

```
shellS'> mysqld --ndbcluster --server-id=4 \
                --skip-slave-start &
```

5. Finally, initiate replication on the primary channel by executing the `START SLAVE` statement on the primary slave as shown here:

```
mysqlS> START SLAVE;
```

> **Warning**
>
> Only the primary channel is to be started at this point. The secondary replication channel is to be started only in the event that the primary replication channel fails, as described in Section 8.8, "Implementing Failover with MySQL Cluster Replication". Running multiple replication channels simultaneously can result in unwanted duplicate records being created on the replication slaves.

As mentioned previously, it is not necessary to enable binary logging on replication slaves.

8.8 Implementing Failover with MySQL Cluster Replication

In the event that the primary Cluster replication process fails, it is possible to switch over to the secondary replication channel. The following procedure describes the steps required to accomplish this.

1. Obtain the time of the most recent global checkpoint (GCP). That is, you need to determine the most recent epoch from the `ndb_apply_status` table on the slave cluster, which can be found using the following query:

```
mysqlS'> SELECT @latest:=MAX(epoch)
      ->        FROM mysql.ndb_apply_status;
```

   In a circular replication topology, with a master and a slave running on each host, when you are using `ndb_log_apply_status=1`, MySQL Cluster epochs are written in the slave binary logs. This means that the `ndb_apply_status` table contains information for the slave on this host as well as for any other host which acts as a slave of the master running on this host.

   In this case, you need to determine the latest epoch on this slave to the exclusion of any epochs from any other slaves in this slave's binary log that were not listed in the `IGNORE_SERVER_IDS` options of the `CHANGE MASTER TO` statement used to set up this slave. The reason for excluding such epochs is that rows in the `mysql.ndb_apply_status` table whose server IDs have a match in the `IGNORE_SERVER_IDS` list used with the CHANGE MASTER TO statement used to prepare this slave's master are also considered to be from local servers, in addition to those having the slave's own server ID. You can retrieve this list as `Replicate_Ignore_Server_Ids` from the output of `SHOW SLAVE STATUS`. We assume that you have obtained this list and are substituting it for *`ignore_server_ids`* in the query shown here, which like the previous version of the query, selects the greatest epoch into a variable named `@latest`:

```
mysqlS'> SELECT @latest:=MAX(epoch)
      ->        FROM mysql.ndb_apply_status
      ->        WHERE server_id NOT IN (ignore_server_ids);
```

In some cases, it may be simpler or more efficient (or both) to use a list of the server IDs to be included and `server_id IN` *`server_id_list`* in the `WHERE` condition of the preceding query.

2. Using the information obtained from the query shown in Step 1, obtain the corresponding records from the `ndb_binlog_index` table on the master cluster.

   In MySQL Cluster NDB 7.3 and later, you can use the following query to obtain the needed records from the master's `ndb_binlog_index` table:

```
mysqlM'> SELECT
      ->     @file:=SUBSTRING_INDEX(next_file, '/', -1),
      ->     @pos:=next_position
      -> FROM mysql.ndb_binlog_index
      -> WHERE epoch = @latest
      -> ORDER BY epoch ASC LIMIT 1;
```

   These are the records saved on the master since the failure of the primary replication channel. We have employed a user variable `@latest` here to represent the value obtained in Step 1. Of course, it is not possible for one `mysqld` instance to access user variables set on another server instance directly. These values must be “plugged in” to the second query manually or in application code.

   Important

   You must ensure that the slave `mysqld` is started with `--slave-skip-errors=ddl_exist_errors` before executing `START SLAVE`. Otherwise, replication may stop with duplicate DDL errors.

3. Now it is possible to synchronize the secondary channel by running the following query on the secondary slave server:

```
mysqlS'> CHANGE MASTER TO
      ->     MASTER_LOG_FILE='@file',
      ->     MASTER_LOG_POS=@pos;
```

   Again we have employed user variables (in this case `@file` and `@pos`) to represent the values obtained in Step 2 and applied in Step 3; in practice these values must be inserted manually or using application code that can access both of the servers involved.

   Note

   `@file` is a string value such as `'/var/log/mysql/replication-master-bin.00001'`, and so must be quoted when used in SQL or application code. However, the value represented by `@pos` must *not* be quoted. Although MySQL normally attempts to convert strings to numbers, this case is an exception.

4. You can now initiate replication on the secondary channel by issuing the appropriate command on the secondary slave `mysqld`:

```
mysqlS'> START SLAVE;
```

Once the secondary replication channel is active, you can investigate the failure of the primary and effect repairs. The precise actions required to do this will depend upon the reasons for which the primary channel failed.

Warning

The secondary replication channel is to be started only if and when the primary replication channel has failed. Running multiple replication channels simultaneously can result in unwanted duplicate records being created on the replication slaves.

If the failure is limited to a single server, it should (in theory) be possible to replicate from *M* to *S'*, or from *M'* to *S*; however, this has not yet been tested.

8.9 MySQL Cluster Backups With MySQL Cluster Replication

This section discusses making backups and restoring from them using MySQL Cluster replication. We assume that the replication servers have already been configured as covered previously (see Section 8.5, “Preparing the MySQL Cluster for Replication”, and the sections immediately following). This having been done, the procedure for making a backup and then restoring from it is as follows:

1. There are two different methods by which the backup may be started.

   - **Method A.** This method requires that the cluster backup process was previously enabled on the master server, prior to starting the replication process. This can be done by including the following line in a `[mysql_cluster]` section in the `my.cnf file`, where *`management_host`* is the IP address or host name of the `NDB` management server for the master cluster, and *`port`* is the management server's port number:

     ```
     ndb-connectstring=management_host[:port]
     ```

     Note

     The port number needs to be specified only if the default port (1186) is not being used. See Section 4.4, “Initial Configuration of MySQL Cluster”, for more information about ports and port allocation in MySQL Cluster.

     In this case, the backup can be started by executing this statement on the replication master:

     ```
     shellM> ndb_mgm -e "START BACKUP"
     ```

   - **Method B.** If the `my.cnf` file does not specify where to find the management host, you can start the backup process by passing this information to the `NDB` management client as part of the `START BACKUP` command. This can be done as shown here, where *`management_host`* and *`port`* are the host name and port number of the management server:

     ```
     shellM> ndb_mgm management_host:port -e "START BACKUP"
     ```

     In our scenario as outlined earlier (see Section 8.5, “Preparing the MySQL Cluster for Replication”), this would be executed as follows:

     ```
     shellM> ndb_mgm rep-master:1186 -e "START BACKUP"
     ```

2. Copy the cluster backup files to the slave that is being brought on line. Each system running an `ndbd` process for the master cluster will have cluster backup files located on it, and *all* of these files must be copied to the slave to ensure a successful restore. The backup files can be copied into any directory on the computer where the slave management host resides, so long as the MySQL and

NDB binaries have read permissions in that directory. In this case, we will assume that these files have been copied into the directory `/var/BACKUPS/BACKUP-1`.

It is not necessary that the slave cluster have the same number of `ndbd` processes (data nodes) as the master; however, it is highly recommended this number be the same. It *is* necessary that the slave be started with the `--skip-slave-start` option, to prevent premature startup of the replication process.

3. Create any databases on the slave cluster that are present on the master cluster that are to be replicated to the slave.

> **Important**
>
> A `CREATE DATABASE` (or `CREATE SCHEMA`) statement corresponding to each database to be replicated must be executed on each SQL node in the slave cluster.

4. Reset the slave cluster using this statement in the MySQL Monitor:

```
mysqlS> RESET SLAVE;
```

5. You can now start the cluster restoration process on the replication slave using the `ndb_restore` command for each backup file in turn. For the first of these, it is necessary to include the `-m` option to restore the cluster metadata:

```
shellS> ndb_restore -c slave_host:port -n node-id \
        -b backup-id -m -r dir
```

`dir` is the path to the directory where the backup files have been placed on the replication slave. For the `ndb_restore` commands corresponding to the remaining backup files, the `-m` option should *not* be used.

For restoring from a master cluster with four data nodes (as shown in the figure in Chapter 8, *MySQL Cluster Replication*) where the backup files have been copied to the directory `/var/BACKUPS/BACKUP-1`, the proper sequence of commands to be executed on the slave might look like this:

```
shellS> ndb_restore -c rep-slave:1186 -n 2 -b 1 -m \
        -r ./var/BACKUPS/BACKUP-1
shellS> ndb_restore -c rep-slave:1186 -n 3 -b 1 \
        -r ./var/BACKUPS/BACKUP-1
shellS> ndb_restore -c rep-slave:1186 -n 4 -b 1 \
        -r ./var/BACKUPS/BACKUP-1
shellS> ndb_restore -c rep-slave:1186 -n 5 -b 1 -e \
        -r ./var/BACKUPS/BACKUP-1
```

> **Important**
>
> The `-e` (or `--restore_epoch` [307]) option in the final invocation of `ndb_restore` in this example is required in order that the epoch is written to the slave `mysql.ndb_apply_status`. Without this information, the slave will not be able to synchronize properly with the master. (See Section 6.20, "`ndb_restore` — Restore a MySQL Cluster Backup".)

6. Now you need to obtain the most recent epoch from the `ndb_apply_status` table on the slave (as discussed in Section 8.8, "Implementing Failover with MySQL Cluster Replication"):

```
mysqlS> SELECT @latest:=MAX(epoch)
        FROM mysql.ndb_apply_status;
```

7. Using `@latest` as the epoch value obtained in the previous step, you can obtain the correct starting position `@pos` in the correct binary log file `@file` from the master's `mysql.ndb_binlog_index` table using the query shown here:

   ```
   mysqlM> SELECT
        ->     @file:=SUBSTRING_INDEX(File, '/', -1),
        ->     @pos:=Position
        -> FROM mysql.ndb_binlog_index
        -> WHERE epoch > @latest
        -> ORDER BY epoch ASC LIMIT 1;
   ```

   In the event that there is currently no replication traffic, you can get this information by running `SHOW MASTER STATUS` on the master and using the value in the `Position` column for the file whose name has the suffix with the greatest value for all files shown in the `File` column. However, in this case, you must determine this and supply it in the next step manually or by parsing the output with a script.

8. Using the values obtained in the previous step, you can now issue the appropriate `CHANGE MASTER TO` statement in the slave's `mysql` client:

   ```
   mysqlS> CHANGE MASTER TO
        ->     MASTER_LOG_FILE='@file',
        ->     MASTER_LOG_POS=@pos;
   ```

9. Now that the slave "knows" from what point in which binary log file to start reading data from the master, you can cause the slave to begin replicating with this standard MySQL statement:

   ```
   mysqlS> START SLAVE;
   ```

To perform a backup and restore on a second replication channel, it is necessary only to repeat these steps, substituting the host names and IDs of the secondary master and slave for those of the primary master and slave replication servers where appropriate, and running the preceding statements on them.

For additional information on performing Cluster backups and restoring Cluster from backups, see Section 7.3, "Online Backup of MySQL Cluster".

8.9.1 MySQL Cluster Replication: Automating Synchronization of the Replication Slave to the Master Binary Log

It is possible to automate much of the process described in the previous section (see Section 8.9, "MySQL Cluster Backups With MySQL Cluster Replication"). The following Perl script `reset-slave.pl` serves as an example of how you can do this.

```
#!/user/bin/perl -w
#  file: reset-slave.pl
#  Copyright ©2005 MySQL AB
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to:
#  Free Software Foundation, Inc.
#  59 Temple Place, Suite 330
#  Boston, MA 02111-1307 USA
#
#  Version 1.1
######################## Includes ###############################
```

```
use DBI;
######################## Globals ################################
my  $m_host='';
my  $m_port='';
my  $m_user='';
my  $m_pass='';
my  $s_host='';
my  $s_port='';
my  $s_user='';
my  $s_pass='';
my  $dbhM='';
my  $dbhS='';
####################### Sub Prototypes ##########################
sub CollectCommandPromptInfo;
sub ConnectToDatabases;
sub DisconnectFromDatabases;
sub GetSlaveEpoch;
sub GetMasterInfo;
sub UpdateSlave;
######################## Program Main ###########################
CollectCommandPromptInfo;
ConnectToDatabases;
GetSlaveEpoch;
GetMasterInfo;
UpdateSlave;
DisconnectFromDatabases;
################## Collect Command Prompt Info ##################
sub CollectCommandPromptInfo
{
  ### Check that user has supplied correct number of command line args
  die "Usage:\n
       reset-slave >master MySQL host< >master MySQL port< \n
                   >master user< >master pass< >slave MySQL host< \n
                   >slave MySQL port< >slave user< >slave pass< \n
       All 8 arguments must be passed. Use BLANK for NULL passwords\n"
       unless @ARGV == 8;
  $m_host  =  $ARGV[0];
  $m_port  =  $ARGV[1];
  $m_user  =  $ARGV[2];
  $m_pass  =  $ARGV[3];
  $s_host  =  $ARGV[4];
  $s_port  =  $ARGV[5];
  $s_user  =  $ARGV[6];
  $s_pass  =  $ARGV[7];
  if ($m_pass eq "BLANK") { $m_pass = '';}
  if ($s_pass eq "BLANK") { $s_pass = '';}
}
###############  Make connections to both databases #############
sub ConnectToDatabases
{
  ### Connect to both master and slave cluster databases
  ### Connect to master
  $dbhM
    = DBI->connect(
    "dbi:mysql:database=mysql;host=$m_host;port=$m_port",
    "$m_user", "$m_pass")
      or die "Can't connect to Master Cluster MySQL process!
              Error: $DBI::errstr\n";
  ### Connect to slave
  $dbhS
    = DBI->connect(
          "dbi:mysql:database=mysql;host=$s_host",
          "$s_user", "$s_pass")
    or die "Can't connect to Slave Cluster MySQL process!
            Error: $DBI::errstr\n";
}
################  Disconnect from both databases ################
sub DisconnectFromDatabases
{
  ### Disconnect from master
  $dbhM->disconnect
  or warn " Disconnection failed: $DBI::errstr\n";
```

```
  ### Disconnect from slave
  $dbhS->disconnect
  or warn " Disconnection failed: $DBI::errstr\n";
}
######################  Find the last good GCI ##################
sub GetSlaveEpoch
{
  $sth = $dbhS->prepare("SELECT MAX(epoch)
                         FROM mysql.ndb_apply_status;")
      or die "Error while preparing to select epoch from slave: ",
             $dbhS->errstr;
  $sth->execute
      or die "Selecting epoch from slave error: ", $sth->errstr;
  $sth->bind_col (1, \$epoch);
  $sth->fetch;
  print "\tSlave Epoch =  $epoch\n";
  $sth->finish;
}
#######  Find the position of the last GCI in the binary log ########
sub GetMasterInfo
{
  $sth = $dbhM->prepare("SELECT
                           SUBSTRING_INDEX(File, '/', -1), Position
                         FROM mysql.ndb_binlog_index
                         WHERE epoch > $epoch
                         ORDER BY epoch ASC LIMIT 1;")
      or die "Prepare to select from master error: ", $dbhM->errstr;
  $sth->execute
      or die "Selecting from master error: ", $sth->errstr;
  $sth->bind_col (1, \$binlog);
  $sth->bind_col (2, \$binpos);
  $sth->fetch;
  print "\tMaster binary log =  $binlog\n";
  print "\tMaster binary log position =  $binpos\n";
  $sth->finish;
}
##########  Set the slave to process from that location #########
sub UpdateSlave
{
  $sth = $dbhS->prepare("CHANGE MASTER TO
                         MASTER_LOG_FILE='$binlog',
                         MASTER_LOG_POS=$binpos;")
      or die "Prepare to CHANGE MASTER error: ", $dbhS->errstr;
  $sth->execute
       or die "CHANGE MASTER on slave error: ", $sth->errstr;
  $sth->finish;
  print "\tSlave has been updated. You may now start the slave.\n";
}
# end reset-slave.pl
```

8.9.2 Point-In-Time Recovery Using MySQL Cluster Replication

Point-in-time recovery—that is, recovery of data changes made since a given point in time—is performed after restoring a full backup that returns the server to its state when the backup was made. Performing point-in-time recovery of MySQL Cluster tables with MySQL Cluster and MySQL Cluster Replication can be accomplished using a native `NDB` data backup (taken by issuing `CREATE BACKUP` in the `ndb_mgm` client) and restoring the `ndb_binlog_index` table (from a dump made using `mysqldump`).

To perform point-in-time recovery of MySQL Cluster, it is necessary to follow the steps shown here:

1. Back up all `NDB` databases in the cluster, using the `START BACKUP` command in the `ndb_mgm` client (see Section 7.3, “Online Backup of MySQL Cluster”).

2. At some later point, prior to restoring the cluster, make a backup of the `mysql.ndb_binlog_index` table. It is probably simplest to use `mysqldump` for this task. Also back up the binary log files at this time.

   This backup should be updated regularly—perhaps even hourly—depending on your needs.

3. (*Catastrophic failure or error occurs*.)

4. Locate the last known good backup.

5. Clear the data node file systems (using `ndbd --initial` or `ndbmtd --initial`).

   Note

   MySQL Cluster Disk Data tablespace and log files are not removed by `--initial`. You must delete these manually.

6. Use `DROP TABLE` or `TRUNCATE TABLE` with the `mysql.ndb_binlog_index` table.

7. Execute `ndb_restore`, restoring all data. You must include the `--restore_epoch` [307] option when you run `ndb_restore`, so that the `ndb_apply_status` table is populated correctly. (See Section 6.20, "`ndb_restore` — Restore a MySQL Cluster Backup", for more information.)

8. Restore the `ndb_binlog_index` table from the output of `mysqldump` and restore the binary log files from backup, if necessary.

9. Find the epoch applied most recently—that is, the maximum `epoch` column value in the `ndb_apply_status` table—as the user variable `@LATEST_EPOCH` (emphasized):

```
SELECT @LATEST_EPOCH:=MAX(epoch)
    FROM mysql.ndb_apply_status;
```

10. Find the latest binary log file (`@FIRST_FILE`) and position (`Position` column value) within this file that correspond to `@LATEST_EPOCH` in the `ndb_binlog_index` table:

```
SELECT Position, @FIRST_FILE:=File
    FROM mysql.ndb_binlog_index
    WHERE epoch > @LATEST_EPOCH ORDER BY epoch ASC LIMIT 1;
```

11. Using `mysqlbinlog`, replay the binary log events from the given file and position up to the point of the failure. (See `mysqlbinlog` — Utility for Processing Binary Log Files.)

See also Point-in-Time (Incremental) Recovery Using the Binary Log, for more information about the binary log, replication, and incremental recovery.

8.10 MySQL Cluster Replication: Multi-Master and Circular Replication

It is possible to use MySQL Cluster in multi-master replication, including circular replication between a number of MySQL Clusters.

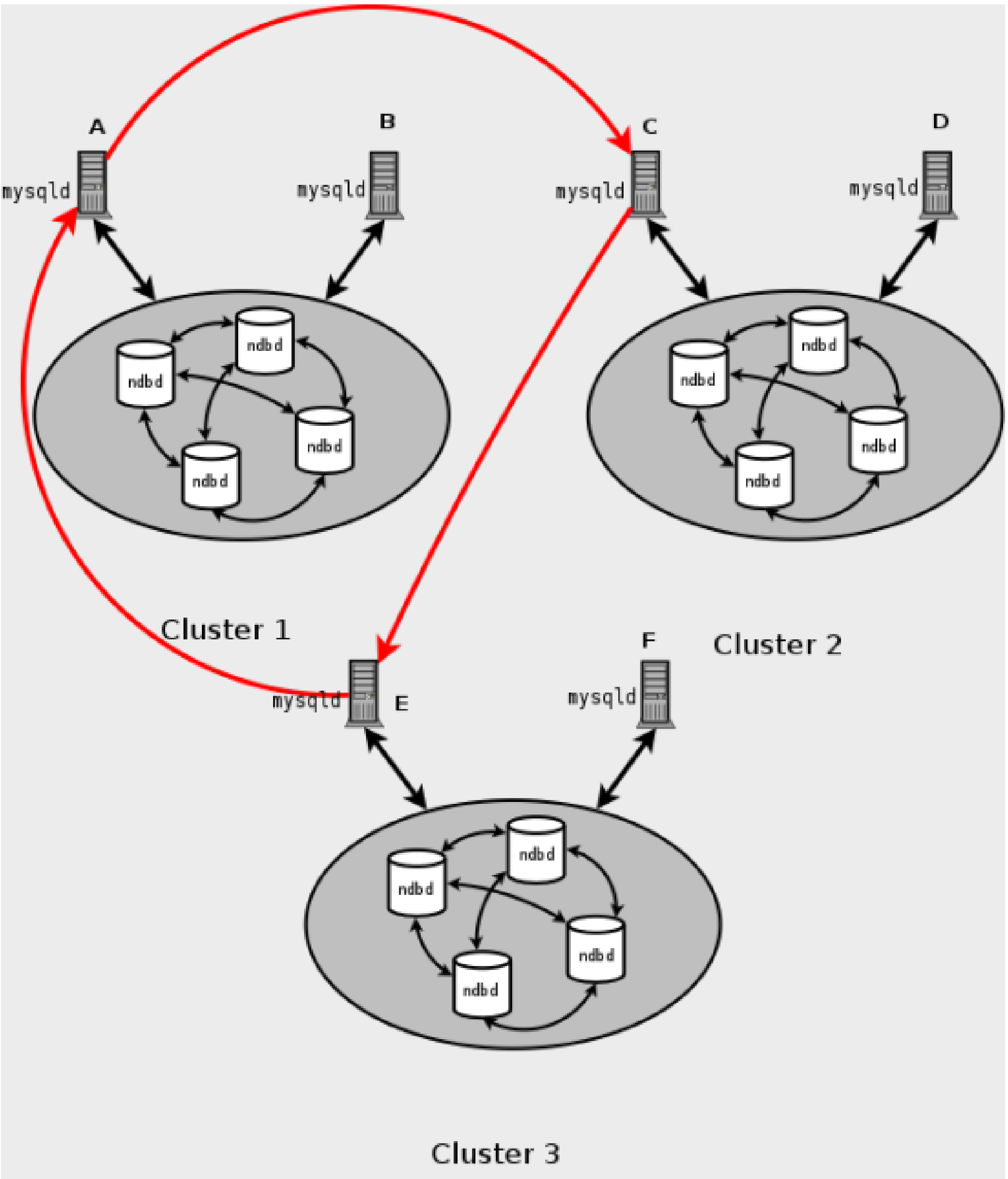
Circular replication example. In the next few paragraphs we consider the example of a replication setup involving three MySQL Clusters numbered 1, 2, and 3, in which Cluster 1 acts as the replication master for Cluster 2, Cluster 2 acts as the master for Cluster 3, and Cluster 3 acts as the master for Cluster 1. Each cluster has two SQL nodes, with SQL nodes A and B belonging to Cluster 1, SQL nodes C and D belonging to Cluster 2, and SQL nodes E and F belonging to Cluster 3.

Circular replication using these clusters is supported as long as the following conditions are met:

- The SQL nodes on all masters and slaves are the same
- All SQL nodes acting as replication masters and slaves are started using the `--log-slave-updates` option

This type of circular replication setup is shown in the following diagram:

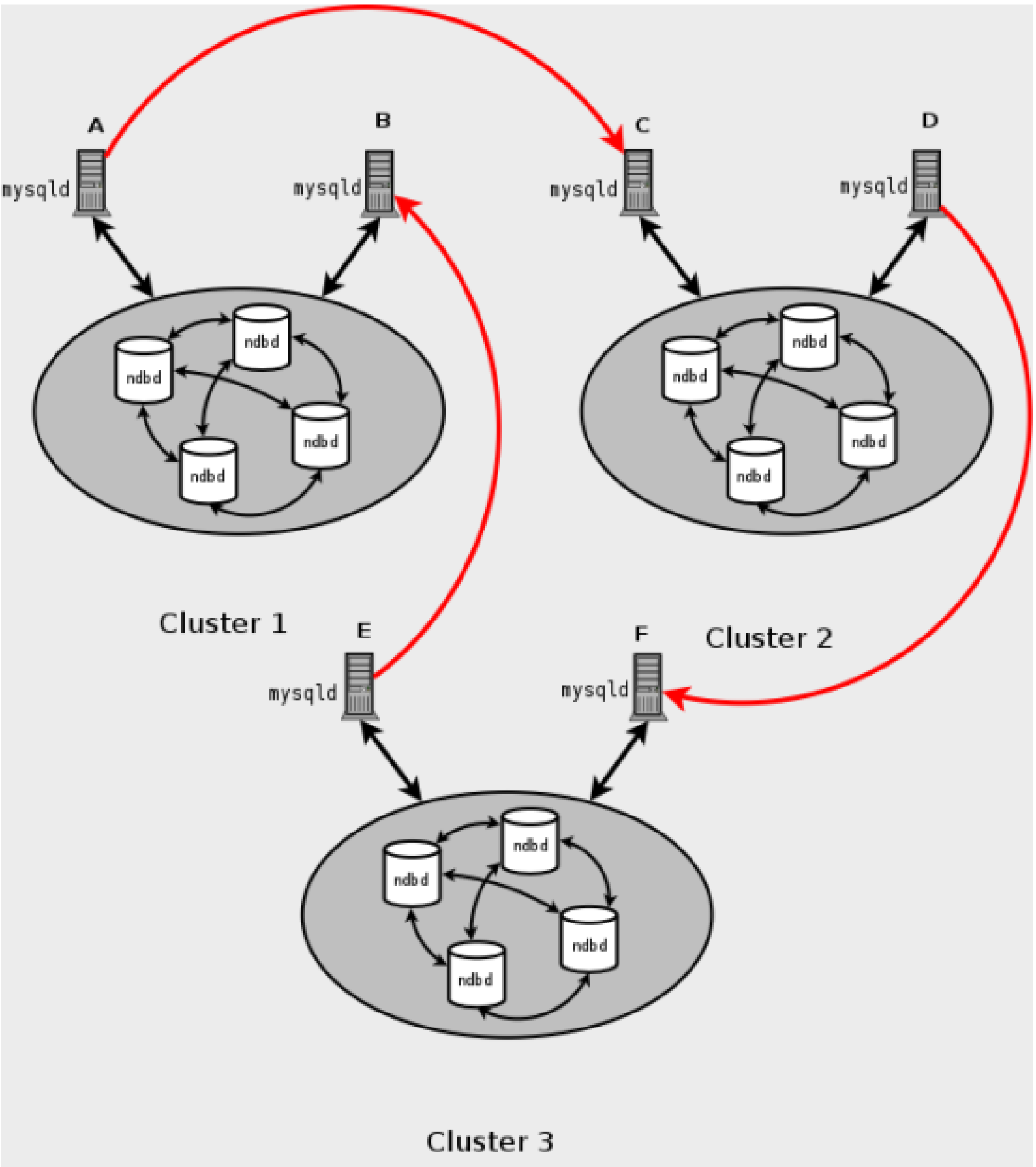
Figure 8.6 MySQL Cluster Circular Replication with All Masters As Slaves



In this scenario, SQL node A in Cluster 1 replicates to SQL node C in Cluster 2; SQL node C replicates to SQL node E in Cluster 3; SQL node E replicates to SQL node A. In other words, the replication line (indicated by the red arrows in the diagram) directly connects all SQL nodes used as replication masters and slaves.

It is also possible to set up circular replication in such a way that not all master SQL nodes are also slaves, as shown here:

Figure 8.7 MySQL Cluster Circular Replication Where Not All Masters Are Slaves



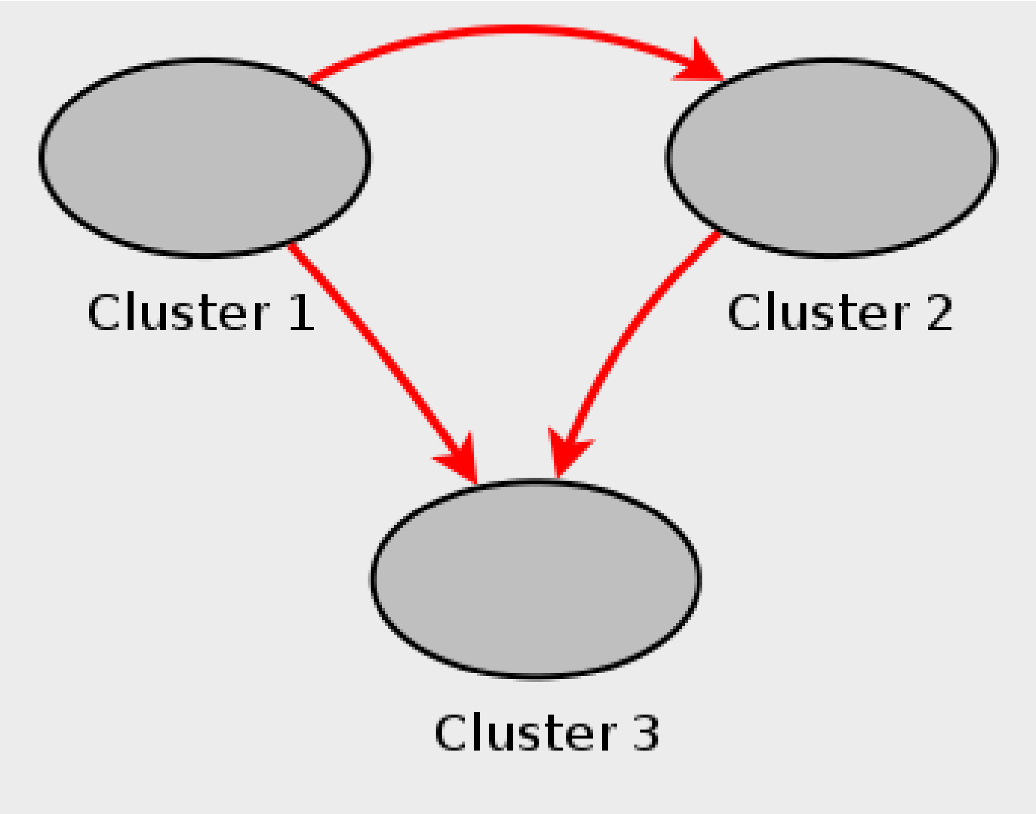
In this case, different SQL nodes in each cluster are used as replication masters and slaves. However, you must *not* start any of the SQL nodes using `--log-slave-updates`. This type of circular replication scheme for MySQL Cluster, in which the line of replication (again indicated by the red arrows in the diagram) is discontinuous, should be possible, but it should be noted that it has not yet been thoroughly tested and must therefore still be considered experimental.

Using NDB-native backup and restore to initialize a slave MySQL Cluster. When setting up circular replication, it is possible to initialize the slave cluster by using the management client `BACKUP` command on one MySQL Cluster to create a backup and then applying this backup on another MySQL Cluster using `ndb_restore`. However, this does not automatically create binary logs on the second MySQL Cluster's SQL node acting as the replication slave. In order to cause the binary logs to be created, you must issue a `SHOW TABLES` statement on that SQL node; this should be done prior to running `START SLAVE`.

This is a known issue which we intend to address in a future release.

Multi-master failover example. In this section, we discuss failover in a multi-master MySQL Cluster replication setup with three MySQL Clusters having server IDs 1, 2, and 3. In this scenario, Cluster 1 replicates to Clusters 2 and 3; Cluster 2 also replicates to Cluster 3. This relationship is shown here:

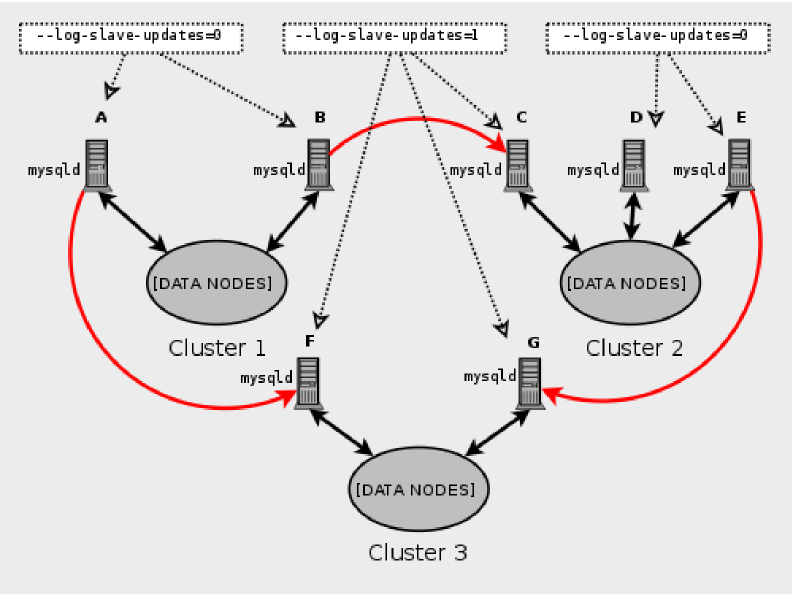
Figure 8.8 MySQL Cluster Multi-Master Replication With 3 Masters



In other words, data replicates from Cluster 1 to Cluster 3 through 2 different routes: directly, and by way of Cluster 2.

Not all MySQL servers taking part in multi-master replication must act as both master and slave, and a given MySQL Cluster might use different SQL nodes for different replication channels. Such a case is shown here:

Figure 8.9 MySQL Cluster Multi-Master Replication, With MySQL Servers



MySQL servers acting as replication slaves must be run with the `--log-slave-updates` option. Which `mysqld` processes require this option is also shown in the preceding diagram.

> **Note**
>
> Using the `--log-slave-updates` option has no effect on servers not being run as replication slaves.

The need for failover arises when one of the replicating clusters goes down. In this example, we consider the case where Cluster 1 is lost to service, and so Cluster 3 loses 2 sources of updates from Cluster 1. Because replication between MySQL Clusters is asynchronous, there is no guarantee that Cluster 3's updates originating directly from Cluster 1 are more recent than those received through Cluster 2. You can handle this by ensuring that Cluster 3 catches up to Cluster 2 with regard to updates from Cluster 1. In terms of MySQL servers, this means that you need to replicate any outstanding updates from MySQL server C to server F.

On server C, perform the following queries:

```
mysqlC> SELECT @latest:=MAX(epoch)
     ->     FROM mysql.ndb_apply_status
     ->     WHERE server_id=1;
mysqlC> SELECT
     ->     @file:=SUBSTRING_INDEX(File, '/', -1),
     ->     @pos:=Position
     ->     FROM mysql.ndb_binlog_index
     ->     WHERE orig_epoch >= @latest
     ->     AND orig_server_id = 1
     ->     ORDER BY epoch ASC LIMIT 1;
```

> **Note**
>
> You can improve the performance of this query, and thus likely speed up failover times significantly, by adding the appropriate index to the `ndb_binlog_index` table. See Section 8.4, "MySQL Cluster Replication Schema and Tables", for more information.

Copy over the values for *`@file`* and *`@pos`* manually from server C to server F (or have your application perform the equivalent). Then, on server F, execute the following `CHANGE MASTER TO` statement:

```
mysqlF> CHANGE MASTER TO
     ->     MASTER_HOST = 'serverC'
     ->     MASTER_LOG_FILE='@file',
     ->     MASTER_LOG_POS=@pos;
```

Once this has been done, you can issue a `START SLAVE` statement on MySQL server F, and any missing updates originating from server B will be replicated to server F.

The `CHANGE MASTER TO` statement also supports an `IGNORE_SERVER_IDS` option which takes a comma-separated list of server IDs and causes events originating from the corresponding servers to be ignored. For more information, see CHANGE MASTER TO Syntax, and SHOW SLAVE STATUS Syntax. For information about how this option intereacts with the `ndb_log_apply_status` variable, see Section 8.8, "Implementing Failover with MySQL Cluster Replication".

8.11 MySQL Cluster Replication Conflict Resolution

When using a replication setup involving multiple masters (including circular replication), it is possible that different masters may try to update the same row on the slave with different data. Conflict resolution in MySQL Cluster Replication provides a means of resolving such conflicts by permitting a user-defined resolution column to be used to determine whether or not an update on a given master should be applied on the slave.

Some types of conflict resolution supported by MySQL Cluster (`NDB$OLD()`, `NDB$MAX()`, `NDB$MAX_DELETE_WIN()`) implement this user-defined column as a "timestamp" column (although its type cannot be `TIMESTAMP`, as explained later in this section). These types of conflict resolution are always applied a row-by-row basis rather than a transactional basis. The epoch-based conflict resolution functions NDB$EPOCH() and NDB$EPOCH_TRANS() compare the order in which epochs are replicated (and thus these functions are transactional). Different methods can be used to compare resolution column values on the slave when conflicts occur, as explained later in this section; the method used can be set on a per-table basis.

You should also keep in mind that it is the application's responsibility to ensure that the resolution column is correctly populated with relevant values, so that the resolution function can make the appropriate choice when determining whether to apply an update.

Requirements. Preparations for conflict resolution must be made on both the master and the slave. These tasks are described in the following list:

- On the master writing the binary logs, you must determine which columns are sent (all columns or only those that have been updated). This is done for the MySQL Server as a whole by applying the mysqld startup option --ndb-log-updated-only [474] (described later in this section) or on a per-table basis by entries in the mysql.ndb_replication table (see The ndb_replication system table).

  > **Note**
  >
  > If you are replicating tables with very large columns (such as TEXT or BLOB columns), --ndb-log-updated-only [474] can also be useful for reducing the size of the master and slave binary logs and avoiding possible replication failures due to exceeding max_allowed_packet.
  >
  > See Replication and max_allowed_packet, for more information about this issue.

- On the slave, you must determine which type of conflict resolution to apply ("latest timestamp wins", "same timestamp wins", "primary wins", "primary wins, complete transaction", or none). This is done using the mysql.ndb_replication system table, on a per-table basis (see The ndb_replication system table).

- MySQL Cluster NDB 7.4.1 and later support read conflict detection, that is, detecting conflicts between reads of a given row in one cluster and updates or deletes of the same row in another cluster. This requires exclusive read locks obtained by setting ndb_log_exclusive_reads equal to 1 on the slave. All rows read by a conflicting read are logged in the exceptions table. For more information, see Read conflict detection and resolution.

When using the functions NDB$OLD(), NDB$MAX(), and NDB$MAX_DELETE_WIN() for timestamp-based conflict resolution, we often refer to the column used for determining updates as a "timestamp" column. However, the data type of this column is never TIMESTAMP; instead, its data type should be INT (INTEGER) or BIGINT. The "timestamp" column should also be UNSIGNED and NOT NULL.

The NDB$EPOCH() and NDB$EPOCH_TRANS() functions discussed later in this section work by comparing the relative order of replication epochs applied on a primary and secondary MySQL Cluster, and do not make use of timestamps.

Master column control. We can see update operations in terms of "before" and "after" images—that is, the states of the table before and after the update is applied. Normally, when updating a table with a primary key, the "before" image is not of great interest; however, when we need to determine on a per-update basis whether or not to use the updated values on a replication slave, we need to make sure that both images are written to the master's binary log. This is done with the --ndb-log-update-as-write [474] option for mysqld, as described later in this section.

> **Important**
>
> Whether logging of complete rows or of updated columns only is done is decided when the MySQL server is started, and cannot be changed online;

you must either restart mysqld, or start a new mysqld instance with different logging options.

Logging Full or Partial Rows (--ndb-log-updated-only Option)

Command-Line Format	--ndb-log-updated-only	
System Variable	**Name**	ndb_log_updated_only
	Variable Scope	Global
	Dynamic Variable	Yes
Permitted Values	**Type**	boolean
	Default	ON

For purposes of conflict resolution, there are two basic methods of logging rows, as determined by the setting of the --ndb-log-updated-only [474] option for mysqld:

- Log complete rows
- Log only column data that has been updated—that is, column data whose value has been set, regardless of whether or not this value was actually changed. This is the default behavior.

It is usually sufficient—and more efficient—to log updated columns only; however, if you need to log full rows, you can do so by setting --ndb-log-updated-only [474] to 0 or OFF.

--ndb-log-update-as-write Option: Logging Changed Data as Updates

Command-Line Format	--ndb-log-update-as-write	
System Variable	**Name**	ndb_log_update_as_write
	Variable Scope	Global
	Dynamic Variable	Yes
Permitted Values	**Type**	boolean
	Default	ON

The setting of the MySQL Server's --ndb-log-update-as-write [474] option determines whether logging is performed with or without the “before” image. Because conflict resolution is done in the MySQL Server's update handler, it is necessary to control logging on the master such that updates are updates and not writes; that is, such that updates are treated as changes in existing rows rather than the writing of new rows (even though these replace existing rows). This option is turned on by default; in other words, updates are treated as writes. (That is, updates are by default written as write_row events in the binary log, rather than as update_row events.)

To turn off the option, start the master mysqld with --ndb-log-update-as-write=0 or --ndb-log-update-as-write=OFF. You must do this when replicating from NDB tables to tables using a different storage engine; see Replication from NDB to other storage engines, and Replication from NDB to a nontransactional storage engine, for more information.

Conflict resolution control. Conflict resolution is usually enabled on the server where conflicts can occur. Like logging method selection, it is enabled by entries in the mysql.ndb_replication table.

The ndb_replication system table. To enable conflict resolution, it is necessary to create an ndb_replication table in the mysql system database on the master, the slave, or both, depending on the conflict resolution type and method to be employed. This table is used to control logging and conflict resolution functions on a per-table basis, and has one row per table involved in replication. ndb_replication is created and filled with control information on the server where the conflict is to be resolved. In a simple master-slave setup where data can also be changed locally on the slave this will typically be the slave. In a more complex master-master (2-way) replication schema this will usually be all of the masters involved. Each row in mysql.ndb_replication corresponds to a table being replicated, and specifies how to log and resolve conflicts (that is, which conflict resolution function, if any, to use) for that table. The definition of the mysql.ndb_replication table is shown here:

```
CREATE TABLE mysql.ndb_replication  (
    db VARBINARY(63),
    table_name VARBINARY(63),
    server_id INT UNSIGNED,
    binlog_type INT UNSIGNED,
    conflict_fn VARBINARY(128),
    PRIMARY KEY USING HASH (db, table_name, server_id)
)   ENGINE=NDB
PARTITION BY KEY(db,table_name);
```

The columns in this table are described in the next few paragraphs.

db. The name of the database containing the table to be replicated. You may employ either or both of the wildcards _ and % as part of the database name. Matching is similar to what is implemented for the LIKE operator.

table_name. The name of the table to be replicated. The table name may include either or both of the wildcards _ and %. Matching is similar to what is implemented for the LIKE operator.

server_id. The unique server ID of the MySQL instance (SQL node) where the table resides.

binlog_type. The type of binary logging to be employed. This is determined as shown in the following table:

Value	Internal Value	Description
0	NBT_DEFAULT	Use server default
1	NBT_NO_LOGGING	Do not log this table in the binary log
2	NBT_UPDATED_ONLY	Only updated attributes are logged
3	NBT_FULL	Log full row, even if not updated (MySQL server default behavior)
4	NBT_USE_UPDATE	(For generating NBT_UPDATED_ONLY_USE_UPDATE and NBT_FULL_USE_UPDATE values only—not intended for separate use)
5	[*Not used*]	---
6	NBT_UPDATED_ONLY_USE_UPDATE (equal to NBT_UPDATED_ONLY | NBT_USE_UPDATE)	Use updated attributes, even if values are unchanged
7	NBT_FULL_USE_UPDATE (equal to NBT_FULL | NBT_USE_UPDATE)	Use full row, even if values are unchanged

conflict_fn. The conflict resolution function to be applied. This function must be specified as one of those shown in the following list:

- NDB$OLD(column_name)
- NDB$MAX(column_name)

- NDB$MAX_DELETE_WIN()
- NDB$EPOCH() and NDB$EPOCH_TRANS()
- NDB$EPOCH_TRANS()
- NDB$EPOCH2() (MySQL Cluster NDB 7.4.2 and later)
- NDB$EPOCH2_TRANS() (MySQL Cluster NDB 7.4.2 and later)
- `NULL`: Indicates that conflict resolution is not to be used for the corresponding table.

These functions are described in the next few paragraphs.

NDB$OLD(column_name). If the value of *`column_name`* is the same on both the master and the slave, then the update is applied; otherwise, the update is not applied on the slave and an exception is written to the log. This is illustrated by the following pseudocode:

```
if (master_old_column_value == slave_current_column_value)
  apply_update();
else
  log_exception();
```

This function can be used for "same value wins" conflict resolution. This type of conflict resolution ensures that updates are not applied on the slave from the wrong master.

> **Important**
>
> The column value from the master's "before" image is used by this function.

NDB$MAX(column_name). If the "timestamp" column value for a given row coming from the master is higher than that on the slave, it is applied; otherwise it is not applied on the slave. This is illustrated by the following pseudocode:

```
if (master_new_column_value > slave_current_column_value)
  apply_update();
```

This function can be used for "greatest timestamp wins" conflict resolution. This type of conflict resolution ensures that, in the event of a conflict, the version of the row that was most recently updated is the version that persists.

> **Important**
>
> The column value from the master's "after" image is used by this function.

NDB$MAX_DELETE_WIN(). This is a variation on `NDB$MAX()`. Due to the fact that no timestamp is available for a delete operation, a delete using `NDB$MAX()` is in fact processed as `NDB$OLD`. However, for some use cases, this is not optimal. For `NDB$MAX_DELETE_WIN()`, if the "timestamp" column value for a given row adding or updating an existing row coming from the master is higher than that on the slave, it is applied. However, delete operations are treated as always having the higher value. This is illustrated in the following pseudocode:

```
if ( (master_new_column_value > slave_current_column_value)
        ||
      operation.type == "delete")
  apply_update();
```

This function can be used for "greatest timestamp, delete wins" conflict resolution. This type of conflict resolution ensures that, in the event of a conflict, the version of the row that was deleted or (otherwise) most recently updated is the version that persists.

> **Note**
>
> As with NDB$MAX(), the column value from the master's "after" image is the value used by this function.

NDB$EPOCH() and NDB$EPOCH_TRANS(). The NDB$EPOCH() function tracks the order in which replicated epochs are applied on a slave MySQL Cluster relative to changes originating on the slave. This relative ordering is used to determine whether changes originating on the slave are concurrent with any changes that originate locally, and are therefore potentially in conflict.

Most of what follows in the description of NDB$EPOCH() also applies to NDB$EPOCH_TRANS(). Any exceptions are noted in the text.

NDB$EPOCH() is asymmetric, operating on one MySQL Cluster in a two-cluster circular replication configuration (sometimes referred to as "active-active" replication). We refer here to cluster on which it operates as the primary, and the other as the secondary. The slave on the primary is responsible for detecting and handling conflicts, while the slave on the secondary is not involved in any conflict detection or handling.

When the slave on the primary detects conflicts, it injects events into its own binary log to compensate for these; this ensures that the secondary MySQL Cluster eventually realigns itself with the primary and so keeps the primary and secondary from diverging. This compensation and realignment mechanism requires that the primary MySQL Cluster always wins any conflicts with the secondary—that is, that the primary's changes are always used rather than those from the secondary in event of a conflict. This "primary always wins" rule has the following implications:

- Operations that change data, once committed on the primary, are fully persistent and will not be undone or rolled back by conflict detection and resolution.
- Data read from the primary is fully consistent. Any changes committed on the Primary (locally or from the slave) will not be reverted later.
- Operations that change data on the secondary may later be reverted if the primary determines that they are in conflict.
- Individual rows read on the secondary are self-consistent at all times, each row always reflecting either a state committed by the secondary, or one committed by the primary.
- Sets of rows read on the secondary may not necessarily be consistent at a given single point in time. For NDB$EPOCH_TRANS(), this is a transient state; for NDB$EPOCH(), it can be a persistent state.
- Assuming a period of sufficient length without any conflicts, all data on the secondary MySQL Cluster (eventually) becomes consistent with the primary's data.

NDB$EPOCH() and NDB$EPOCH_TRANS() do not require any user schema modifications, or application changes to provide conflict detection. However, careful thought must be given to the schema used, and the access patterns used, to verify that the complete system behaves within specified limits.

Each of the NDB$EPOCH() and NDB$EPOCH_TRANS() functions can take an optional parameter; this is the number of bits to use to represent the lower 32 bits of the epoch, and should be set to no less than

```
CEIL( LOG2( TimeBetweenGlobalCheckpoints / TimeBetweenEpochs ), 1)
```

For the default values of these configuration parameters (2000 and 100 milliseconds, respectively), this gives a value of 5 bits, so the default value (6) should be sufficient, unless other values are used for TimeBetweenGlobalCheckpoints, TimeBetweenEpochs, or both. A value that is too small can result in false positives, while one that is too large could lead to excessive wasted space in the database.

Both NDB$EPOCH() and NDB$EPOCH_TRANS() insert entries for conflicting rows into the relevant exceptions tables, provided that these tables have been defined according to the same exceptions table schema rules as described elsewhere in this section (see NDB$OLD(column_name)). You need to create any exceptions table before creating the table with which it is to be used.

As with the other conflict detection functions discussed in this section, NDB$EPOCH() and NDB $EPOCH_TRANS() are activated by including relevant entries in the mysql.ndb_replication table (see The ndb_replication system table). The roles of the primary and secondary MySQL Clusters in this scenario are fully determined by mysql.ndb_replication table entries.

Because the conflict detection algorithms employed by NDB$EPOCH() and NDB$EPOCH_TRANS() are asymmetric, you must use different values for the primary slave's and secondary slave's server_id entries.

Prior to MySQL Cluster NDB 7.3.6, conflicts between DELETE operations were handled like those for UPDATE operations, and within the same epoch were considered in conflict. In MySQL Cluster NDB 7.3.6 and later, a conflict between DELETE operations alone is not sufficient to trigger a conflict using NDB$EPOCH() or NDB$EPOCH_TRANS(), and the relative placement within epochs does not matter. (Bug #18454499)

Conflict detection status variables. Several status variables can be used to monitor conflict detection. You can see how many rows have been found in conflict by NDB$EPOCH() since this slave was last restarted from the current value of the Ndb_conflict_fn_epoch system status variable.

Ndb_conflict_fn_epoch_trans provides the number of rows that have been found directly in conflict by NDB$EPOCH_TRANS(). Ndb_conflict_fn_epoch2 and Ndb_conflict_fn_epoch2_trans, added in MySQL Cluster NDB 7.4.2, show the number of rows found in conflict by NDB$EPOCH2() and NDB$EPOCH2_TRANS(), respectively. The number of rows actually realigned, including those affected due to their membership in or dependency on the same transactions as other conflicting rows, is given by Ndb_conflict_trans_row_reject_count.

For more information, see Section 5.3.8.3, “MySQL Cluster Status Variables”.

Limitations on NDB$EPOCH(). The following limitations currently apply when using NDB $EPOCH() to perform conflict detection:

- Conflicts are detected using MySQL Cluster epoch boundaries, with granularity proportional to TimeBetweenEpochs (default: 100 milliseconds). The minimum conflict window is the minimum time during which concurrent updates to the same data on both clusters always report a conflict. This is always a nonzero length of time, and is roughly proportional to 2 * (latency + queueing + TimeBetweenEpochs). This implies that—assuming the default for TimeBetweenEpochs and ignoring any latency between clusters (as well as any queuing delays)—the minimum conflict window size is approximately 200 milliseconds. This minimum window should be considered when looking at expected application “race” patterns.

- Additional storage is required for tables using the NDB$EPOCH() and NDB$EPOCH_TRANS() functions; from 1 to 32 bits extra space per row is required, depending on the value passed to the function.

- Conflicts between delete operations may result in divergence between the primary and secondary. When a row is deleted on both clusters concurrently, the conflict can be detected, but is not recorded, since the row is deleted. This means that further conflicts during the propagation of any subsequent realignment operations will not be detected, which can lead to divergence.

  Deletes should be externally serialized, or routed to one cluster only. Alternatively, a separate row should be updated transactionally with such deletes and any inserts that follow them, so that conflicts can be tracked across row deletes. This may require changes in applications.

- Only two MySQL Clusters in a circular “active-active” configuration are currently supported when using NDB$EPOCH() or NDB$EPOCH_TRANS() for conflict detection.

- Tables having BLOB or TEXT columns are not currently supported with NDB$EPOCH() or NDB $EPOCH_TRANS().

NDB$EPOCH_TRANS(). NDB$EPOCH_TRANS() extends the NDB$EPOCH() function. Conflicts are detected and handled in the same way using the “primary wins all” rule (see NDB$EPOCH() and NDB$EPOCH_TRANS()) but with the extra condition that any other rows updated in the same transaction in which the conflict occurred are also regarded as being in conflict. In other words, where NDB$EPOCH() realigns individual conflicting rows on the secondary, NDB$EPOCH_TRANS() realigns conflicting transactions.

In addition, any transactions which are detectably dependent on a conflicting transaction are also regarded as being in conflict, these dependencies being determined by the contents of the secondary cluster's binary log. Since the binary log contains only data modification operations (inserts, updates, and deletes), only overlapping data modifications are used to determine dependencies between transactions.

NDB$EPOCH_TRANS() is subject to the same conditions and limitations as NDB$EPOCH(), and in addition requires that Version 2 binary log row events are used (--log-bin-use-v1-row-events equal to 0), which adds a storage overhead of 2 bytes per event in the binary log. In addition, all transaction IDs must be recorded in the secondary's binary log (--ndb-log-transaction-id option), which adds a further variable overhead (up to 13 bytes per row).

See NDB$EPOCH() and NDB$EPOCH_TRANS().

Status information. A server status variable Ndb_conflict_fn_max provides a count of the number of times that a row was not applied on the current SQL node due to “greatest timestamp wins” conflict resolution since the last time that mysqld was started.

The number of times that a row was not applied as the result of “same timestamp wins” conflict resolution on a given mysqld since the last time it was restarted is given by the global status variable Ndb_conflict_fn_old. In addition to incrementing Ndb_conflict_fn_old, the primary key of the row that was not used is inserted into an *exceptions table*, as explained later in this section.

NDB$EPOCH2(). The NDB$EPOCH2() function, added in MySQL Cluster NDB 7.4.2, is similar to NDB$EPOCH(), except that NDB$EPOCH2() provides for delete-delete handling with a circular replication (“master-master”) topology. In this scenario, primary and secondary roles are assigned to the two masters by setting the ndb_slave_conflict_role system variable to the appropriate value on each master (usually one each of PRIMARY, SECONDARY). When this is done, modifications made by the secondary are reflected by the primary back to the secondary which then conditionally applies them.

NDB$EPOCH2_TRANS(). In MySQL Cluster NDB 7.4.2 and later, NDB$EPOCH2_TRANS() extends the NDB$EPOCH2() function. Conflicts are detected and handled in the same way, and assigning primary and secondary roles to the replicating clusters, but with the extra condition that any other rows updated in the same transaction in which the conflict occurred are also regarded as being in conflict. That is, NDB$EPOCH2() realigns individual conflicting rows on the secondary, while NDB $EPOCH_TRANS() realigns conflicting transactions.

Where NDB$EPOCH() and NDB$EPOCH_TRANS() use metadata that is specified per row, per last modified epoch, to determine on the primary whether an incoming replicated row change from the secondary is concurrent with a locally committed change; concurrent changes are regarded as conflicting, with subesequent exceptions table updates and realignment of the secondary. A problem arises when a row is deleted on the primary so there is no longer any last-modified epoch available to determine whether any replicated operations conflict, which means that conflicting delete operationss are not detected. This can result in divergence, an example being a delete on one cluster which is concurrent with a delete and insert on the other; this why delete operations can be routed to only one cluster when using NDB$EPOCH() and NDB$EPOCH_TRANS().

NDB$EPOCH2() bypasses the issue just described—storing information about deleted rows on the PRIMARY—by ignoring any delete-delete conflict, and by avoiding any potential resultant divergence as well. This is accomplished by reflecting any operation successfully applied on and replicated from the secondary back to the secondary. On its return to the secondary, it can be used to reapply an operation on the secondary which was deleted by an operation originating from the primary.

When using `NDB$EPOCH2()`, you should keep in mind that the secondary applies the delete from the primary, removing the new row until it is restored by a reflected operation. In theory, the subsequent insert or update on the secondary conflicts with the delete from the primary, but in this case, we choose to ignore this and allow the secondary to “win”, in the interest of preventing divergence between the clusters. In other words, after a delete, the primary does not detect conflicts, and instead adopts the secondary's following changes immediately. Because of this, the secondary's state can revisit multiple previous committed states as it progresses to a final (stable) state, and some of these may be visible.

You should also be aware that reflecting all operations from the secondary back to the primary increases the size of the primary's logbinary log, as well as demands on bandwidth, CPU usage, and disk I/O.

Application of reflected operations on the secondary depends on the state of the target row on the secondary. Whether or not reflected changes are applied on the secondary can be tracked by checking the `Ndb_conflict_reflected_op_prepare_count` and `Ndb_conflict_reflected_op_discard_count` status variables (both added in MySQL Cluster NDB 7.4.2). The number of changes applied is simply the difference between these two values (note that `Ndb_conflict_reflected_op_prepare_count` is always greater than or equal to `Ndb_conflict_reflected_op_discard_count`).

Events are applied if and only if both of the following conditions are true:

- The existence of the row—that is, whether or not it exists—is in accordance with the type of event. For delete and update operations, the row must already exist; for insert operations, the row must *not* exist.
- The row was last modified by the primary. It is possible that the modification was accomplished through the execution of a reflected operation.

If both of the conditions are not met, the reflected operation is discarded by the secondary.

Conflict resolution exceptions table. To use the `NDB$OLD()` conflict resolution function, it is also necessary to create an exceptions table corresponding to each `NDB` table for which this type of conflict resolution is to be employed. This is also true when using `NDB$EPOCH()` or `NDB$EPOCH_TRANS()`. The name of this table is that of the table for which conflict resolution is to be applied, with the string `$EX` appended. (For example, if the name of the original table is `mytable`, the name of the corresponding exceptions table name should be `mytable$EX`.) Prior to MySQL Cluster NDB 7.4.1, this table is created as shown:

```
CREATE TABLE original_table$EX  (
    server_id INT UNSIGNED,
    master_server_id INT UNSIGNED,
    master_epoch BIGINT UNSIGNED,
    count INT UNSIGNED,
    original_table_pk_columns,
    [additional_columns,]
    PRIMARY KEY(server_id, master_server_id, master_epoch, count)
) ENGINE=NDB;
```

MySQL Cluster NDB 7.4.1 and later support an extended exceptions table definition that includes optional columns providing information about an exception's type, cause, and originating transaction. In these versions, the syntax for creating the exceptions table is as shown here:

```
CREATE TABLE original_table$EX  (
    [NDB$]server_id INT UNSIGNED,
    [NDB$]master_server_id INT UNSIGNED,
    [NDB$]master_epoch BIGINT UNSIGNED,
    [NDB$]count INT UNSIGNED,
    [NDB$OP_TYPE ENUM('WRITE_ROW','UPDATE_ROW', 'DELETE_ROW',
```

```
      'REFRESH_ROW', 'READ_ROW') NOT NULL,]
    [NDB$CFT_CAUSE ENUM('ROW_DOES_NOT_EXIST', 'ROW_ALREADY_EXISTS',
      'DATA_IN_CONFLICT', 'TRANS_IN_CONFLICT') NOT NULL,]
    [NDB$ORIG_TRANSID BIGINT UNSIGNED NOT NULL,]
    original_table_pk_columns,
    [orig_table_column|orig_table_column$OLD|orig_table_column$NEW,]
    [additional_columns,]
    PRIMARY KEY([NDB$]server_id, [NDB$]master_server_id, [NDB$]master_epoch, [NDB$]count)
) ENGINE=NDB;
```

The first four columns are required. The names of the first four columns and the columns matching the original table's primary key columns are not critical; however, we suggest for reasons of clarity and consistency, that you use the names shown here for the `server_id`, `master_server_id`, `master_epoch`, and `count` columns, and that you use the same names as in the original table for the columns matching those in the original table's primary key.

Starting with MySQL Cluster NDB 7.4.1, if the exceptions table uses one or more of the optional columns `NDB$OP_TYPE`, `NDB$CFT_CAUSE`, or `NDB$ORIG_TRANSID` discussed later in this section, then each of the required columns must also be named using the prefix `NDB$`. If desired, you can use the `NDB$` prefix to name the required columns even if you do not define any optional columns, but in this case, all four of the required columns must be named using the prefix.

Following these columns, the columns making up the original table's primary key should be copied in the order in which they are used to define the primary key of the original table. The data types for the columns duplicating the primary key columns of the original table should be the same as (or larger than) those of the original columns. In MySQL Cluster NDB 7.3 and earlier, the exceptions table's primary key must be reproduced column for column. Beginning with MySQL Cluster NDB 7.4.1, a subset of the primary key columns may be used instead.

Regardless of the MySQL Cluster version employed, the exceptions table must use the `NDB` storage engine. (An example that uses `NDB$OLD()` with an exceptions table is shown later in this section.)

Additional columns may optionally be defined following the copied primary key columns, but not before any of them; any such extra columns cannot be `NOT NULL`. In MySQL Cluster NDB 7.4.1 and later, support is provided for three additional, predefined optional columns `NDB$OP_TYPE`, `NDB$CFT_CAUSE`, and `NDB$ORIG_TRANSID`, which are described in the next few paragraphs.

`NDB$OP_TYPE`: This column can be used to obtain the type of operation causing the conflict. If you use this column, define it as shown here:

```
NDB$OP_TYPE ENUM('WRITE_ROW', 'UPDATE_ROW', 'DELETE_ROW',
    'REFRESH_ROW', 'READ_ROW') NOT NULL
```

The `WRITE_ROW`, `UPDATE_ROW`, and `DELETE_ROW` operation types represent user-initiated operations. `REFRESH_ROW` operations are operations generated by conflict resolution in compensating transactions sent back to the originating cluster from the cluster that detected the conflict. `READ_ROW` operations are user-initiated read tracking operations defined with exclusive row locks.

`NDB$CFT_CAUSE`: You can define an optional column `NDB$CFT_CAUSE` which provides the cause of the registered conflict. This column, if used, is defined as shown here:

```
NDB$CFT_CAUSE ENUM('ROW_DOES_NOT_EXIST', 'ROW_ALREADY_EXISTS',
    'DATA_IN_CONFLICT', 'TRANS_IN_CONFLICT') NOT NULL
```

`ROW_DOES_NOT_EXIST` can be reported as the cause for `UPDATE_ROW` and `WRITE_ROW` operations; `ROW_ALREADY_EXISTS` can be reported for `WRITE_ROW` events. `DATA_IN_CONFLICT` is reported when a row-based conflict function detects a conflict; `TRANS_IN_CONFLICT` is reported when a transactional conflict function rejects all of the operations belonging to a complete transaction.

`NDB$ORIG_TRANSID`: The `NDB$ORIG_TRANSID` column, if used, contains the ID of the originating transaction. This column should be defined as follows:

```
NDB$ORIG_TRANSID BIGINT UNSIGNED NOT NULL
```

NDB$ORIG_TRANSID is a 64-bit value generated by NDB. This value can be used to correlate multiple exceptions table entries belonging to the same conflicting transaction from the same or different exceptions tables.

In MySQL Cluster NDB 7.4.1 and later, additional reference columns which are not part of the original table's primary key can be named *colname*$OLD or *colname*$NEW. *colname*$OLD references old values in update and delete operations—that is, operations containing DELETE_ROW events. *colname*$NEW can be used to reference new values in insert and update operations—in other words, operations using WRITE_ROW events, UPDATE_ROW events, or both types of events. Where a conflicting operation does not supply a value for a given non-primary-key reference column, the exceptions table row contains either NULL, or a defined default value for that column.

> **Important**
>
> The mysql.ndb_replication table is read when a data table is set up for replication, so the row corresponding to a table to be replicated must be inserted into mysql.ndb_replication *before* the table to be replicated is created.

Examples

The following examples assume that you have already a working MySQL Cluster replication setup, as described in Section 8.5, “Preparing the MySQL Cluster for Replication”, and Section 8.6, “Starting MySQL Cluster Replication (Single Replication Channel)”.

NDB$MAX() example. Suppose you wish to enable “greatest timestamp wins” conflict resolution on table test.t1, using column mycol as the “timestamp”. This can be done using the following steps:

1. Make sure that you have started the master mysqld with --ndb-log-update-as-write=OFF [474].

2. On the master, perform this INSERT statement:

   ```
   INSERT INTO mysql.ndb_replication
       VALUES ('test', 't1', 0, NULL, 'NDB$MAX(mycol)');
   ```

   Inserting a 0 into the server_id indicates that all SQL nodes accessing this table should use conflict resolution. If you want to use conflict resolution on a specific mysqld only, use the actual server ID.

   Inserting NULL into the binlog_type column has the same effect as inserting 0 (NBT_DEFAULT); the server default is used.

3. Create the test.t1 table:

   ```
   CREATE TABLE test.t1 (
       columns
       mycol INT UNSIGNED,
       columns
   ) ENGINE=NDB;
   ```

   Now, when updates are done on this table, conflict resolution is applied, and the version of the row having the greatest value for mycol is written to the slave.

   > **Note**
   >
   > Other binlog_type options—such as NBT_UPDATED_ONLY_USE_UPDATE should be used to control logging on the master using the ndb_replication table rather than by using command-line options.

NDB$OLD() example. Suppose an NDB table such as the one defined here is being replicated, and you wish to enable “same timestamp wins” conflict resolution for updates to this table:

```
CREATE TABLE test.t2  (
    a INT UNSIGNED NOT NULL,
    b CHAR(25) NOT NULL,
    columns,
    mycol INT UNSIGNED NOT NULL,
    columns,
    PRIMARY KEY pk (a, b)
)   ENGINE=NDB;
```

The following steps are required, in the order shown:

1. First—and *prior* to creating test.t2—you must insert a row into the mysql.ndb_replication table, as shown here:

   ```
   INSERT INTO mysql.ndb_replication
       VALUES ('test', 't2', 0, NULL, 'NDB$OLD(mycol)');
   ```

   Possible values for the binlog_type column are shown earlier in this section. The value 'NDB $OLD(mycol)' should be inserted into the conflict_fn column.

2. Create an appropriate exceptions table for test.t2. The table creation statement shown here includes all required columns; any additional columns must be declared following these columns, and before the definition of the table's primary key.

   ```
   CREATE TABLE test.t2$EX  (
       server_id SMALLINT UNSIGNED,
       master_server_id INT UNSIGNED,
       master_epoch BIGINT UNSIGNED,
       count BIGINT UNSIGNED,
       a INT UNSIGNED NOT NULL,
       b CHAR(25) NOT NULL,
       [additional_columns,]
       PRIMARY KEY(server_id, master_server_id, master_epoch, count)
   )   ENGINE=NDB;
   ```

   In MySQL Cluster NDB 7.4.1 and later, we can include additional columns for information about the type, cause, and originating transaction ID for a given conflict. We are also not required to supply matching columns for all primary key columns in the original table. In these versions, you can create the exceptions table like this:

   ```
   CREATE TABLE test.t2$EX  (
       NDB$server_id SMALLINT UNSIGNED,
       NDB$master_server_id INT UNSIGNED,
       NDB$master_epoch BIGINT UNSIGNED,
       NDB$count BIGINT UNSIGNED,
       a INT UNSIGNED NOT NULL,

       NDB$OP_TYPE ENUM('WRITE_ROW','UPDATE_ROW', 'DELETE_ROW',
         'REFRESH_ROW', 'READ_ROW') NOT NULL,
       NDB$CFT_CAUSE ENUM('ROW_DOES_NOT_EXIST', 'ROW_ALREADY_EXISTS',
         'DATA_IN_CONFLICT', 'TRANS_IN_CONFLICT') NOT NULL,
       NDB$ORIG_TRANSID BIGINT UNSIGNED NOT NULL,
       [additional_columns,]
       PRIMARY KEY(NDB$server_id, NDB$master_server_id, NDB$master_epoch, NDB$count)
   )   ENGINE=NDB;
   ```

   > **Note**
   >
   > The NDB$ prefix is required for the four required columns since we included at least one of the columns NDBOP_TYPE, NDBCFT_CAUSE, or NDB $ORIG_TRANSID in the table definition.

3. Create the table `test.t2` as shown previously.

These steps must be followed for every table for which you wish to perform conflict resolution using `NDB$OLD()`. For each such table, there must be a corresponding row in `mysql.ndb_replication`, and there must be an exceptions table in the same database as the table being replicated.

Read conflict detection and resolution. MySQL Cluster NDB 7.4.1 and later support tracking of read operations, which makes it possible in circular replication setups to manage conflicts between reads of a given row in one cluster and updates or deletes of the same row in another. This example uses `employee` and `department` tables to model a scenario in which an employee is moved from one department to another on the master cluster (which we refer to hereafter as cluster *A*) while the slave cluster (hereafter *B*) updates the employee count of the employee's former department in an interleaved transaction.

The data tables have been created using the following SQL statements:

```
# Employee table
CREATE TABLE employee (
    id INT PRIMARY KEY,
    name VARCHAR(2000),
    dept INT NOT NULL
)   ENGINE=NDB;
# Department table
CREATE TABLE department (
    id INT PRIMARY KEY,
    name VARCHAR(2000),
    members INT
)   ENGINE=NDB;
```

The contents of the two tables include the rows shown in the (partial) output of the following `SELECT` statements:

```
mysql> SELECT id, name, dept FROM employee;
+---------------+------+
| id   | name   | dept |
+------+--------+------+
...
| 998  |  Mike  | 3    |
| 999  |  Joe   | 3    |
| 1000 |  Mary  | 3    |
...
+------+--------+------+
mysql> SELECT id, name, members FROM department;
+-----+-------------+---------+
| id  | name        | members |
+-----+-------------+---------+
...
| 3   | Old project | 24      |
...
+-----+-------------+---------+
```

We assume that we are already using an exceptions table that includes the four required columns (and these are used for this table's primary key), the optional columns for operation type and cause, and the original table's primary key column, created using the SQL statement shown here:

```
CREATE TABLE employee$EX  (
    NDB$server_id INT UNSIGNED,
    NDB$master_server_id INT UNSIGNED,
    NDB$master_epoch BIGINT UNSIGNED,
    NDB$count INT UNSIGNED,
    NDB$OP_TYPE ENUM( 'WRITE_ROW','UPDATE_ROW', 'DELETE_ROW',
                      'REFRESH_ROW','READ_ROW') NOT NULL,
    NDB$CFT_CAUSE ENUM( 'ROW_DOES_NOT_EXIST',
                        'ROW_ALREADY_EXISTS',
                        'DATA_IN_CONFLICT',
```

```
                         'TRANS_IN_CONFLICT') NOT NULL,
    id INT NOT NULL,

    PRIMARY KEY(NDB$server_id, NDB$master_server_id, NDB$master_epoch, NDB$count)
)   ENGINE=NDB;
```

Suppose there occur the two simultaneous transactions on the two clusters. On cluster *A*, we create a new department, then move employee number 999 into that department, using the following SQL statements:

```
BEGIN;
  INSERT INTO department VALUES (4, "New project", 1);
  UPDATE employee SET dept = 4 WHERE id = 999;
COMMIT;
```

At the same time, on cluster *B*, another transaction reads from `employee`, as shown here:

```
BEGIN;
  SELECT name FROM employee WHERE id = 999;
  UPDATE department SET members = members - 1  WHERE id = 3;
commit;
```

The conflicting transactions are not normally detected by the conflict resolution mechanism, since the conflict is between a read (`SELECT`) and an update operation. Beginning with MySQL Cluster NDB 7.4.1, we can circumvent this issue by executing `SET ndb_log_exclusive_reads = 1` on the slave cluster. Acquiring exclusive read locks in this way causes any rows read on the master to be flagged as needing conflict resolution on the slave cluster. If we enable exclusive reads in this way prior to the logging of these transactions, the read on cluster *B* is tracked and sent to cluster *A* for resolution; the conflict on the employee row will be detected and the transaction on cluster *B* is aborted.

The conflict is registered in the exceptions table (on cluster *A*) as a `READ_ROW` operation (see Conflict resolution exceptions table, for a description of operation types), as shown here:

```
mysql> SELECT id, NDB$OP_TYPE, NDB$CFT_CAUSE FROM employee$EX;
+-------+-------------+-------------------+
| id    | NDB$OP_TYPE | NDB$CFT_CAUSE     |
+-------+-------------+-------------------+
...
| 999   | READ_ROW    | TRANS_IN_CONFLICT |
+-------+-------------+-------------------+
```

Any existing rows found in the read operation are flagged. This means that multiple rows resulting from the same conflict may be logged in the exception table, as shown by examining the effects a conflict between an update on cluster *A* and a read of multiple rows on cluster *B* from the same table in simultaneous transactions. The transaction executed on cluster *A* is shown here:

```
BEGIN;
  INSERT INTO department VALUES (4, "New project", 0);
  UPDATE employee SET dept = 4 WHERE dept = 3;
  SELECT COUNT(*) INTO @count FROM employee WHERE dept = 4;
  UPDATE department SET members = @count WHERE id = 4;
COMMIT;
```

Concurrently a transaction containing the statements shown here runs on cluster *B*:

```
SET ndb_log_exclusive_reads = 1;  # Must be set if not already enabled
...
BEGIN;
  SELECT COUNT(*) INTO @count FROM employee WHERE dept = 3 FOR UPDATE;
  UPDATE department SET members = @count WHERE id = 3;
COMMIT;
```

In this case, all three rows matching the `WHERE` condition in the second transaction's `SELECT` are read, and are thus flagged in the exceptions table, as shown here:

```
mysql> SELECT id, NDB$OP_TYPE, NDB$CFT_CAUSE FROM employee$EX;
+-------+-------------+-------------------+
| id    | NDB$OP_TYPE | NDB$CFT_CAUSE     |
+-------+-------------+-------------------+
...
| 998   | READ_ROW    | TRANS_IN_CONFLICT |
| 999   | READ_ROW    | TRANS_IN_CONFLICT |
| 1000  | READ_ROW    | TRANS_IN_CONFLICT |
...
+-------+-------------+-------------------+
```

Read tracking is performed on the basis of existing rows only. A read based on a given condition track conflicts only of any rows that are *found* and not of any rows that are inserted in an interleaved transaction. This is similar to how exclusive row locking is performed in a single instance of MySQL Cluster.

Appendix A Licenses for Third-Party Components

Table of Contents

The following is a list of the libraries we have included with the MySQL Server source and components used to test MySQL. We are thankful to all individuals that have created these. Some of the components require that their licensing terms be included in the documentation of products that include them. Cross references to these licensing terms are given with the applicable items in the list.

- GroupLens Research Project

  The MySQL Quality Assurance team would like to acknowledge the use of the MovieLens Data Sets (10 million ratings and 100,000 tags for 10681 movies by 71567 users) to help test MySQL products and to thank the GroupLens Research Project at the University of Minnesota for making the data sets available.

MySQL 5.6

- Section A.2, “Artistic License (Perl) 1.0”
- Section A.3, “Boost Library License”
- Section A.4, “dtoa.c License”
- Section A.5, “Editline Library (libedit) License”
- Section A.7, “Expect.pm License”
- Section A.8, “Facebook Fast Checksum Patch License”
- Section A.9, “Facebook Patches License”
- Section A.10, “FindGTest.cmake License”
- Section A.11, “Fred Fish's Dbug Library License”
- Section A.12, “getarg License”
- Section A.13, “GNU General Public License Version 2.0, June 1991”
- Section A.14, “GNU General Public License Version 3.0, 29 June 2007 and GCC Runtime Library Exception Version 3.1, 31 March 2009”
- Section A.15, “GNU Lesser General Public License Version 2.1, February 1999”
- Section A.17, “GNU Readline License”
- Section A.18, “GNU Standard C++ Library (libstdc++) License”
- Section A.19, “Google C++ Mocking Framework (Google Mock) License”
- Section A.20, “Google Controlling Master Thread I/O Rate Patch License”
- Section A.21, “Google Perftools (TCMalloc utility) License”
- Section A.22, “Google SMP Patch License”
- Section A.23, “Janson License”
- Section A.24, “lib_sql.cc License”
- Section A.25, “Libaio License”
- Section A.26, “libevent License”
- Section A.27, “Linux-PAM License”
- Section A.28, “md5 (Message-Digest Algorithm 5) License”
- Section A.29, “memcached License”
- Section A.30, “Memcached.pm License”

- Section A.31, “mkpasswd.pl License”
- Section A.34, “nt_servc (Windows NT Service class library) License”
- Section A.35, “OpenPAM License”
- Section A.36, “OpenSSL v1.0 License”
- Section A.38, “Percona Multiple I/O Threads Patch License”
- Section A.39, “Pion License”
- Section A.41, “Red HAT RPM Spec File License”
- Section A.42, “RegEX-Spencer Library License”
- Section A.43, “Richard A. O'Keefe String Library License”
- Section A.44, “sajson License”
- Section A.45, “SHA-1 in C License”
- Section A.46, “Unicode Data Files”
- Section A.48, “zlib License”

MySQL Cluster 7.3

- Section A.1, “ANTLR 3 License”
- Section A.4, “dtoa.c License”
- Section A.6, “Editline Library (libedit) License”
- Section A.10, “FindGTest.cmake License”
- Section A.11, “Fred Fish's Dbug Library License”
- Section A.12, “getarg License”
- Section A.13, “GNU General Public License Version 2.0, June 1991”
- Section A.15, “GNU Lesser General Public License Version 2.1, February 1999”
- Section A.16, “GNU Libtool License”
- Section A.17, “GNU Readline License”
- Section A.20, “Google Controlling Master Thread I/O Rate Patch License”
- Section A.21, “Google Perftools (TCMalloc utility) License”
- Section A.22, “Google SMP Patch License”
- Section A.24, “lib_sql.cc License”
- Section A.26, “libevent License”
- Section A.27, “Linux-PAM License”
- Section A.28, “md5 (Message-Digest Algorithm 5) License”
- Section A.29, “memcached License”
- Section A.32, “node-mysql License”
- Section A.33, “Node.js License”
- Section A.34, “nt_servc (Windows NT Service class library) License”

A.1 ANTLR 3 License

```
The following software may be included in this product:

ANTLR 3

ANTLR 3 License
[The BSD License]
Copyright (c) 2003-2007, Terence Parr
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in
  the documentation and/or other materials provided with the
  distribution.
* Neither the name of the author nor the names of its contributors
  may be used to endorse or promote products derived from this
  software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

A.2 Artistic License (Perl) 1.0

```
The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a
Package may be copied, such that the Copyright Holder maintains some
semblance of artistic control over the development of the package,
while giving the users of the package the right to use and distribute
the Package in a more-or-less customary fashion, plus the right to make
reasonable modifications.
```

```
Definitions:

        "Package" refers to the collection of files distributed by the
        Copyright Holder, and derivatives of that collection of files
        created through textual modification.

        "Standard Version" refers to such a Package if it has not been
        modified, or has been modified in accordance with the wishes
        of the Copyright Holder as specified below.

        "Copyright Holder" is whoever is named in the copyright or
        copyrights for the package.

        "You" is you, if you're thinking about copying or distributing
        this Package.

        "Reasonable copying fee" is whatever you can justify on the
        basis of media cost, duplication charges, time of people involved,
        and so on.  (You will not be required to justify it to the
        Copyright Holder, but only to the computing community at large
        as a market that must bear the fee.)

        "Freely Available" means that no fee is charged for the item
        itself, though there may be fees involved in handling the item.
        It also means that recipients of the item may redistribute it
        under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the
Standard Version of this Package without restriction, provided that you
duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications
derived from the Public Domain or from the Copyright Holder.  A Package
modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided
that you insert a prominent notice in each changed file stating how and
when you changed that file, and provided that you do at least ONE of the
following:

    a) place your modifications in the Public Domain or otherwise make them
    Freely Available, such as by posting said modifications to Usenet or
    an equivalent medium, or placing the modifications on a major archive
    site such as uunet.uu.net, or by allowing the Copyright Holder to include
    your modifications in the Standard Version of the Package.

    b) use the modified Package only within your corporation or organization.

    c) rename any non-standard executables so the names do not conflict
    with standard executables, which must also be provided, and provide
    a separate manual page for each non-standard executable that clearly
    documents how it differs from the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or
executable form, provided that you do at least ONE of the following:

    a) distribute a Standard Version of the executables and library files,
    together with instructions (in the manual page or equivalent) on where
    to get the Standard Version.

    b) accompany the distribution with the machine-readable source of
    the Package with your modifications.

    c) give non-standard executables non-standard names, and clearly
    document the differences in manual pages (or equivalent), together
    with instructions on where to get the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.
```

```
5. You may charge a reasonable copying fee for any distribution of this
Package.  You may charge any fee you choose for support of this
Package.  You may not charge a fee for this Package itself.  However,
you may distribute this Package in aggregate with other (possibly
commercial) programs as part of a larger (possibly commercial) software
distribution provided that you do not advertise this Package as a
product of your own.  You may embed this Package's interpreter within
an executable of yours (by linking); this shall be construed as a mere
form of aggregation, provided that the complete Standard Version of the
interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as
output from the programs of this Package do not automatically fall
under the copyright of this Package, but belong to whoever generated
them, and may be sold commercially, and may be aggregated with this
Package.  If such scripts or library files are aggregated with this
Package via the so-called "undump" or "unexec" methods of producing a
binary executable image, then distribution of such an image shall
neither be construed as a distribution of this Package nor shall it
fall under the restrictions of Paragraphs 3 and 4, provided that you do
not represent such an executable image as a Standard Version of this
Package.

7. C subroutines (or comparably compiled subroutines in other
languages) supplied by you and linked into this Package in order to
emulate subroutines and variables of the language defined by this
Package shall not be considered part of this Package, but are the
equivalent of input as in Paragraph 6, provided these subroutines do
not change the language in any way that would cause it to fail the
regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always
permitted provided that the use of this Package is embedded; that is,
when no overt attempt is made to make this Package's interfaces visible
to the end user of the commercial distribution.  Such use shall not be
construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written
permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

                                The End
```

A.3 Boost Library License

The following software may be included in this product:

Boost C++ Libraries

Use of any of this software is governed by the terms of the license below:

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or
organization obtaining a copy of the software and accompanying
documentation covered by this license (the "Software") to use,
reproduce, display, distribute, execute, and transmit the Software,
and to prepare derivative works of the Software, and to permit
third-parties to whom the Software is furnished to do so, all
subject to the following:

The copyright notices in the Software and this entire statement,
including the above license grant, this restriction and the
following disclaimer, must be included in all copies of the
Software, in whole or in part, and all derivative works of the
Software, unless such copies or derivative works are solely in the
```

A.4 dtoa.c License

The following software may be included in this product:

`dtoa.c`

A.5 Editline Library (libedit) License

The following software may be included in this product:

Editline Library (libedit)

Some files are:

Some files are:

Some files are:

Some files are:

Some files are:

Some files are:

A.6 Editline Library (libedit) License

The following software may be included in this product:

Editline Library (libedit)

Some files are:

Some files are:

Some files are:

```
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

Some files are:

```
Copyright (c) 1998 Todd C. Miller <Todd.Miller@courtesan.com>

Permission to use, copy, modify, and distribute this
software for any purpose with or without fee is hereby
granted, provided that the above copyright notice and
this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND TODD C. MILLER
DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL TODD C. MILLER BE LIABLE
FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION
OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

A.7 Expect.pm License

The following software may be included in this product:

Expect.pm Perl module

```
Expect.pm is licensed under the Perl license, which is essentially a dual
license.

Oracle may use, redistribute and/or modify this code under the terms of
either:

 a) the GNU General Public License as published by the Free Software
Foundation; either version 1, or (at your option) any later version, or

 b) the "Artistic License" which comes with the Expect/pr code.

Oracle elects to use the GPLv2 for version of MySQL that are licensed under
the GPL.

Oracle elects to use the Artistic license for all other (commercial) versions
of MySQL.

A copy of the GPLv2 and the Artistic License (Perl) 1.0 must be included with
any distribution:

The GNU General Public License (GPL-2.0)
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to
share and change it. By contrast, the GNU General Public License is intended
```

```
2. You may modify your copy or copies of the Program or any portion of it,
thus forming a work based on the Program, and copy and distribute such
modifications or work under the terms of Section 1 above, provided that you
also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices stating
that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in whole
or in part contains or is derived from the Program or any part thereof, to be
licensed as a whole at no charge to all third parties under the terms of this
License.

    c) If the modified program normally reads commands interactively when
run, you must cause it, when started running for such interactive use in the
most ordinary way, to print or display an announcement including an
appropriate copyright notice and a notice that there is no warranty (or else,
saying that you provide a warranty) and that users may redistribute the
program under these conditions, and telling the user how to view a copy of
this License. (Exception: if the Program itself is interactive but does not
normally print such an announcement, your work based on the Program is not
required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable
sections of that work are not derived from the Program, and can be reasonably
considered independent and separate works in themselves, then this License,
and its terms, do not apply to those sections when you distribute them as
separate works. But when you distribute the same sections as part of a whole
which is a work based on the Program, the distribution of the whole must be
on the terms of this License, whose permissions for other licensees extend to
the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your
rights to work written entirely by you; rather, the intent is to exercise the
right to control the distribution of derivative or collective works based on
the Program.

In addition, mere aggregation of another work not based on the Program with
the Program (or with a work based on the Program) on a volume of a storage or
distribution medium does not bring the other work under the scope of this
License.

3. You may copy and distribute the Program (or a work based on it, under
Section 2) in object code or executable form under the terms of Sections 1
and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable source
code, which must be distributed under the terms of Sections 1 and 2 above on
a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three years, to
give any third party, for a charge no more than your cost of physically
performing source distribution, a complete machine-readable copy of the
corresponding source code, to be distributed under the terms of Sections 1
and 2 above on a medium customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer to
distribute corresponding source code. (This alternative is allowed only for
noncommercial distribution and only if you received the program in object
code or executable form with such an offer, in accord with Subsection b
above.)

The source code for a work means the preferred form of the work for making
modifications to it. For an executable work, complete source code means all
the source code for all modules it contains, plus any associated interface
definition files, plus the scripts used to control compilation and
installation of the executable. However, as a special exception, the source
code distributed need not include anything that is normally distributed (in
either source or binary form) with the major components (compiler, kernel,
and so on) of the operating system on which the executable runs, unless that
component itself accompanies the executable.
```

```
If distribution of executable or object code is made by offering access to
copy from a designated place, then offering equivalent access to copy the
source code from the same place counts as distribution of the source code,
even though third parties are not compelled to copy the source along with the
object code.

4. You may not copy, modify, sublicense, or distribute the Program except as
expressly provided under this License. Any attempt otherwise to copy, modify,
sublicense or distribute the Program is void, and will automatically
terminate your rights under this License. However, parties who have received
copies, or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it.
However, nothing else grants you permission to modify or distribute the
Program or its derivative works. These actions are prohibited by law if you
do not accept this License. Therefore, by modifying or distributing the
Program (or any work based on the Program), you indicate your acceptance of
this License to do so, and all its terms and conditions for copying,
distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program),
the recipient automatically receives a license from the original licensor to
copy, distribute or modify the Program subject to these terms and conditions.
You may not impose any further restrictions on the recipients' exercise of
the rights granted herein. You are not responsible for enforcing compliance
by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not excuse
you from the conditions of this License. If you cannot distribute so as to
satisfy simultaneously your obligations under this License and any other
pertinent obligations, then as a consequence you may not distribute the
Program at all. For example, if a patent license would not permit
royalty-free redistribution of the Program by all those who receive copies
directly or indirectly through you, then the only way you could satisfy both
it and this License would be to refrain entirely from distribution of the
Program.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply and
the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents
or other property right claims or to contest validity of any such claims;
this section has the sole purpose of protecting the integrity of the free
software distribution system, which is implemented by public license
practices. Many people have made generous contributions to the wide range of
software distributed through that system in reliance on consistent
application of that system; it is up to the author/donor to decide if he or
she is willing to distribute software through any other system and a licensee
cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a
consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain
countries either by patents or by copyrighted interfaces, the original
copyright holder who places the Program under this License may add an
explicit geographical distribution limitation excluding those countries, so
that distribution is permitted only in or among countries not thus excluded.
In such case, this License incorporates the limitation as if written in the
body of this License.

9. The Free Software Foundation may publish revised and/or new versions of
the General Public License from time to time. Such new versions will be
similar in spirit to the present version, but may differ in detail to address
new problems or concerns.

Each version is given a distinguishing version number. If the Program
```

```
specifies a version number of this License which applies to it and "any later
version", you have the option of following the terms and conditions either of
that version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of this License,
you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs
whose distribution conditions are different, write to the author to ask for
permission. For software which is copyrighted by the Free Software
Foundation, write to the Free Software Foundation; we sometimes make
exceptions for this. Our decision will be guided by the two goals of
preserving the free status of all derivatives of our free software and of
promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR
THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE
STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE
PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND
PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE,
YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO
LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR
THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible
use to the public, the best way to achieve this is to make it free software
which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach
them to the start of each source file to most effectively convey the
exclusion of warranty; and each file should have at least the "copyright"
line and a pointer to where the full notice is found.

    One line to give the program's name and a brief idea of what it does.
    Copyright (C) <year> <name of author>

    This program is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the Free
Software Foundation; either version 2 of the License, or (at your option) any
later version.

    This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
details.

    You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc., 59
Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when
it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author Gnomovision
comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free
```

```
software, and you are welcome to redistribute it under certain conditions;
type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License. Of course, the commands you use may be
called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

    Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.

    signature of Ty Coon, 1 April 1989
    Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General Public
License instead of this License.

________________________________________________________

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a
Package may be copied, such that the Copyright Holder maintains some
semblance of artistic control over the development of the package,
while giving the users of the package the right to use and distribute
the Package in a more-or-less customary fashion, plus the right to make
reasonable modifications.

Definitions:

        "Package" refers to the collection of files distributed by the
        Copyright Holder, and derivatives of that collection of files
        created through textual modification.

        "Standard Version" refers to such a Package if it has not been
        modified, or has been modified in accordance with the wishes
        of the Copyright Holder as specified below.

        "Copyright Holder" is whoever is named in the copyright or
        copyrights for the package.

        "You" is you, if you're thinking about copying or distributing
        this Package.

        "Reasonable copying fee" is whatever you can justify on the
        basis of media cost, duplication charges, time of people involved,
        and so on.  (You will not be required to justify it to the
        Copyright Holder, but only to the computing community at large
        as a market that must bear the fee.)

        "Freely Available" means that no fee is charged for the item
        itself, though there may be fees involved in handling the item.
        It also means that recipients of the item may redistribute it
        under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the
Standard Version of this Package without restriction, provided that you
duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications
derived from the Public Domain or from the Copyright Holder.  A Package
modified in such a way shall still be considered the Standard Version.
```

```
3. You may otherwise modify your copy of this Package in any way, provided
that you insert a prominent notice in each changed file stating how and
when you changed that file, and provided that you do at least ONE of the
following:

    a) place your modifications in the Public Domain or otherwise make them
    Freely Available, such as by posting said modifications to Usenet or
    an equivalent medium, or placing the modifications on a major archive
    site such as uunet.uu.net, or by allowing the Copyright Holder to include
    your modifications in the Standard Version of the Package.

    b) use the modified Package only within your corporation or organization.

    c) rename any non-standard executables so the names do not conflict
    with standard executables, which must also be provided, and provide
    a separate manual page for each non-standard executable that clearly
    documents how it differs from the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or
executable form, provided that you do at least ONE of the following:

    a) distribute a Standard Version of the executables and library files,
    together with instructions (in the manual page or equivalent) on where
    to get the Standard Version.

    b) accompany the distribution with the machine-readable source of
    the Package with your modifications.

    c) give non-standard executables non-standard names, and clearly
    document the differences in manual pages (or equivalent), together
    with instructions on where to get the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this
Package.  You may charge any fee you choose for support of this
Package.  You may not charge a fee for this Package itself.  However,
you may distribute this Package in aggregate with other (possibly
commercial) programs as part of a larger (possibly commercial) software
distribution provided that you do not advertise this Package as a
product of your own.  You may embed this Package's interpreter within
an executable of yours (by linking); this shall be construed as a mere
form of aggregation, provided that the complete Standard Version of the
interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as
output from the programs of this Package do not automatically fall
under the copyright of this Package, but belong to whoever generated
them, and may be sold commercially, and may be aggregated with this
Package.  If such scripts or library files are aggregated with this
Package via the so-called "undump" or "unexec" methods of producing a
binary executable image, then distribution of such an image shall
neither be construed as a distribution of this Package nor shall it
fall under the restrictions of Paragraphs 3 and 4, provided that you do
not represent such an executable image as a Standard Version of this
Package.

7. C subroutines (or comparably compiled subroutines in other
languages) supplied by you and linked into this Package in order to
emulate subroutines and variables of the language defined by this
Package shall not be considered part of this Package, but are the
equivalent of input as in Paragraph 6, provided these subroutines do
not change the language in any way that would cause it to fail the
regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always
permitted provided that the use of this Package is embedded; that is,
when no overt attempt is made to make this Package's interfaces visible
to the end user of the commercial distribution.  Such use shall not be
construed as a distribution of this Package.
```

```
9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written
permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

                                The End
```

A.8 Facebook Fast Checksum Patch License

The following software may be included in this product:

```
Facebook Fast Checksum Patch

Copyright (C) 2009-2010 Facebook, Inc.  All Rights Reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY FACEBOOK, INC. “AS IS” AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
EVENT SHALL FACEBOOK, INC. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Also included:

crc32.c -- compute the CRC-32 of a buf stream
Copyright (C) 1995-2005 Mark Adler

This software is provided 'as-is', without any express or implied
warranty.  In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu
```

A.9 Facebook Patches License

The following software may be included in this product:

A.10 FindGTest.cmake License

The following software may be included in this product:

A.11 Fred Fish's Dbug Library License

The following software may be included in this product:

Fred Fish's Dbug Library

A.12 getarg License

The following software may be included in this product:

`getarg` Function (`getarg.h`, `getarg.c` files)

A.13 GNU General Public License Version 2.0, June 1991

```
  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software,
and (2) offer you this license which gives you legal permission to
copy, distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on,
we want its recipients to know that what they have is not the original,
so that any problems introduced by others will not reflect on the
original authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
```

```
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software
    interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as
a special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
```

```
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
```

```
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new
versions of the General Public License from time to time.  Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation.  If the Program does not specify a
version number of this License, you may choose any version ever
published by the Free Software Foundation.

  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the
author to ask for permission.  For software which is copyrighted by the
Free Software Foundation, write to the Free Software Foundation; we
sometimes make exceptions for this.  Our decision will be guided by the
two goals of preserving the free status of all derivatives of our free
software and of promoting the sharing and reuse of software generally.

                            NO WARRANTY

  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS
WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it
    does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License as
    published by     the Free Software Foundation; either version
    2 of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
```

```
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
    02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
    type 'show w'. This is free software, and you are welcome
    to redistribute it under certain conditions; type 'show c'
    for details.

The hypothetical commands 'show w' and 'show c' should show the
appropriate parts of the General Public License.  Of course, the
commands you use may be called something other than 'show w' and
'show c'; they could even be mouse-clicks or menu items--whatever
suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  program 'Gnomovision' (which makes passes at compilers) written
  by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library,
you may consider it more useful to permit linking proprietary
applications with the library.  If this is what you want to do, use
the GNU Lesser General Public License instead of this License.
```

A.14 GNU General Public License Version 3.0, 29 June 2007 and GCC Runtime Library Exception Version 3.1, 31 March 2009

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of
this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are
designed to take away your freedom to share and change the works.
By contrast, the GNU General Public License is intended to guarantee
your freedom to share and change all versions of a program--to make
sure it remains free software for all its users.  We, the Free
Software Foundation, use the GNU General Public License for most
of our software; it applies also to any other work released this
way by its authors.  You can apply it to your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and
charge for them if you wish), that you receive source code or can
get it if you want it, that you can change the software or use
```

```
pieces of it in new free programs, and that you know you can do
these things.

  To protect your rights, we need to prevent others from denying
you these rights or asking you to surrender the rights.  Therefore,
you have certain responsibilities if you distribute copies of the
software, or if you modify it: responsibilities to respect the
freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too,
receive or can get the source code.  And you must show them these
terms so they know their rights.

  Developers that use the GNU GPL protect your rights with two
steps: (1) assert copyright on the software, and (2) offer you this
License giving you legal permission to copy, distribute and/or
modify it.

  For the developers' and authors' protection, the GPL clearly
explains that there is no warranty for this free software.  For
both users' and authors' sake, the GPL requires that modified
versions be marked as changed, so that their problems will not be
attributed erroneously to authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the
manufacturer can do so.  This is fundamentally incompatible with
the aim of protecting users' freedom to change the software.  The
systematic pattern of such abuse occurs in the area of products for
individuals to use, which is precisely where it is most unacceptable.
Therefore, we have designed this version of the GPL to prohibit the
practice for those products.  If such problems arise substantially
in other domains, we stand ready to extend this provision to those
domains in future versions of the GPL, as needed to protect the
freedom of users.

  Finally, every program is threatened constantly by software
patents. States should not allow patents to restrict development
and use of software on general-purpose computers, but in those that
do, we wish to avoid the special danger that patents applied to a
free program could make it effectively proprietary.  To prevent
this, the GPL assures that patents cannot be used to render the
program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds
of works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the
work in a fashion requiring copyright permission, other than the
making of an exact copy.  The resulting work is called a "modified
version" of the earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
```

```
infringement under applicable copyright law, except executing it
on a computer or modifying a private copy.  Propagation includes
copying, distribution (with or without modification), making available
to the public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user
through a computer network, with no transfer of a copy, is not
conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to
the extent that warranties are provided), that licensees may convey
the work under this License, and how to view a copy of this License.
If the interface presents a list of user commands or options, such
as a menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case
of interfaces specified for a particular programming language, one
that is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form
of packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts
to control those activities.  However, it does not include the
work's System Libraries, or general-purpose tools or generally
available free programs which are used unmodified in performing
those activities but which are not part of the work.  For example,
Corresponding Source includes interface definition files associated
with source files for the work, and the source code for shared
libraries and dynamically linked subprograms that the work is
specifically designed to require, such as by intimate data communication
or control flow between those subprograms and other parts of the
work.

  The Corresponding Source need not include anything that users can
regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running
a covered work is covered by this License only if the output, given
its content, constitutes a covered work.  This License acknowledges
your rights of fair use or other equivalent, as provided by copyright
law.
```

```
  You may make, run and propagate covered works that you do not convey,
without conditions so long as your license otherwise remains in
force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide
you with facilities for running those works, provided that you
comply with the terms of this License in conveying all material for
which you do not control copyright.  Those thus making or running
the covered works for you must do so exclusively on your behalf,
under your direction and control, on terms that prohibit them from
making any copies of your copyrighted material outside their
relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section
10 makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect
to the covered work, and you disclaim any intention to limit operation
or modification of the work as a means of enforcing, against the
work's users, your or third parties' legal rights to forbid
circumvention of technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any non-permissive
terms added in accord with section 7 apply to the code; keep intact
all notices of the absence of any warranty; and give all recipients
a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications
to produce it from the Program, in the form of source code under
the terms of section 4, provided that you also meet all of these
conditions:

    a) The work must carry prominent notices stating that you
    modified it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under
    section 7.  This requirement modifies the requirement in
    section 4 to "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has
    interactive interfaces that do not display Appropriate Legal
```

```
    Notices, your work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called
an "aggregate" if the compilation and its resulting copyright are
not used to limit the access or legal rights of the compilation's
users beyond what the individual works permit.  Inclusion of a
covered work in an aggregate does not cause this License to apply
to the other parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the machine-readable
Corresponding Source under the terms of this License, in one of
these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that
    product model, to give anyone who possesses the object code
    either (1) a copy of the Corresponding Source for all the
    software in the product that is covered by this License, on a
    durable physical medium customarily used for software
    interchange, for a price no more than your reasonable cost
    of physically performing this conveying of source, or (2)
    access to copy the Corresponding Source from a network server
    at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in
    accord with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to
    the Corresponding Source in the same way through the same place
    at no further charge.  You need not require recipients to copy
    the Corresponding Source along with the object code.  If the
    place to copy the object code is a network server, the
    Corresponding Source may be on a different server (operated
    by you or a third party) that supports equivalent copying
    facilities, provided you maintain clear directions next to the
    object code saying where to find the Corresponding Source.
    Regardless of what server hosts the Corresponding Source, you
    remain obligated to ensure that it is available for as long
    as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission,
    provided you inform other peers where the object code and
    Corresponding Source of the work are being offered to the
    general public at no charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means
any tangible personal property which is normally used for personal,
family, or household purposes, or (2) anything designed or sold for
incorporation into a dwelling.  In determining whether a product
is a consumer product, doubtful cases shall be resolved in favor
of coverage.  For a particular product received by a particular
```

```
user, "normally used" refers to a typical or common use of that
class of product, regardless of the status of the particular user
or of the way in which the particular user actually uses, or expects
or is expected to use, the product.  A product is a consumer product
regardless of whether the product has substantial commercial,
industrial or non-consumer uses, unless such uses represent the
only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to
install and execute modified versions of a covered work in that
User Product from a modified version of its Corresponding Source.
The information must suffice to ensure that the continued functioning
of the modified object code is in no case prevented or interfered
with solely because modification has been made.

  If you convey an object code work under this section in, or with,
or specifically for use in, a User Product, and the conveying occurs
as part of a transaction in which the right of possession and use
of the User Product is transferred to the recipient in perpetuity
or for a fixed term (regardless of how the transaction is characterized),
the Corresponding Source conveyed under this section must be
accompanied by the Installation Information.  But this requirement
does not apply if neither you nor any third party retains the ability
to install modified object code on the User Product (for example,
the work has been installed in ROM).

  The requirement to provide Installation Information does not include
a requirement to continue to provide support service, warranty, or
updates for a work that has been modified or installed by the
recipient, or for the User Product in which it has been modified
or installed.  Access to a network may be denied when the modification
itself materially and adversely affects the operation of the network
or violates the rules and protocols for communication across the
network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of
this License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program
shall be treated as though they were included in this License, to
the extent that they are valid under applicable law.  If additional
permissions apply only to part of the Program, that part may be
used separately under those permissions, but the entire Program
remains governed by this License without regard to the additional
permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part
of it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material
you add to a covered work, you may (if authorized by the copyright
holders of that material) supplement the terms of this License with
terms:

    a) Disclaiming warranty or limiting liability differently from
    the terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices
    or author attributions in that material or in the Appropriate
    Legal Notices displayed by works containing it; or
```

```
    c) Prohibiting misrepresentation of the origin of that material,
    or requiring that modified versions of such material be marked
    in reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors
    or authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions
    of it) with contractual assumptions of liability to the
    recipient, for any liability that these contractual assumptions
    directly impose on those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as
you received it, or any part of it, contains a notice stating that
it is governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document
contains a further restriction but permits relicensing or conveying
under this License, you may add to a covered work material governed
by the terms of that license document, provided that the further
restriction does not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in
the form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate
or modify it is void, and will automatically terminate your rights
under this License (including any patent licenses granted under the
third paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from
that copyright holder, and you cure the violation prior to 30 days
after your receipt of the notice.

  Termination of your rights under this section does not terminate
the licenses of parties who have received copies or rights from you
under this License.  If your rights have been terminated and not
permanently reinstated, you do not qualify to receive new licenses
for the same material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
```

```
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you
may not impose a license fee, royalty, or other charge for exercise
of rights granted under this License, and you may not initiate
litigation (including a cross-claim or counterclaim in a lawsuit)
alleging that any patent claim is infringed by making, using,
selling, offering for sale, or importing the Program or any portion
of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired
or hereafter acquired, that would be infringed by some manner,
permitted by this License, of making, using, or selling its contributor
version, but do not include claims that would be infringed only as
a consequence of further modification of the contributor version.
For purposes of this definition, "control" includes the right to
grant patent sublicenses in a manner consistent with the requirements
of this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify
and propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a
patent (such as an express permission to practice a patent or
covenant not to sue for patent infringement).  To "grant" such a
patent license to a party means to make such an agreement or
commitment not to enforce a patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through
a publicly available network server or other readily accessible
means, then you must either (1) cause the Corresponding Source to
be so available, or (2) arrange to deprive yourself of the benefit
of the patent license for this particular work, or (3) arrange, in
a manner consistent with the requirements of this License, to extend
the patent license to downstream recipients.  "Knowingly relying"
means you have actual knowledge that, but for the patent license,
your conveying the covered work in a country, or your recipient's
use of the covered work in a country, would infringe one or more
identifiable patents in that country that you have reason to believe
are valid.
```

```
  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of,
a covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate,
modify or convey a specific copy of the covered work, then the
patent license you grant is automatically extended to all recipients
of the covered work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is conditioned
on the non-exercise of one or more of the rights that are specifically
granted under this License.  You may not convey a covered work if
you are a party to an arrangement with a third party that is in the
business of distributing software, under which you make payment to
the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement
or otherwise) that contradict the conditions of this License, they
do not excuse you from the conditions of this License.  If you
cannot convey a covered work so as to satisfy simultaneously your
obligations under this License and any other pertinent obligations,
then as a consequence you may not convey it at all.  For example,
if you agree to terms that obligate you to collect a royalty for
further conveying from those to whom you convey the Program, the
only way you could satisfy both those terms and this License would
be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a
single combined work, and to convey the resulting work.  The terms
of this License will continue to apply to the part which is the
covered work, but the special requirements of the GNU Affero General
Public License, section 13, concerning interaction through a network
will apply to the combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions
of the GNU General Public License from time to time.  Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of
the GNU General Public License, you may choose any version ever
published by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes
you to choose that version for the Program.
```

```
  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow
a later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT
NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND
PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE PROGRAM PROVE
DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR
OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR
CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING
BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE
OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE
PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER
OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make
it free software which everyone can redistribute and change under
these terms.

To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is
found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation, either version 3
    of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see
    <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.
```

```
  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the
appropriate parts of the General Public License.  Of course, your
program's commands might be different; for a GUI interface, you
would use an "about box".

  You should also get your employer (if you work as a programmer) or
school, if any, to sign a "copyright disclaimer" for the program,
if necessary. For more information on this, and how to apply and
follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your
program into proprietary programs.  If your program is a subroutine
library, you may consider it more useful to permit linking proprietary
applications with the library.  If this is what you want to do, use
the GNU Lesser General Public License instead of this License.  But
first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.
==

==
GCC RUNTIME LIBRARY EXCEPTION

Version 3.1, 31 March 2009

Copyright © 2009 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of
this license document, but changing it is not allowed.

This GCC Runtime Library Exception ("Exception") is an additional
permission under section 7 of the GNU General Public License, version
3 ("GPLv3"). It applies to a given file (the "Runtime Library")
that bears a notice placed by the copyright holder of the file
stating that the file is governed by GPLv3 along with this Exception.

When you use GCC to compile a program, GCC may combine portions of
certain GCC header files and runtime libraries with the compiled
program. The purpose of this Exception is to allow compilation of
non-GPL (including proprietary) programs to use, in this way, the
header files and runtime libraries covered by this Exception.

0. Definitions.

A file is an "Independent Module" if it either requires the Runtime
Library for execution after a Compilation Process, or makes use of
an interface provided by the Runtime Library, but is not otherwise
based on the Runtime Library.

"GCC" means a version of the GNU Compiler Collection, with or without
modifications, governed by version 3 (or a specified later version)
of the GNU General Public License (GPL) with the option of using
any subsequent versions published by the FSF.

"GPL-compatible Software" is software whose conditions of propagation,
modification and use would permit combination with GCC in accord
with the license of GCC.

"Target Code" refers to output from any compiler for a real or
virtual target processor architecture, in executable form or suitable
for input to an assembler, loader, linker and/or execution phase.
Notwithstanding that, Target Code does not include data in any
format that is used as a compiler intermediate representation, or
used for producing a compiler intermediate representation.

The "Compilation Process" transforms code entirely represented in
```

```
non-intermediate languages designed for human-written code, and/or
in Java Virtual Machine byte code, into Target Code. Thus, for
example, use of source code generators and preprocessors need not
be considered part of the Compilation Process, since the Compilation
Process can be understood as starting with the output of the
generators or preprocessors.

A Compilation Process is "Eligible" if it is done using GCC, alone
or with other GPL-compatible software, or if it is done without
using any work based on GCC. For example, using non-GPL-compatible
Software to optimize any GCC intermediate representations would not
qualify as an Eligible Compilation Process.

1. Grant of Additional Permission.

You have permission to propagate a work of Target Code formed by
combining the Runtime Library with Independent Modules, even if
such propagation would otherwise violate the terms of GPLv3, provided
that all Target Code was generated by Eligible Compilation Processes.
You may then convey such a combination under terms of your choice,
consistent with the licensing of the Independent Modules.

2. No Weakening of GCC Copyleft.

The availability of this Exception does not imply any general
presumption that third-party software is unaffected by the copyleft
requirements of the license of GCC.
==
```

A.15 GNU Lesser General Public License Version 2.1, February 1999

```

The following applies to all products licensed under the
GNU Lesser General Public License, Version 2.1: You may
not use the identified files except in compliance with
the GNU Lesser General Public License, Version 2.1 (the
"License"). You may obtain a copy of the License at
http://www.gnu.org/licenses/lgpl-2.1.html. A copy of the
license is also reproduced below. Unless required by
applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied. See the License for the specific language governing
permissions and limitations under the License.

                  GNU LESSER GENERAL PUBLIC LICENSE
                       Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
```

```
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

  To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library.  Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

  Finally, software patents pose a constant threat to the existence of
any free program.  We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder.  Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License.  This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License.  We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library.  The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom.  The Lesser General
Public License permits more lax criteria for linking other code with
the library.

  We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License.  It also provides other free software developers Less
of an advantage over competing non-free programs.  These disadvantages
are the reason we use the ordinary General Public License for many
libraries.  However, the Lesser license provides advantages in certain
special circumstances.

  For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it
becomes a de-facto standard.  To achieve this, non-free programs
must be allowed to use the library.  A more frequent case is that
a free library does the same job as widely used non-free libraries.
In this case, there is little to gain by limiting the free library
to free software only, so we use the Lesser General Public License.
```

```
  In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software.  For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

  Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

  The precise terms and conditions for copying, distribution and
modification follow.  Pay close attention to the difference between a
"work based on the library" and a "work that uses the library".  The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

                  GNU LESSER GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

  A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

  The "Library", below, refers to any such software library or work
which has been distributed under these terms.  A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language.  (Hereinafter, translation is
included without limitation in the term "modification".)

  "Source code" for a work means the preferred form of the work for
making modifications to it.  For a library, complete source code means
all the source code for all modules it contains, plus any associated
interface definition files, plus the scripts used to control
compilation and installation of the library.

  Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running a program using the Library is not restricted, and output from
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
and what the program that uses the Library does.

  1. You may copy and distribute verbatim copies of the Library's
complete source code as you receive it, in any medium, provided that
you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

  You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

  2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.
```

```
    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application.  Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library.  To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License.  (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.)  Do not make any other change in
these notices.

  Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.

  This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.

  4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.

  If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
distribute the source code, even though third parties are not
compelled to copy the source along with the object code.

  5. A program that contains no derivative of any portion of the
```

```
Library, but is designed to work with the Library by being compiled or
linked with it, is called a "work that uses the Library".  Such a
work, in isolation, is not a derivative work of the Library, and
therefore falls outside the scope of this License.

  However, linking a "work that uses the Library" with the Library
creates an executable that is a derivative of the Library (because it
contains portions of the Library), rather than a "work that uses the
library".  The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.

  When a "work that uses the Library" uses material from a header file
that is part of the Library, the object code for the work may be a
derivative work of the Library even though the source code is not.
Whether this is true is especially significant if the work can be
linked without the Library, or if the work is itself a library.  The
threshold for this to be true is not precisely defined by law.

  If such an object file uses only numerical parameters, data
structure layouts and accessors, and small macros and small inline
functions (ten lines or less in length), then the use of the object
file is unrestricted, regardless of whether it is legally a derivative
work.  (Executables containing this object code plus portions of the
Library will still fall under Section 6.)

  Otherwise, if the work is a derivative of the Library, you may
distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

  6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.

  You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License.  You must supply a copy of this License.  If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License.  Also, you must do one
of these things:

    a) Accompany the work with the complete corresponding
    machine-readable source code for the Library including whatever
    changes were used in the work (which must be distributed under
    Sections 1 and 2 above); and, if the work is an executable linked
    with the Library, with the complete machine-readable "work that
    uses the Library", as object code and/or source code, so that the
    user can modify the Library and then relink to produce a modified
    executable containing the modified Library.  (It is understood
    that the user who changes the contents of definitions files in the
    Library will not necessarily be able to recompile the application
    to use the modified definitions.)

    b) Use a suitable shared library mechanism for linking with the
    Library.  A suitable mechanism is one that (1) uses at run time a
    copy of the library already present on the user's computer system,
    rather than copying library functions into the executable, and (2)
    will operate properly with a modified version of the library, if
    the user installs one, as long as the modified version is
    interface-compatible with the version that the work was made with.

    c) Accompany the work with a written offer, valid for at
    least three years, to give the same user the materials
    specified in Subsection 6a, above, for a charge no more
    than the cost of performing this distribution.

    d) If distribution of the work is made by offering access to copy
    from a designated place, offer equivalent access to copy the above
```

```
    specified materials from the same place.

    e) Verify that the user has already received a copy of these
    materials or that you have already sent this user a copy.

  For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it.  However, as a special exception,
the materials to be distributed need not include anything that is
normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.

  It may happen that this requirement contradicts the license
restrictions of other proprietary libraries that do not normally
accompany the operating system.  Such a contradiction means you cannot
use both them and the Library together in an executable that you
distribute.

  7. You may place library facilities that are a work based on the
Library side-by-side in a single library together with other library
facilities not covered by this License, and distribute such a combined
library, provided that the separate distribution of the work based on
the Library and of the other library facilities is otherwise
permitted, and provided that you do these two things:

    a) Accompany the combined library with a copy of the same work
    based on the Library, uncombined with any other library
    facilities.  This must be distributed under the terms of the
    Sections above.

    b) Give prominent notice with the combined library of the fact
    that part of it is a work based on the Library, and explaining
    where to find the accompanying uncombined form of the same work.

  8. You may not copy, modify, sublicense, link with, or distribute
the Library except as expressly provided under this License.  Any
attempt otherwise to copy, modify, sublicense, link with, or
distribute the Library is void, and will automatically terminate your
rights under this License.  However, parties who have received copies,
or rights, from you under this License will not have their licenses
terminated so long as such parties remain in full compliance.

  9. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Library or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Library (or any work based on the
Library), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Library or works based on it.

  10. Each time you redistribute the Library (or any work based on the
Library), the recipient automatically receives a license from the
original licensor to copy, distribute, link with or modify the Library
subject to these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties with
this License.

  11. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all.  For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
```

```
refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended
to apply, and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number.  If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation.  If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

  14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission.  For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this.  Our
decision will be guided by the two goals of preserving the free status
of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

                            NO WARRANTY

  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU.  SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
```

```
DAMAGES.

                     END OF TERMS AND CONDITIONS

           How to Apply These Terms to Your New Libraries

  If you develop a new library, and you want it to be of the greatest
possible use to the public, we recommend making it free software that
everyone can redistribute and change.  You can do so by permitting
redistribution under these terms (or, alternatively, under the terms
of the ordinary General Public License).

  To apply these terms, attach the following notices to the library.
It is safest to attach them to the start of each source file to most
effectively convey the exclusion of warranty; and each file should
have at least the "copyright" line and a pointer to where the full
notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
    02110-1301  USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James
  Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!
```

A.16 GNU Libtool License

The following software may be included in this product:

GNU Libtool (The GNU Portable Library Tool)

```
If you are receiving a copy of the Oracle software in
source code, you are also receiving a copy of two files
(ltmain.sh and ltdl.h) generated by the GNU Libtool in
source code. If you received the Oracle software under
a license other than a commercial (non-GPL) license,
then the terms of the Oracle license do NOT apply to
these files from GNU Libtool; they are licensed under
the following licenses, separately from the Oracle
programs you receive.

Oracle elects to use GNU General Public License version
2 (GPL) for any software where a choice of GPL or GNU
Lesser/Library General Public License (LGPL) license
```

```
versions are made available with the language indicating
that GPL/LGPL or any later version may be used, or where
a choice of which version of the GPL/LGPL is applied is
unspecified.

From GNU Libtool:

ltmain.sh - Provide generalized library-building support
services.
NOTE: Changing this file will not affect anything until
you rerun configure.
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2003, 2004,
2005, 2006, 2007 Free Software Foundation, Inc.
Originally by Gordon Matzigkeit, 1996

This program is free software; you can redistribute it
and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation;
either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be
useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more
details. You should have received a copy of the GNU General
Public License along with this program; if not, write to
the Free Software Foundation, Inc., 51 Franklin Street,
Fifth Floor, Boston, MA 02110-1301, USA.

As a special exception to the GNU General Public License,
if you distribute this file as part of a program that
contains a configuration script generated by Autoconf,
you may include it under the same distribution terms that
you use for the rest of that program.
```

This component is licensed under Section A.13, "GNU General Public License Version 2.0, June 1991"

A.17 GNU Readline License

The following software may be included in this product:

GNU Readline Library

```
GNU Readline Library
With respect to MySQL Server/Cluster software licensed
under GNU General Public License, you are receiving a
copy of the GNU Readline Library in source code. The
terms of any Oracle license that might accompany the
Oracle programs do NOT apply to the GNU Readline Library;
it is licensed under the following license, separately
from the Oracle programs you receive. Oracle elects to
use GNU General Public License version 2 (GPL) for any
software where a choice of GPL license versions are
made available with the language indicating that GPLv2
or any later version may be used, or where a choice of
which version of the GPL is applied is unspecified.
```

This component is licensed under Section A.13, "GNU General Public License Version 2.0, June 1991"

A.18 GNU Standard C++ Library (libstdc++) License

The following software may be included in this product: GNU Standard C++ Library (libstdc++)

This component is licensed under Section A.14, "GNU General Public License Version 3.0, 29 June 2007 and GCC Runtime Library Exception Version 3.1, 31 March 2009".

Additional notices:

A.19 Google C++ Mocking Framework (Google Mock) License

This Oracle Product includes or references Gmock (including gtest), which is licensed to Oracle under the following terms:

A.20 Google Controlling Master Thread I/O Rate Patch License

The following software may be included in this product:

Google Controlling master thread I/O rate patch

```
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

A.21 Google Perftools (TCMalloc utility) License

The following software may be included in this product:

```
Google Perftools (TCMalloc utility)

Copyright (c) 1998-2006, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

    * Redistributions of source code must retain the above
      copyright notice, this list of conditions and the following
      disclaimer.
    * Redistributions in binary form must reproduce the above
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
      with the distribution.
    * Neither the name of Google Inc. nor the names of its
      contributors may be used to endorse or promote products
      derived from this software without specific prior written
      permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

A.22 Google SMP Patch License

The following software may be included in this product:

Google SMP Patch

```
Google SMP patch

Copyright (c) 2008, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
```

A.23 Janson License

The following software may be included in this product:

Janson 2.6

A.24 lib_sql.cc License

The following software may be included in this product:

`lib_sql.cc`

A.25 Libaio License

The following software may be included in this product:

```
libaio
```

This component is licensed under Section A.15, “GNU Lesser General Public License Version 2.1, February 1999”.

A.26 libevent License

The following software may be included in this product:

```
libevent

Copyright (c) 2000-2007 Niels Provos <provos@citi.umich.edu>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

==
Parts developed by Adam Langley
==

==
log.c
Based on err.c, which was adapted from OpenBSD libc *err*warncode.

Copyright (c) 2005 Nick Mathewson
Copyright (c) 2000 Dug Song
Copyright (c) 1993 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. Neither the name of the University nor the names of its
   contributors may be used to endorse or promote products derived
   from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
```

```
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
==

==
min_heap.h

Copyright (c) 2006 Maxim Yegorushkin
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==

==
win32.c

Copyright 2000-2002 Niels Provos
Copyright 2003 Michael A. Davis
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3.  The name of the author may not be used to endorse or promote
    products derived from this software without specific prior
    written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
```

```
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==
```

A.27 Linux-PAM License

The following software may be included in this product:

```

Linux-PAM (pam-devel, Pluggable authentication modules for Linux)

Copyright Theodore Ts'o, 1996. All rights reserved.

(For the avoidance of doubt, Oracle uses and distributes this
component under the terms below and elects not to do so under
the GPL even though the GPL is referenced as an option below.)

Redistribution and use in source and binary forms, with or
without modification, are permitted provided that the following
conditions are met:

1. Redistributions of source code must retain the above copyright
   notice, and the entire permission notice in its entirety,
   including the disclaimer of warranties.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

ALTERNATIVELY, this product may be distributed under the terms
of the GNU Public License, in which case the provisions of the
GPL are required INSTEAD OF the above restrictions. (This clause
is necessary due to a potential bad interaction between the GPL
and the restrictions contained in a BSD-style copyright.)

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
```

A.28 md5 (Message-Digest Algorithm 5) License

The following software may be included in this product:

```

md5 (Message-Digest Algorithm 5)

 This code implements the MD5 message-digest algorithm.
 The algorithm is due to Ron Rivest.  This code was
 written by Colin Plumb in 1993, no copyright is claimed.
 This code is in the public domain; do with it what you wish.

 Equivalent code is available from RSA Data Security, Inc.
 This code has been tested against that, and is equivalent,
 except that you don't need to include two pages of legalese
 with every copy.

 The code has been modified by Mikael Ronstroem to handle
 calculating a hash value of a key that is always a multiple
```

```
of 4 bytes long. Word 0 of the calculated 4-word hash value
is returned as the hash value.
```

A.29 memcached License

The following software may be included in this product:

memcached

```
Copyright (c) 2003, Danga Interactive, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

    * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.

    * Neither the name of the Danga Interactive nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

A.30 Memcached.pm License

The following software may be included in this product:

Memcached.pm

```
Memcached.pm is licensed under the Perl license.

Oracle may use, redistribute and/or modify this code under the terms of
either:

 a) the GNU General Public License as published by the Free Software
Foundation; either version 1, or (at your option) any later version, or

 b) the "Artistic License" which comes with the Expect/pr code.

Oracle elects to use the GPLv2 for version of MySQL that are licensed under
the GPL.

Oracle elects to use the Artistic license for all other (commercial) versions
of MySQL.

A copy of the GPLv2 and the Artistic License (Perl) 1.0 must be included with
any distribution.
```

This component is licensed under Section A.13, “GNU General Public License Version 2.0, June 1991”

This component is licensed under Section A.2, “Artistic License (Perl) 1.0”

A.31 mkpasswd.pl License

The following software may be included in this product:

mkpasswd.pl Perl module

```
Copyright (C) 2003-2004 by Chris Grau

This library is free software; you can redistribute it and/or modify it under
the same terms as Perl itself, either Perl version 5.8.1 or, at your option,
any later version of Perl 5 you may have available.

The Perl 5.8.1 license (from http://www.cpan.org/src/5.0/perl-5.8.1.tar.gz - main readme file):


    Perl Kit, Version 5

                Copyright (C) 1993, 1994, 1995, 1996, 1997, 1998
                   1999, 2000, 2001, by Larry Wall and others

    All rights reserved.

    This program is free software; you can redistribute it and/or modify
    it under the terms of either:

a) the GNU General Public License as published by the Free
Software Foundation; either version 1, or (at your option) any
later version, or

b) the "Artistic License" which comes with this Kit.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See either
    the GNU General Public License or the Artistic License for more details.

    You should have received a copy of the Artistic License with this
    Kit, in the file named "Artistic".  If not, I'll be glad to provide one.

    You should also have received a copy of the GNU General Public License
    along with this program in the file named "Copying". If not, write to the

    Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
    02111-1307, USA or visit their web page on the internet at
    http://www.gnu.org/copyleft/gpl.html.

    For those of you that choose to use the GNU General Public License,
    my interpretation of the GNU General Public License is that no Perl
    script falls under the terms of the GPL unless you explicitly put
    said script under the terms of the GPL yourself.  Furthermore, any
    object code linked with perl does not automatically fall under the
    terms of the GPL, provided such object code only adds definitions
    of subroutines and variables, and does not otherwise impair the
    resulting interpreter from executing any standard Perl script.  I
    consider linking in C subroutines in this manner to be the moral
    equivalent of defining subroutines in the Perl language itself.  You
    may sell such an object file as proprietary provided that you provide
    or offer to provide the Perl source, as specified by the GNU General
    Public License.  (This is merely an alternate way of specifying input
    to the program.)  You may also sell a binary produced by the dumping of
    a running Perl script that belongs to you, provided that you provide or
    offer to provide the Perl source as specified by the GPL.  (The
    fact that a Perl interpreter and your code are in the same binary file
    is, in this case, a form of mere aggregation.)  This is my interpretation
    of the GPL.  If you still have concerns or difficulties understanding
    my intent, feel free to contact me.  Of course, the Artistic License
    spells all this out for your protection, so you may prefer to use that.

--------------------------------------------------------------------------
```

```
Perl is a language that combines some of the features of C, sed, awk
and shell.  See the manual page for more hype.  There are also many Perl
books available, covering a wide variety of topics, from various publishers.
See pod/perlbook.pod for more information.

Please read all the directions below before you proceed any further, and
then follow them carefully.

After you have unpacked your kit, you should have all the files listed
in MANIFEST.

Installation

1) Detailed instructions are in the file "INSTALL", which you should
read if you are either installing on a system resembling Unix
or porting perl to another platform.  For non-Unix platforms, see the
corresponding README.

2) Read the manual entries before running perl.

3) IMPORTANT!  Help save the world!  Communicate any problems and suggested
patches to perlbug@perl.org so we can keep the world in sync.
If you have a problem, there's someone else out there who either has had
or will have the same problem.  It's usually helpful if you send the
output of the "myconfig" script in the main perl directory.

If you've succeeded in compiling perl, the perlbug script in the "utils"
subdirectory can be used to help mail in a bug report.

If possible, send in patches such that the patch program will apply them.
Context diffs are the best, then normal diffs.  Don't send ed scripts--
I've probably changed my copy since the version you have.

The latest versions of perl are always available on the various CPAN
(Comprehensive Perl Archive Network) sites around the world.
See <URL:http://www.cpan.org/src/>.


Just a personal note:  I want you to know that I create nice things like this
because it pleases the Author of my story.  If this bothers you, then your
notion of Authorship needs some revision.  But you can use perl anyway. :-)

The author.
===============================================

The "Artistic License"

Preamble

The intent of this document is to state the conditions under which a
Package may be copied, such that the Copyright Holder maintains some
semblance of artistic control over the development of the package,
while giving the users of the package the right to use and distribute
the Package in a more-or-less customary fashion, plus the right to make
reasonable modifications.

Definitions:

"Package" refers to the collection of files distributed by the
Copyright Holder, and derivatives of that collection of files
created through textual modification.

"Standard Version" refers to such a Package if it has not been
modified, or has been modified in accordance with the wishes
of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or
copyrights for the package.

"You" is you, if you're thinking about copying or distributing
this Package.
```

```
"Reasonable copying fee" is whatever you can justify on the
basis of media cost, duplication charges, time of people involved,
and so on.  (You will not be required to justify it to the
Copyright Holder, but only to the computing community at large
as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item
itself, though there may be fees involved in handling the item.
It also means that recipients of the item may redistribute it
under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the
Standard Version of this Package without restriction, provided that you
duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications
derived from the Public Domain or from the Copyright Holder.  A Package
modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided
that you insert a prominent notice in each changed file stating how and
when you changed that file, and provided that you do at least ONE of the
following:

    a) place your modifications in the Public Domain or otherwise make them
    Freely Available, such as by posting said modifications to Usenet or
    an equivalent medium, or placing the modifications on a major archive
    site such as uunet.uu.net, or by allowing the Copyright Holder to include
    your modifications in the Standard Version of the Package.

    b) use the modified Package only within your corporation or organization.

    c) rename any non-standard executables so the names do not conflict
    with standard executables, which must also be provided, and provide
    a separate manual page for each non-standard executable that clearly
    documents how it differs from the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or
executable form, provided that you do at least ONE of the following:

    a) distribute a Standard Version of the executables and library files,
    together with instructions (in the manual page or equivalent) on where
    to get the Standard Version.

    b) accompany the distribution with the machine-readable source of
    the Package with your modifications.

    c) give non-standard executables non-standard names, and clearly
    document the differences in manual pages (or equivalent), together
    with instructions on where to get the Standard Version.

    d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this
Package.  You may charge any fee you choose for support of this
Package.  You may not charge a fee for this Package itself.  However,
you may distribute this Package in aggregate with other (possibly
commercial) programs as part of a larger (possibly commercial) software
distribution provided that you do not advertise this Package as a
product of your own.  You may embed this Package's interpreter within
an executable of yours (by linking); this shall be construed as a mere
form of aggregation, provided that the complete Standard Version of the
interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as
output from the programs of this Package do not automatically fall
under the copyright of this Package, but belong to whoever generated
them, and may be sold commercially, and may be aggregated with this
Package.  If such scripts or library files are aggregated with this
Package via the so-called "undump" or "unexec" methods of producing a
```

```
binary executable image, then distribution of such an image shall
neither be construed as a distribution of this Package nor shall it
fall under the restrictions of Paragraphs 3 and 4, provided that you do
not represent such an executable image as a Standard Version of this
Package.

7. C subroutines (or comparably compiled subroutines in other
languages) supplied by you and linked into this Package in order to
emulate subroutines and variables of the language defined by this
Package shall not be considered part of this Package, but are the
equivalent of input as in Paragraph 6, provided these subroutines do
not change the language in any way that would cause it to fail the
regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always
permitted provided that the use of this Package is embedded; that is,
when no overt attempt is made to make this Package's interfaces visible
to the end user of the commercial distribution.  Such use shall not be
construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote
products derived from this software without specific prior written
permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End
```

A.32 node-mysql License

The following software may be included in this product:

```
Copyright (c) 2012  Felix Geisendörfer (felix@debuggable.com) and contributors

 Permission is hereby granted, free of charge, to any person obtaining a copy
 of this software and associated documentation files (the "Software"), to deal
 in the Software without restriction, including without limitation the rights
 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 copies of the Software, and to permit persons to whom the Software is
 furnished to do so, subject to the following conditions:

 The above copyright notice and this permission notice shall be included in
 all copies or substantial portions of the Software.

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 THE SOFTWARE.
```

A.33 Node.js License

The following software may be included in this product:

```
Copyright Joyent, Inc. and other Node contributors. All rights reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to
deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE SOFTWARE.

====

This license applies to all parts of Node that are not externally
maintained libraries. The externally maintained libraries used by Node are:

- V8, located at deps/v8. V8's license follows:
  """
    This license applies to all parts of V8 that are not externally
    maintained libraries.  The externally maintained libraries used by V8
    are:

      - PCRE test suite, located in
        test/mjsunit/third_party/regexp-pcre.js.  This is based on the
        test suite from PCRE-7.3, which is copyrighted by the University
        of Cambridge and Google, Inc.  The copyright notice and license
        are embedded in regexp-pcre.js.

      - Layout tests, located in test/mjsunit/third_party.  These are
        based on layout tests from webkit.org which are copyrighted by
        Apple Computer, Inc. and released under a 3-clause BSD license.

      - Strongtalk assembler, the basis of the files assembler-arm-inl.h,
        assembler-arm.cc, assembler-arm.h, assembler-ia32-inl.h,
        assembler-ia32.cc, assembler-ia32.h, assembler-x64-inl.h,
        assembler-x64.cc, assembler-x64.h, assembler-mips-inl.h,
        assembler-mips.cc, assembler-mips.h, assembler.cc and assembler.h.
        This code is copyrighted by Sun Microsystems Inc. and released
        under a 3-clause BSD license.

      - Valgrind client API header, located at third_party/valgrind/valgrind.h
        This is release under the BSD license.

    These libraries have their own licenses; we recommend you read them,
    as their terms may differ from the terms below.

    Copyright 2006-2012, the V8 project authors. All rights reserved.
    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are
    met:

        * Redistributions of source code must retain the above copyright
          notice, this list of conditions and the following disclaimer.
        * Redistributions in binary form must reproduce the above
          copyright notice, this list of conditions and the following
          disclaimer in the documentation and/or other materials provided
          with the distribution.
        * Neither the name of Google Inc. nor the names of its
          contributors may be used to endorse or promote products derived
          from this software without specific prior written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  """

- libev, located at deps/uv/src/unix/ev. libev's license follows:
```

```
    The above copyright notice and this permission notice shall be included in
    all copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
    IN THE SOFTWARE.
  """

- tools/cpplint.py is a C++ linter. Its license follows:
  """
    # Copyright (c) 2009 Google Inc. All rights reserved.
    #
    # Redistribution and use in source and binary forms, with or without
    # modification, are permitted provided that the following conditions are
    # met:
    #
    #    * Redistributions of source code must retain the above copyright
    # notice, this list of conditions and the following disclaimer.
    #    * Redistributions in binary form must reproduce the above
    # copyright notice, this list of conditions and the following disclaimer
    # in the documentation and/or other materials provided with the
    # distribution.
    #    * Neither the name of Google Inc. nor the names of its
    # contributors may be used to endorse or promote products derived from
    # this software without specific prior written permission.
    #
    # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    # "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    # A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
    # OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    # SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    # LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    # DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    # THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    # (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    # OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  """

- lib/buffer_ieee754.js. Its license follows:
  """
    // Copyright (c) 2008, Fair Oaks Labs, Inc.
    // All rights reserved.
    //
    // Redistribution and use in source and binary forms, with or without
    // modification, are permitted provided that the following conditions are met:
    //
    //  * Redistributions of source code must retain the above copyright notice,
    //    this list of conditions and the following disclaimer.
    //
    //  * Redistributions in binary form must reproduce the above copyright notice,
    //    this list of conditions and the following disclaimer in the documentation
    //    and/or other materials provided with the distribution.
    //
    //  * Neither the name of Fair Oaks Labs, Inc. nor the names of its contributors
    //    may be used to endorse or promote products derived from this software
    //    without specific prior written permission.
    //
    // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
    // AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    // IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
    // ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
    // LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
    // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
    // SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
    // INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
    // CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
    // ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
```

```
    // POSSIBILITY OF SUCH DAMAGE.
  """

- lib/punycode.js is copyright 2011 Mathias Bynens <http://mathiasbynens.be/>
  and released under the MIT license.
  """
    * Punycode.js <http://mths.be/punycode>
    * Copyright 2011 Mathias Bynens <http://mathiasbynens.be/>
    * Available under MIT license <http://mths.be/mit>
  """

- tools/gyp GYP is a meta-build system. GYP's license follows:
  """
    Copyright (c) 2009 Google Inc. All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, are permitted provided that the following conditions are
    met:

       * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
       * Redistributions in binary form must reproduce the above
    copyright notice, this list of conditions and the following disclaimer
    in the documentation and/or other materials provided with the
    distribution.
       * Neither the name of Google Inc. nor the names of its
    contributors may be used to endorse or promote products derived from
    this software without specific prior written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
    "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
    A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
    OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
    SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
    LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
    DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
    THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
    (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  """

- Zlib at deps/zlib. zlib's license follows
  """
    /* zlib.h -- interface of the 'zlib' general purpose compression library
      version 1.2.4, March 14th, 2010

      Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

      This software is provided 'as-is', without any express or implied
      warranty.  In no event will the authors be held liable for any damages
      arising from the use of this software.

      Permission is granted to anyone to use this software for any purpose,
      including commercial applications, and to alter it and redistribute it
      freely, subject to the following restrictions:

      1. The origin of this software must not be misrepresented; you must not
         claim that you wrote the original software. If you use this software
         in a product, an acknowledgment in the product documentation would be
         appreciated but is not required.
      2. Altered source versions must be plainly marked as such, and must not be
         misrepresented as being the original software.
      3. This notice may not be removed or altered from any source distribution.

      Jean-loup Gailly
      Mark Adler

    */
  """

- tools/doc/node_modules/marked Marked is a Markdown parser. Marked's
```

```
  license follows
  """
@     Copyright (c) 2011-2012, Christopher Jeffrey (https://github.com/chjj/)

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in
    all copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
    THE SOFTWARE.
  """
```

A.34 nt_servc (Windows NT Service class library) License

The following software may be included in this product:

nt_servc (Windows NT Service class library)

```
Windows NT Service class library
Copyright Abandoned 1998 Irena Pancirov - Irnet Snc
This file is public domain and comes with NO WARRANTY of any kind
```

A.35 OpenPAM License

The following software may be included in this product:

OpenPAM

```
Copyright (c) 2002-2003 Networks Associates Technology, Inc.
Copyright (c) 2004-2007 Dag-Erling Smørgrav
All rights reserved.

This software was developed for the FreeBSD Project by
ThinkSec AS and Network Associates Laboratories, the
Security Research Division of Network Associates, Inc.
under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"),
as part of the DARPA CHATS research program.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the
   following disclaimer.
2. Redistributions in binary form must reproduce the
   above copyright notice, this list of conditions and
   the following disclaimer in the documentation and/or
   other materials provided with the distribution.
3. The name of the author may not be used to endorse or
   promote products derived from this software without
   specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN
```

```
NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

A.36 OpenSSL v1.0 License

The following software may be included in this product:

OpenSSL v1.0

```
LICENSE ISSUES
==============
The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
the OpenSSL License and the original SSLeay license apply to the toolkit. See
below for the actual license texts. Actually both licenses are BSD-style Open
Source licenses. In case of any license issues related to OpenSSL please
contact openssl-core@openssl.org.

OpenSSL License
---------------
/ ====================================================================
Copyright (c) 1998-2008 The OpenSSL Project.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must
display the following acknowledgment:  "This product includes software
developed by the OpenSSL Project for use in the OpenSSL Toolkit. (Link1 /)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
endorse or promote products derived from this software without prior written
permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may
"OpenSSL" appear in their names without prior written permission of the
OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following
acknowledgment:  "This product includes software developed by the OpenSSL
Project for use in the OpenSSL Toolkit (Link2 /)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY  EXPRESSED
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR  PURPOSE ARE DISCLAIMED. IN
NO EVENT SHALL THE OpenSSL PROJECT OR  ITS CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL,  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT  NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)  HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,  STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE)  ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED  OF THE POSSIBILITY OF SUCH DAMAGE.
====================================================================
This product includes cryptographic software written by Eric Young
(eay@cryptsoft.com). This product includes software written by Tim Hudson
(tjh@cryptsoft.com).

Original SSLeay License
-----------------------
/ Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
All rights reserved.
This package is an SSL implementation written by Eric Young
```

```
(eay@cryptsoft.com).  The implementation was written so as to conform with
Netscapes SSL.   This library is free for commercial and non-commercial use
as long as the following conditions are aheared to. The following conditions
apply to all code found in this distribution, be it the RC4, RSA,  lhash,
DES, etc., code; not just the SSL code. The SSL documentation included with
this distribution is covered by the same copyright terms except that the
holder is Tim Hudson (tjh@cryptsoft.com).   Copyright remains Eric Young's,
and as such any Copyright notices in the code are not to be removed.  If this
package is used in a product, Eric Young should be given attribution as the
author of the parts of the library used.  This can be in the form of a
textual message at program startup or in documentation (online or textual)
provided with the package.   Redistribution and use in source and binary
forms, with or without modification, are permitted provided that the
following conditions are met:  1. Redistributions of source code must retain
the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.  3. All advertising
materials mentioning features or use of this software must display the
following acknowledgement:  "This product includes cryptographic software
written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be
left out if the routines from the library being used are not cryptographic
related :-).  4. If you include any Windows specific code (or a derivative
thereof) from the apps directory (application code) you must include an
acknowledgement:  "This product includes software written by Tim Hudson
(tjh@cryptsoft.com)"   THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  ARE
DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE  FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL  DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS  OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)  HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT  LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY  OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF  SUCH DAMAGE.   The
license and distribution terms for any publically available version or
derivative of this code cannot be changed. i.e. this code cannot simply be
copied and put under another distribution license  [including the GNU Public
License.]
```

A.37 Paramiko License

The following software may be included in this product:

```
Paramiko

You are receiving a copy of Paramiko in both source and object code. The
terms of the Oracle license do NOT apply to the Paramiko program; it is
licensed under the following license, separately from the Oracle
programs you receive. If you do not wish to install this program, you
may delete the Paramiko folder and all its contents.
```

This component is licensed under Section A.15, “GNU Lesser General Public License Version 2.1, February 1999”.

A.38 Percona Multiple I/O Threads Patch License

The following software may be included in this product:

Percona Multiple I/O threads patch

```
Copyright (c) 2008, 2009 Percona Inc
All rights reserved.

Redistribution and use of this software in source and binary forms,
with or without modification, are permitted provided that the
following conditions are met:
```

```
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of Percona Inc. nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission of Percona Inc.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

A.39 Pion License

This Oracle Product includes or references Pion which is licensed to Oracle under the following terms:

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

A.40 Python License

The following software may be included in this product:

```
Python Programming Language

This is the official license for the Python 2.7 release:

A. HISTORY OF THE SOFTWARE
Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands
as a successor of a language called ABC.  Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see http://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.
```

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

Release	Derived from	Year	Owner	GPL-compatible? (1)
0.9.0 thru 1.2		1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	yes (2)
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.2	2.1.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2.1	2.2	2002	PSF	yes
2.2.2	2.2.1	2002	PSF	yes
2.2.3	2.2.2	2003	PSF	yes
2.3	2.2.2	2002-2003	PSF	yes
2.3.1	2.3	2002-2003	PSF	yes
2.3.2	2.3.1	2002-2003	PSF	yes
2.3.3	2.3.2	2002-2003	PSF	yes
2.3.4	2.3.3	2004	PSF	yes
2.3.5	2.3.4	2005	PSF	yes
2.4	2.3	2004	PSF	yes
2.4.1	2.4	2005	PSF	yes
2.4.2	2.4.1	2005	PSF	yes
2.4.3	2.4.2	2006	PSF	yes
2.5	2.4	2006	PSF	yes
2.5.1	2.5	2007	PSF	yes
2.5.2	2.5.1	2008	PSF	yes
2.5.3	2.5.2	2008	PSF	yes
2.6	2.5	2008	PSF	yes
2.6.1	2.6	2008	PSF	yes
2.6.2	2.6.1	2009	PSF	yes
2.6.3	2.6.2	2009	PSF	yes
2.6.4	2.6.3	2010	PSF	yes
2.7	2.6	2010	PSF	yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's direction to make these releases possible.

B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

```
-------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python
alone or in any derivative version, provided, however, that PSF's
License Agreement and PSF's notice of copyright, i.e., "Copyright (c)
2001, 2002, 2003, 2004, 2005, 2006 Python Software Foundation; All Rights
Reserved" are retained in Python alone or in any derivative version
prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS"
basis.  PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee.  This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.


BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0
-------------------------------------------

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an
office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the
Individual or Organization ("Licensee") accessing and otherwise using
this software in source or binary form and its associated
documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License
Agreement, BeOpen hereby grants Licensee a non-exclusive,
royalty-free, world-wide license to reproduce, analyze, test, perform
and/or display publicly, prepare derivative works, distribute, and
otherwise use the Software alone or in any derivative version,
provided, however, that the BeOpen Python License is retained in the
Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS"
basis.  BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
```

```
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE
SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS
AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY
DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all
respects by the law of the State of California, excluding conflict of
law provisions.  Nothing in this License Agreement shall be deemed to
create any relationship of agency, partnership, or joint venture
between BeOpen and Licensee.  This License Agreement does not grant
permission to use BeOpen trademarks or trade names in a trademark
sense to endorse or promote products or services of Licensee, or any
third party.  As an exception, the "BeOpen Python" logos available at
http://www.pythonlabs.com/logos.html may be used according to the
permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.


CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1
---------------------------------------

1. This LICENSE AGREEMENT is between the Corporation for National
Research Initiatives, having an office at 1895 Preston White Drive,
Reston, VA 20191 ("CNRI"), and the Individual or Organization
("Licensee") accessing and otherwise using Python 1.6.1 software in
source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python 1.6.1
alone or in any derivative version, provided, however, that CNRI's
License Agreement and CNRI's notice of copyright, i.e., "Copyright (c)
1995-2001 Corporation for National Research Initiatives; All Rights
Reserved" are retained in Python 1.6.1 alone or in any derivative
version prepared by Licensee.  Alternately, in lieu of CNRI's License
Agreement, Licensee may substitute the following text (omitting the
quotes): "Python 1.6.1 is made available subject to the terms and
conditions in CNRI's License Agreement.  This Agreement together with
Python 1.6.1 may be located on the Internet using the following
unique, persistent identifier (known as a handle): 1895.22/1013.  This
Agreement may also be obtained from a proxy server on the Internet
using the following URL: http://hdl.handle.net/1895.22/1013".

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python 1.6.1 or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS"
basis.  CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
```

```
breach of its terms and conditions.

7. This License Agreement shall be governed by the federal
intellectual property law of the United States, including without
limitation the federal copyright law, and, to the extent such
U.S. federal law does not apply, by the law of the Commonwealth of
Virginia, excluding Virginia's conflict of law provisions.
Notwithstanding the foregoing, with regard to derivative works based
on Python 1.6.1 that incorporate non-separable material that was
previously distributed under the GNU General Public License (GPL), the
law of the Commonwealth of Virginia shall govern this License
Agreement only as to issues arising under or with respect to
Paragraphs 4, 5, and 7 of this License Agreement.  Nothing in this
License Agreement shall be deemed to create any relationship of
agency, partnership, or joint venture between CNRI and Licensee.  This
License Agreement does not grant permission to use CNRI trademarks or
trade name in a trademark sense to endorse or promote products or
services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying,
installing or otherwise using Python 1.6.1, Licensee agrees to be
bound by the terms and conditions of this License Agreement.

        ACCEPT


CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2
--------------------------------------------------

Copyright (c) 1991 - 1995, Stichting Mathematisch Centrum Amsterdam,
The Netherlands.  All rights reserved.

Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Stichting Mathematisch
Centrum or CWI not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Licenses and Acknowledgements for Incorporated Software
========================================================

This section is an incomplete, but growing list of licenses and acknowledgements
for third-party software incorporated in the Python distribution.

Mersenne Twister
================
The _random module includes code based on a download from
  http://www.math.keio.ac.jp/ matumoto/MT2002/emt19937ar.html.
The following are the verbatim comments from the original code:

A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using init_genrand(seed)
or init_by_array(init_key, key_length).

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
```

```
are met:

 1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

 2. Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the
    distribution.

 3. The names of its contributors may not be used to endorse or promote
    products derived from this software without specific prior written
    permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Any feedback is very welcome.
http://www.math.keio.ac.jp/matumoto/emt.html
email: matumoto@math.keio.ac.jp

Sockets
=======
The socket module uses the functions, getaddrinfo(), and getnameinfo(), which
are coded in separate source files from the WIDE Project, http://www.wide.ad.jp/.

Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors
   may be used to endorse or promote products derived from this
   software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Floating point exception control
================================
The source for the fpectl module includes the following notice:

  ---------------------------------------------------------------------
 /                       Copyright (c) 1996.                           \
|          The Regents of the University of California.                 |
|                        All rights reserved.                           |
|                                                                       |
|   Permission to use, copy, modify, and distribute this software for   |
```

```
|   any purpose without fee is hereby granted, provided that this en-   |
|   tire notice is included in all copies of any software which is or  |
|   includes  a  copy  or  modification  of  this software and in all   |
|   copies of the supporting documentation for such software.          |
|                                                                       |
|   This  work was produced at the University of California, Lawrence  |
|   Livermore National Laboratory under  contract  no.  W-7405-ENG-48  |
|   between  the  U.S.  Department  of  Energy and The Regents of the   |
|   University of California for the operation of UC LLNL.             |
|                                                                       |
|                              DISCLAIMER                               |
|                                                                       |
|   This  software was prepared as an account of work sponsored by an  |
|   agency of the United States Government. Neither the United States  |
|   Government  nor the University of California nor any of their em-  |
|   ployees, makes any warranty, express or implied, or  assumes  any   |
|   liability  or  responsibility  for the accuracy, completeness, or   |
|   usefulness of any information,  apparatus,  product,  or  process   |
|   disclosed,   or  represents  that  its  use  would  not  infringe   |
|   privately-owned rights. Reference herein to any specific  commer-   |
|   cial  products,  process,  or  service  by trade name, trademark,   |
|   manufacturer, or otherwise, does not  necessarily  constitute  or   |
|   imply  its endorsement, recommendation, or favoring by the United   |
|   States Government or the University of California. The views  and  |
|   opinions  of authors expressed herein do not necessarily state or   |
|   reflect those of the United States Government or  the  University   |
|   of  California,  and shall not be used for advertising or product   |
 \  endorsement purposes.                                              /
  ---------------------------------------------------------------------

MD5 message digest algorithm
============================
The source code for the md5 module contains the following notice:

Copyright (C) 1999, 2002 Aladdin Enterprises.  All rights reserved.

This software is provided 'as-is', without any express or implied
warranty.  In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would
   be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not
   be misrepresented as being the original software.
3. This notice may not be removed or altered from any source
   distribution.

L. Peter Deutsch
ghost@aladdin.com

Independent implementation of MD5 (RFC 1321).

This code implements the MD5 Algorithm defined in RFC 1321, whose
text is available at
      http://www.ietf.org/rfc/rfc1321.txt
The code is derived from the text of the RFC, including the test suite
(section A.5) but excluding the rest of Appendix A.  It does not include
any code or documentation that is identified in the RFC as being
copyrighted.

The original and principal author of md5.h is L. Peter Deutsch
<ghost@aladdin.com>.  Other authors are noted in the change history
that follows (in reverse chronological order):

2002-04-13 lpd Removed support for non-ANSI compilers; removed
      references to Ghostscript; clarified derivation from RFC 1321;
```

```
      now handles byte order either statically or dynamically.
1999-11-04 lpd Edited comments slightly for automatic TOC extraction.
1999-10-18 lpd Fixed typo in header comment (ansi2knr rather than md5);
      added conditionalization for C++ compilation from Martin
      Purschke <purschke@bnl.gov>.
1999-05-03 lpd Original version.

Asynchronous socket services
============================
The asynchat and asyncore modules contain the following notice:

Copyright 1996 by Sam Rushing

                        All Rights Reserved

Permission to use, copy, modify, and distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of Sam
Rushing not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

SAM RUSHING DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN
NO EVENT SHALL SAM RUSHING BE LIABLE FOR ANY SPECIAL, INDIRECT OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Cookie management
=================
The Cookie module contains the following notice:

Copyright 2000 by Timothy O'Malley <timo@alum.mit.edu>

               All Rights Reserved

Permission to use, copy, modify, and distribute this software
and its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Timothy O'Malley  not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS, IN NO EVENT SHALL Timothy O'Malley BE LIABLE FOR
ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.

Profiling
=========
The profile and pstats modules contain the following notice:

Copyright 1994, by InfoSeek Corporation, all rights reserved.
Written by James Roskind

Permission to use, copy, modify, and distribute this Python software
and its associated documentation for any purpose (subject to the
restriction in the following sentence) without fee is hereby granted,
provided that the above copyright notice appears in all copies, and
that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of InfoSeek not be used in
advertising or publicity pertaining to distribution of the software
```

```
without specific, written prior permission.  This permission is
explicitly restricted to the copying and modification of the software
to remain in Python, compiled Python, or other languages (such as C)
wherein the modified or derived code is exclusively imported into a
Python module.

INFOSEEK CORPORATION DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL INFOSEEK CORPORATION BE LIABLE FOR ANY
SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER
RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF
CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Execution tracing
=================
The trace module contains the following notice:

portions copyright 2001, Autonomous Zones Industries, Inc., all rights...
err...  reserved and offered to the public under the terms of the
Python 2.2 license.
Author: Zooko O'Whielacronx
http://zooko.com/
mailto:zooko@zooko.com

Copyright 2000, Mojam Media, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1999, Bioreason, Inc., all rights reserved.
Author: Andrew Dalke

Copyright 1995-1997, Automatrix, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1991-1995, Stichting Mathematisch Centrum, all rights reserved.

Permission to use, copy, modify, and distribute this Python software and
its associated documentation for any purpose without fee is hereby
granted, provided that the above copyright notice appears in all copies,
and that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of neither Automatrix,
Bioreason or Mojam Media be used in advertising or publicity pertaining to
distribution of the software without specific, written prior permission.

UUencode and UUdecode functions
===============================

The uu module contains the following notice:

Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.
                       All Rights Reserved
Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Ellinghouse
not be used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:
- Use binascii module to do the actual line-by-line conversion
  between ascii and binary. This results in a 1000-fold speedup. The C
  version is still 5 times faster, though.
- Arguments more compliant with Python standard
```

```
XML Remote Procedure Calls¶

The xmlrpclib module contains the following notice:

    The XML-RPC client interface is

Copyright (c) 1999-2002 by Secret Labs AB
Copyright (c) 1999-2002 by Fredrik Lundh

By obtaining, using, and/or copying this software and/or its
associated documentation, you agree that you have read, understood,
and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and
its associated documentation for any purpose and without fee is
hereby granted, provided that the above copyright notice appears in
all copies, and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Secret Labs AB or the author not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANT-
ABILITY AND FITNESS.  IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR
BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY
DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
OF THIS SOFTWARE.

test_epoll
==========
The test_epoll contains the following notice:

Copyright (c) 2001-2006 Twisted Matrix Laboratories.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Select kqueue
=============
The select and contains the following notice for the kqueue interface:

Copyright (c) 2000 Doug White, 2006 James Knight, 2007 Christian Heimes
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
```

```
THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

strtod and dtoa
===============
The file Python/dtoa.c, which supplies C functions dtoa and strtod for conversion
of C doubles to and from strings, is derived from the file of the same name by
David M. Gay, currently available from http://www.netlib.org/fp/. The original
file, as retrieved on March 16, 2009, contains the following copyright and
licensing notice:

/****************************************************************
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for
 * any purpose without fee is hereby granted, provided that this entire
 * notice is included in all copies of any software which is or
 * includes a copy or modification of this software and in all copies
 * of the supporting documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTY.  IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT
 * MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE
 * MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR
 * PURPOSE.
 *
 ***************************************************************/
```

A.41 Red HAT RPM Spec File License

The following software may be included in this product:

Red Hat RPM Spec File

```
You are receiving a copy of the Red Hat spec file. The terms of the Oracle
license do NOT apply to the Red Hat spec file; it is licensed under the
following license, separately from the Oracle programs you receive.

                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
[for rest of text, see following link]
```

This component is licensed under Section A.13, “GNU General Public License Version 2.0, June 1991”

A.42 RegEX-Spencer Library License

The following software may be included in this product: Henry Spencer's Regular-Expression Library (RegEX-Spencer)

```
Copyright 1992, 1993, 1994 Henry Spencer.  All rights reserved.
```

A.43 Richard A. O'Keefe String Library License

The following software may be included in this product:

Richard A. O'Keefe String Library

A.44 sajson License

The following software may be included in this product:

sajson

A.45 SHA-1 in C License

The following software may be included in this product:

SHA-1 in C

```
SHA-1 in C
By Steve Reid <steve@edmweb.com>
100% Public Domain
```

A.46 Unicode Data Files

The following software may be included in this product:

Unicode Data Files

A.47 V8 License

The following software may be included in this product:

```
This license applies to all parts of V8 that are not externally
maintained libraries.  The externally maintained libraries used by V8
are:
.
```

```
  - PCRE test suite, located in
    test/mjsunit/third_party/regexp-pcre.js.  This is based on the
    test suite from PCRE-7.3, which is copyrighted by the University
    of Cambridge and Google, Inc.  The copyright notice and license
    are embedded in regexp-pcre.js.
.
  - Layout tests, located in test/mjsunit/third_party.  These are
    based on layout tests from webkit.org which are copyrighted by
    Apple Computer, Inc. and released under a 3-clause BSD license.
.
  - Strongtalk assembler, the basis of the files assembler-arm-inl.h,
    assembler-arm.cc, assembler-arm.h, assembler-ia32-inl.h,
    assembler-ia32.cc, assembler-ia32.h, assembler-x64-inl.h,
    assembler-x64.cc, assembler-x64.h, assembler-mips-inl.h,
    assembler-mips.cc, assembler-mips.h, assembler.cc and assembler.h.
    This code is copyrighted by Sun Microsystems Inc. and released
    under a 3-clause BSD license.
.
  - Valgrind client API header, located at third_party/valgrind/valgrind.h
    This is release under the BSD license.
.
These libraries have their own licenses; we recommend you read them,
as their terms may differ from the terms below.
.
Copyright 2006-2012, the V8 project authors. All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
.
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above
      copyright notice, this list of conditions and the following
      disclaimer in the documentation and/or other materials provided
      with the distribution.
    * Neither the name of Google Inc. nor the names of its
      contributors may be used to endorse or promote products derived
      from this software without specific prior written permission.
.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
.
=======================================================================
.
License for  PCRE test suite (regexp-pcre.js):
// PCRE LICENCE
// ------------
//
// PCRE is a library of functions to support regular expressions whose syntax
// and semantics are as close as possible to those of the Perl 5 language.
//
// Release 7 of PCRE is distributed under the terms of the "BSD" licence, as
// specified below. The documentation for PCRE, supplied in the "doc"
// directory, is distributed under the same terms as the software itself.
//
// The basic library functions are written in C and are freestanding. Also
// included in the distribution is a set of C++ wrapper functions.
//
// THE BASIC LIBRARY FUNCTIONS
// ---------------------------
//
// Written by:       Philip Hazel
// Email local part: ph10
```

```
// Email domain:     cam.ac.uk
//
// University of Cambridge Computing Service,
// Cambridge, England.
//
// Copyright (c) 1997-2007 University of Cambridge
// All rights reserved.
//
//
// THE C++ WRAPPER FUNCTIONS
// -------------------------
//
// Contributed by:   Google Inc.
//
// Copyright (c) 2007, Google Inc.
// All rights reserved.
//
//
// THE "BSD" LICENCE
// -----------------
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are met:
//
//     * Redistributions of source code must retain the above copyright notice,
//       this list of conditions and the following disclaimer.
//
//     * Redistributions in binary form must reproduce the above copyright
//       notice, this list of conditions and the following disclaimer in the
//       documentation and/or other materials provided with the distribution.
//
//     * Neither the name of the University of Cambridge nor the name of Google
//       Inc. nor the names of their contributors may be used to endorse or
//       promote products derived from this software without specific prior
//       written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
// AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
// ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
// LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
// CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
// SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
// INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
// CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
// ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
// POSSIBILITY OF SUCH DAMAGE.
//
// End
.
=======================================================================
 License for layout test files (array-isarray.js, array-splice-webkit.js,
object-keys.js and string-trim.js):
.
// Copyright (c) 2006-2009 Apple Computer, Inc. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//
// 1. Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//
// 2. Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following
// disclaimer in the documentation and/or other materials provided
// with the distribution.
//
// 3. Neither the name of the copyright holder(s) nor the names of any
// contributors may be used to endorse or promote products derived
// from this software without specific prior written permission.
//
```

A.48 zlib License

The following software may be included in this product:

zlib

Oracle gratefully acknowledges the contributions of Jean-loup Gailly and Mark Adler in creating the zlib general purpose compression library which is used in this product.

Appendix B MySQL 5.6 FAQ: MySQL Cluster

In the following section, we answer questions that are frequently asked about MySQL Cluster and the NDB storage engine.

Questions

- B.1: [572] Which versions of the MySQL software support Cluster? Do I have to compile from source?
- B.2: [573] What do “NDB” and “NDBCLUSTER” mean?
- B.3: [573] What is the difference between using MySQL Cluster versus using MySQL Replication?
- B.4: [573] Do I need any special networking to run MySQL Cluster? How do computers in a cluster communicate?
- B.5: [573] How many computers do I need to run a MySQL Cluster, and why?
- B.6: [573] What do the different computers do in a MySQL Cluster?
- B.7: [574] When I run the SHOW command in the MySQL Cluster management client, I see a line of output that looks like this:

  ```
  id=2    @10.100.10.32  (Version: 5.6.27-ndb-7.3.12 Nodegroup: 0, *)
  ```

  What does the * mean? How is this node different from the others?
- B.8: [575] With which operating systems can I use MySQL Cluster?
- B.9: [575] What are the hardware requirements for running MySQL Cluster?
- B.10: [575] How much RAM do I need to use MySQL Cluster? Is it possible to use disk memory at all?
- B.11: [576] What file systems can I use with MySQL Cluster? What about network file systems or network shares?
- B.12: [576] Can I run MySQL Cluster nodes inside virtual machines (such as those created by VMWare, Parallels, or Xen)?
- B.13: [576] I am trying to populate a MySQL Cluster database. The loading process terminates prematurely and I get an error message like this one: `ERROR 1114: The table 'my_cluster_table' is full` Why is this happening?
- B.14: [577] MySQL Cluster uses TCP/IP. Does this mean that I can run it over the Internet, with one or more nodes in remote locations?
- B.15: [577] Do I have to learn a new programming or query language to use MySQL Cluster?
- B.16: [577] What programming languages and APIs are supported by MySQL Cluster?
- B.17: [578] Does MySQL Cluster include any management tools?
- B.18: [578] How do I find out what an error or warning message means when using MySQL Cluster?
- B.19: [578] Is MySQL Cluster transaction-safe? What isolation levels are supported?
- B.20: [578] What storage engines are supported by MySQL Cluster?
- B.21: [578] In the event of a catastrophic failure—say, for instance, the whole city loses power *and* my UPS fails—would I lose all my data?

Questions and Answers

B.1: Which versions of the MySQL software support Cluster? Do I have to compile from source?

MySQL Cluster is not supported in standard MySQL Server 5.6 releases. Instead, MySQL Cluster is provided as a separate product. Currently, the following MySQL Cluster release series are available for production use:

- **MySQL Cluster NDB 7.2.** This series is a General Availability (GA) version of MySQL Cluster, still available for production, although we recommend that new deployments use the latest MySQL Cluster NDB 7.3 release. The most recent MySQL Cluster NDB 7.2 release can be obtained from http://dev.mysql.com/downloads/cluster/.
- **MySQL Cluster NDB 7.3.** This series is a General Availability (GA) version of MySQL Cluster, still available for production, although we recommend that new deployments use the latest MySQL Cluster NDB 7.4 release. The most recent MySQL Cluster NDB 7.3 release can be obtained from http://dev.mysql.com/downloads/cluster/.
- **MySQL Cluster NDB 7.4.** This series is the latest Generally Available (GA) version of MySQL Cluster, based on version 7.4 of the NDB storage engine and MySQL Server 5.6. New deployments should use the latest release in this series. The most recent MySQL Cluster NDB 7.4 release can be obtained from http://dev.mysql.com/downloads/cluster/.

  For an overview of improvements made in MySQL Cluster NDB 7.4, see MySQL Cluster Development in MySQL Cluster NDB 7.4.

You should use MySQL Cluster NDB 7.3 or MySQL Cluster NDB 7.4 for any new deployments; if you are using an older version of MySQL Cluster, you should upgrade to one of these soon as possible. For an overview of improvements made in MySQL Cluster NDB 7.4, see MySQL Cluster Development in MySQL Cluster NDB 7.4; for information about improvements made in MySQL Cluster NDB 7.3, see MySQL Cluster Development in MySQL Cluster NDB 7.3.

You can determine whether your MySQL Server has `NDB` support using one of the statements `SHOW VARIABLES LIKE 'have_%'`, `SHOW ENGINES`, or `SHOW PLUGINS`.

B.2: What do "NDB" and "NDBCLUSTER" mean?

"NDB" stands for "**N**etwork **D**ata**b**ase". `NDB` and `NDBCLUSTER` are both names for the storage engine that enables clustering support with MySQL. `NDB` is preferred, but either name is correct.

B.3: What is the difference between using MySQL Cluster versus using MySQL Replication?

In traditional MySQL replication, a master MySQL server updates one or more slaves. Transactions are committed sequentially, and a slow transaction can cause the slave to lag behind the master. This means that if the master fails, it is possible that the slave might not have recorded the last few transactions. If a transaction-safe engine such as `InnoDB` is being used, a transaction will either be complete on the slave or not applied at all, but replication does not guarantee that all data on the master and the slave will be consistent at all times. In MySQL Cluster, all data nodes are kept in synchrony, and a transaction committed by any one data node is committed for all data nodes. In the event of a data node failure, all remaining data nodes remain in a consistent state.

In short, whereas standard MySQL replication is *asynchronous*, MySQL Cluster is *synchronous*.

Asynchronous replication is also available in MySQL Cluster. *MySQL Cluster Replication* (also sometimes known as "geo-replication") includes the capability to replicate both between two MySQL Clusters, and from a MySQL Cluster to a non-Cluster MySQL server. See Chapter 8, *MySQL Cluster Replication*.

B.4: Do I need any special networking to run MySQL Cluster? How do computers in a cluster communicate?

MySQL Cluster is intended to be used in a high-bandwidth environment, with computers connecting using TCP/IP. Its performance depends directly upon the connection speed between the cluster's computers. The minimum connectivity requirements for MySQL Cluster include a typical 100-megabit Ethernet network or the equivalent. We recommend you use gigabit Ethernet whenever available.

B.5: How many computers do I need to run a MySQL Cluster, and why?

A minimum of three computers is required to run a viable cluster. However, the minimum *recommended* number of computers in a MySQL Cluster is four: one each to run the management and SQL nodes, and two computers to serve as data nodes. The purpose of the two data nodes is to provide redundancy; the management node must run on a separate machine to guarantee continued arbitration services in the event that one of the data nodes fails.

To provide increased throughput and high availability, you should use multiple SQL nodes (MySQL Servers connected to the cluster). It is also possible (although not strictly necessary) to run multiple management servers.

B.6: What do the different computers do in a MySQL Cluster?

A MySQL Cluster has both a physical and logical organization, with computers being the physical elements. The logical or functional elements of a cluster are referred to as *nodes*, and a computer housing a cluster node is sometimes referred to as a *cluster host*. There are three types of nodes, each corresponding to a specific role within the cluster. These are:

- **Management node.** This node provides management services for the cluster as a whole, including startup, shutdown, backups, and configuration data for the other nodes. The management node server is implemented as the application ndb_mgmd; the management client used to control MySQL Cluster is ndb_mgm. See Section 6.4, “ndb_mgmd — The MySQL Cluster Management Server Daemon”, and Section 6.5, “ndb_mgm — The MySQL Cluster Management Client”, for information about these programs.

- **Data node.** This type of node stores and replicates data. Data node functionality is handled by instances of the NDB data node process ndbd. For more information, see Section 6.1, “ndbd — The MySQL Cluster Data Node Daemon”.

- **SQL node.** This is simply an instance of MySQL Server (mysqld) that is built with support for the NDBCLUSTER storage engine and started with the --ndb-cluster option to enable the engine and the --ndb-connectstring option to enable it to connect to a MySQL Cluster management server. For more about these options, see Section 5.3.8.1, “MySQL Server Options for MySQL Cluster”.

> **Note**
>
> An *API node* is any application that makes direct use of Cluster data nodes for data storage and retrieval. An SQL node can thus be considered a type of API node that uses a MySQL Server to provide an SQL interface to the Cluster. You can write such applications (that do not depend on a MySQL Server) using the NDB API, which supplies a direct, object-oriented transaction and scanning interface to MySQL Cluster data; see MySQL Cluster API Overview: The NDB API, for more information.

B.7: When I run the SHOW command in the MySQL Cluster management client, I see a line of output that looks like this:

```
id=2    @10.100.10.32  (Version: 5.6.27-ndb-7.3.12 Nodegroup: 0, *)
```

What does the * mean? How is this node different from the others?

The simplest answer is, “It's not something you can control, and it's nothing that you need to worry about in any case, unless you're a software engineer writing or analyzing the MySQL Cluster source code”.

If you don't find that answer satisfactory, here's a longer and more technical version:

A number of mechanisms in MySQL Cluster require distributed coordination among the data nodes. These distributed algorithms and protocols include global checkpointing, DDL (schema) changes, and node restart handling. To make this coordination simpler, the data nodes “elect” one of their number to act as leader. (This node was once referred to as a “master”, but this terminology was dropped to avoid confusion with master server in MySQL Replication.) There is no user-facing mechanism for influencing this selection, which is completely automatic; the fact that it *is* automatic is a key part of MySQL Cluster's internal architecture.

When a node acts as the “leader” for any of these mechanisms, it is usually the point of coordination for the activity, and the other nodes act as “followers”, carrying out their parts of the activity as directed by the leader. If the node acting as leader fails, then the remaining nodes elect a new leader. Tasks in progress that were being coordinated by the old leader may either fail or be continued by the new leader, depending on the actual mechanism involved.

It is possible for some of these different mechanisms and protocols to have different leader nodes, but in general the same leader is chosen for all of them. The node indicated as the leader in the output of SHOW in the management client is known internally as the DICT manager (see The DBDICT Block, in the *MySQL Cluster API Developer Guide*, for more information), responsible for coordinating DDL and metadata activity.

MySQL Cluster is designed in such a way that the choice of leader has no discernible effect outside the cluster itself. For example, the current leader does not have significantly higher CPU or resource usage than the other data nodes, and failure of the leader should not have a significantly different impact on the cluster than the failure of any other data node.

B.8: With which operating systems can I use MySQL Cluster?

MySQL Cluster is supported on most Unix-like operating systems. MySQL Cluster is also supported in production settings on Microsoft Windows operating systems.

For more detailed information concerning the level of support which is offered for MySQL Cluster on various operating system versions, operating system distributions, and hardware platforms, please refer to http://www.mysql.com/support/supportedplatforms/cluster.html.

B.9: What are the hardware requirements for running MySQL Cluster?

MySQL Cluster should run on any platform for which `NDB`-enabled binaries are available. For data nodes and API nodes, faster CPUs and more memory are likely to improve performance, and 64-bit CPUs are likely to be more effective than 32-bit processors. There must be sufficient memory on machines used for data nodes to hold each node's share of the database (see *How much RAM do I Need?* for more information). For a computer which is used only for running the MySQL Cluster management server, the requirements are minimal; a common desktop PC (or the equivalent) is generally sufficient for this task. Nodes can communicate through the standard TCP/IP network and hardware. They can also use the high-speed SCI protocol; however, special networking hardware and software are required to use SCI (see Section 5.4, “Using High-Speed Interconnects with MySQL Cluster”).

B.10: How much RAM do I need to use MySQL Cluster? Is it possible to use disk memory at all?

Formerly MySQL Cluster was in-memory only. MySQL 5.1 and later also provide the ability to store MySQL Cluster on disk. (Note that we have no plans to backport this capability to previous releases.) See Section 7.12, “MySQL Cluster Disk Data Tables”, for more information.

For in-memory `NDB` tables, you can use the following formula for obtaining a rough estimate of how much RAM is needed for each data node in the cluster:

```
(SizeofDatabase × NumberOfReplicas × 1.1 ) / NumberOfDataNodes
```

To calculate the memory requirements more exactly requires determining, for each table in the cluster database, the storage space required per row (see Data Type Storage Requirements, for details), and multiplying this by the number of rows. You must also remember to account for any column indexes as follows:

- Each primary key or hash index created for an `NDBCLUSTER` table requires 21−25 bytes per record. These indexes use `IndexMemory`.
- Each ordered index requires 10 bytes storage per record, using `DataMemory`.
- Creating a primary key or unique index also creates an ordered index, unless this index is created with `USING HASH`. In other words:
  - A primary key or unique index on a Cluster table normally takes up 31 to 35 bytes per record.
  - However, if the primary key or unique index is created with `USING HASH`, then it requires only 21 to 25 bytes per record.

Creating MySQL Cluster tables with `USING HASH` for all primary keys and unique indexes will generally cause table updates to run more quickly—in some cases by a much as 20 to 30 percent faster than updates on tables where `USING HASH` was not used in creating primary and unique keys. This is due to the fact that less memory is required (because no ordered indexes are created), and that less CPU must be utilized (because fewer indexes must be read and possibly updated). However, it also means that queries that could otherwise use range scans must be satisfied by other means, which can result in slower selects.

When calculating Cluster memory requirements, you may find useful the `ndb_size.pl` utility which is available in recent MySQL 5.6 releases. This Perl script connects to a current (non-Cluster) MySQL database and creates a report on how much space that database would require if it used the `NDBCLUSTER` storage engine. For more information, see Section 6.25, "`ndb_size.pl` — NDBCLUSTER Size Requirement Estimator".

It is especially important to keep in mind that *every MySQL Cluster table must have a primary key*. The `NDB` storage engine creates a primary key automatically if none is defined; this primary key is created without `USING HASH`.

You can determine how much memory is being used for storage of MySQL Cluster data and indexes at any given time using the `REPORT MEMORYUSAGE` command in the `ndb_mgm` client; see Section 7.2, "Commands in the MySQL Cluster Management Client", for more information. In addition, warnings are written to the cluster log when 80% of available `DataMemory` or `IndexMemory` is in use, and again when usage reaches 85%, 90%, and so on.

B.11: What file systems can I use with MySQL Cluster? What about network file systems or network shares?

Generally, any file system that is native to the host operating system should work well with MySQL Cluster. If you find that a given file system works particularly well (or not so especially well) with MySQL Cluster, we invite you to discuss your findings in the MySQL Cluster Forums.

For Windows, we recommend that you use `NTFS` file systems for MySQL Cluster, just as we do for standard MySQL. We do not test MySQL Cluster with `FAT` or `VFAT` file systems. Because of this, we do not recommend their use with MySQL or MySQL Cluster.

MySQL Cluster is implemented as a shared-nothing solution; the idea behind this is that the failure of a single piece of hardware should not cause the failure of multiple cluster nodes, or possibly even the failure of the cluster as a whole. For this reason, the use of network shares or network file systems is not supported for MySQL Cluster. This also applies to shared storage devices such as SANs.

B.12: Can I run MySQL Cluster nodes inside virtual machines (such as those created by VMWare, Parallels, or Xen)?

MySQL Cluster is supported for use in virtual machines beginning with MySQL Cluster NDB 7.2. We currently support and test using Oracle VM.

Some MySQL Cluster users have successfully deployed MySQL Cluster using other virtualization products; in such cases, Oracle can provide MySQL Cluster support, but issues specific to the virtual environment must be referred to that product's vendor.

B.13: I am trying to populate a MySQL Cluster database. The loading process terminates prematurely and I get an error message like this one: `ERROR 1114: The table 'my_cluster_table' is full` Why is this happening?

The cause is very likely to be that your setup does not provide sufficient RAM for all table data and all indexes, *including the primary key required by the `NDB` storage engine and automatically created in the event that the table definition does not include the definition of a primary key*.

It is also worth noting that all data nodes should have the same amount of RAM, since no data node in a cluster can use more memory than the least amount available to any individual data node. For example, if there are four computers hosting Cluster data nodes, and three of these have 3GB of RAM available to store Cluster data while the remaining data node has only 1GB RAM, then each data node can devote at most 1GB to MySQL Cluster data and indexes.

In some cases it is possible to get `Table is full` errors in MySQL client applications even when `ndb_mgm -e "ALL REPORT MEMORYUSAGE"` shows significant free `DataMemory`. You can force

NDB to create extra partitions for MySQL Cluster tables and thus have more memory available for hash indexes by using the MAX_ROWS option for CREATE TABLE. In general, setting MAX_ROWS to twice the number of rows that you expect to store in the table should be sufficient.

For similar reasons, you can also sometimes encounter problems with data node restarts on nodes that are heavily loaded with data. In MySQL Cluster NDB 7.1 and later, the addition of the MinFreePct parameter helps with this issue by reserving a portion (5% by default) of DataMemory and IndexMemory for use in restarts. This reserved memory is not available for storing NDB tables or data.

B.14: MySQL Cluster uses TCP/IP. Does this mean that I can run it over the Internet, with one or more nodes in remote locations?

It is *very* unlikely that a cluster would perform reliably under such conditions, as MySQL Cluster was designed and implemented with the assumption that it would be run under conditions guaranteeing dedicated high-speed connectivity such as that found in a LAN setting using 100 Mbps or gigabit Ethernet—preferably the latter. We neither test nor warrant its performance using anything slower than this.

Also, it is extremely important to keep in mind that communications between the nodes in a MySQL Cluster are not secure; they are neither encrypted nor safeguarded by any other protective mechanism. The most secure configuration for a cluster is in a private network behind a firewall, with no direct access to any Cluster data or management nodes from outside. (For SQL nodes, you should take the same precautions as you would with any other instance of the MySQL server.) For more information, see Section 7.11, “MySQL Cluster Security Issues”.

B.15: Do I have to learn a new programming or query language to use MySQL Cluster?

No. Although some specialized commands are used to manage and configure the cluster itself, only standard (My)SQL statements are required for the following operations:

- Creating, altering, and dropping tables
- Inserting, updating, and deleting table data
- Creating, changing, and dropping primary and unique indexes

Some specialized configuration parameters and files are required to set up a MySQL Cluster—see Section 5.3, “MySQL Cluster Configuration Files”, for information about these.

A few simple commands are used in the MySQL Cluster management client (ndb_mgm) for tasks such as starting and stopping cluster nodes. See Section 7.2, “Commands in the MySQL Cluster Management Client”.

B.16: What programming languages and APIs are supported by MySQL Cluster?

MySQL Cluster supports the same programming APIs and languages as the standard MySQL Server, including ODBC, .Net, the MySQL C API, and numerous drivers for popular scripting languages such as PHP, Perl, and Python. MySQL Cluster applications written using these APIs behave similarly to other MySQL applications; they transmit SQL statements to a MySQL Server (in the case of MySQL Cluster, an SQL node), and receive responses containing rows of data. For more information about these APIs, see Connectors and APIs.

MySQL Cluster also supports application programming using the NDB API, which provides a low-level C++ interface to MySQL Cluster data without needing to go through a MySQL Server. See The NDB API. In addition, many NDBCLUSTER management functions are exposed by the C-language MGM API; see The MGM API, for more information.

MySQL Cluster (NDB 7.1 and later) also supports Java application programming using ClusterJ, which supports a domain object model of data using sessions and transactions. See Java and MySQL Cluster, for more information.

MySQL Cluster (NDB 7.2 and later) also supports `memcached`, allowing developers to access data stored in MySQL Cluster using the `memcached` interface; for more information, see ndbmemcache—Memcache API for MySQL Cluster.

MySQL Cluster NDB 7.3 adds adapters supporting NoSQL applications written against `Node.js`, with MySQL Cluster as the data store. See MySQL NoSQL Connector for JavaScript, for more information.

B.17: Does MySQL Cluster include any management tools?

MySQL Cluster includes a command line client for performing basic management functions. See Section 6.5, "`ndb_mgm` — The MySQL Cluster Management Client", and Section 7.2, "Commands in the MySQL Cluster Management Client".

MySQL Cluster NDB 7.0 and later is also supported by MySQL Cluster Manager, a separate product providing an advanced command line interface that can automate many MySQL Cluster management tasks such as rolling restarts and configuration changes. For more information about MySQL Cluster Manager, see MySQL™ Cluster Manager 1.3.6 User Manual.

MySQL Cluster NDB 7.3 introduces a graphical, browser-based Auto-Installer for setting up and deploying MySQL Cluster, as part of the MySQL Cluster software distribution. For more information, see Section 4.1, "The MySQL Cluster Auto-Installer".

B.18: How do I find out what an error or warning message means when using MySQL Cluster?

There are two ways in which this can be done:

- From within the `mysql` client, use `SHOW ERRORS` or `SHOW WARNINGS` immediately upon being notified of the error or warning condition.
- From a system shell prompt, use `perror --ndb error_code`.

B.19: Is MySQL Cluster transaction-safe? What isolation levels are supported?

Yes. For tables created with the `NDB` storage engine, transactions are supported. Currently, MySQL Cluster supports only the `READ COMMITTED` transaction isolation level.

B.20: What storage engines are supported by MySQL Cluster?

Clustering with MySQL is supported only by the `NDB` storage engine. That is, in order for a table to be shared between nodes in a MySQL Cluster, the table must be created using `ENGINE=NDB` (or the equivalent option `ENGINE=NDBCLUSTER`).

It is possible to create tables using other storage engines (such as `InnoDB` or `MyISAM`) on a MySQL server being used with a MySQL Cluster, but since these tables do not use `NDB`, they do not participate in clustering; each such table is strictly local to the individual MySQL server instance on which it is created.

B.21: In the event of a catastrophic failure—say, for instance, the whole city loses power and my UPS fails—would I lose all my data?

All committed transactions are logged. Therefore, although it is possible that some data could be lost in the event of a catastrophe, this should be quite limited. Data loss can be further reduced by minimizing the number of operations per transaction. (It is not a good idea to perform large numbers of operations per transaction in any case.)

B.22: Is it possible to use `FULLTEXT` indexes with MySQL Cluster?

`FULLTEXT` indexing is currently supported only by the `InnoDB` (MySQL 5.6.4 and later) and `MyISAM` storage engines. See Full-Text Search Functions, for more information.

B.23: Can I run multiple nodes on a single computer?

It is possible but not always advisable. One of the chief reasons to run a cluster is to provide redundancy. To obtain the full benefits of this redundancy, each node should reside on a separate machine. If you place multiple nodes on a single machine and that machine fails, you lose all of those nodes. For this reason, if you do run multiple data nodes on a single machine, it is *extremely* important that they be set up in such a way that the failure of this machine does not cause the loss of all the data nodes in a given node group.

Given that MySQL Cluster can be run on commodity hardware loaded with a low-cost (or even no-cost) operating system, the expense of an extra machine or two is well worth it to safeguard mission-critical data. It also worth noting that the requirements for a cluster host running a management node are minimal. This task can be accomplished with a 300 MHz Pentium or equivalent CPU and sufficient RAM for the operating system, plus a small amount of overhead for the `ndb_mgmd` and `ndb_mgm` processes.

It is acceptable to run multiple cluster data nodes on a single host that has multiple CPUs, cores, or both. MySQL Cluster NDB 7.0 and later also provide a multi-threaded version of the data node binary intended for use on such systems. For more information, see Section 6.3, “`ndbmtd` — The MySQL Cluster Data Node Daemon (Multi-Threaded)”.

It is also possible in some cases to run data nodes and SQL nodes concurrently on the same machine; how well such an arrangement performs is dependent on a number of factors such as number of cores and CPUs as well as the amount of disk and memory available to the data node and SQL node processes, and you must take these factors into account when planning such a configuration.

B.24: Are there any limitations that I should be aware of when using MySQL Cluster?

Limitations on `NDB` tables in MySQL MySQL Cluster NDB 7.3 and later include the following:

- Temporary tables are not supported; a `CREATE TEMPORARY TABLE` statement using `ENGINE=NDB` or `ENGINE=NDBCLUSTER` fails with an error.
- The only types of user-defined partitioning supported for `NDBCLUSTER` tables are `KEY` and `LINEAR KEY`. Trying to create an `NDB` table using any other partitioning type fails with an error.
- `FULLTEXT` indexes are not supported.
- Index prefixes are not supported. Only complete columns may be indexed.
- Spatial indexes are not supported (although spatial columns can be used). See Extensions for Spatial Data.
- Support for partial transactions and partial rollbacks is comparable to that of other transactional storage engines such as `InnoDB` that can roll back individual statements.
- The maximum number of attributes allowed per table is 512. Attribute names cannot be any longer than 31 characters. For each table, the maximum combined length of the table and database names is 122 characters.
- The maximum size for a table row is 14 kilobytes, not counting `BLOB` values.

  There is no set limit for the number of rows per `NDB` table. Limits on table size depend on a number of factors, in particular on the amount of RAM available to each data node.

For a complete listing of limitations in MySQL Cluster, see Section 3.6, “Known Limitations of MySQL Cluster”. See also Section 3.6.11, “Previous MySQL Cluster Issues Resolved in MySQL Cluster NDB 7.3”.

B.25: Does MySQL Cluster support foreign keys?

MySQL Cluster NDB 7.3 adds support for foreign key constraints, comparable to that found in the `InnoDB` storage engine; see FOREIGN KEY Constraints, for more detailed information, as well as

Using FOREIGN KEY Constraints. Applications requiring foreign key support should use MySQL Cluster NDB 7.3, MySQL Cluster NDB 7.4, or later.

B.26: How do I import an existing MySQL database into a MySQL Cluster?

You can import databases into MySQL Cluster much as you would with any other version of MySQL. Other than the limitations mentioned elsewhere in this FAQ, the only other special requirement is that any tables to be included in the cluster must use the `NDB` storage engine. This means that the tables must be created with `ENGINE=NDB` or `ENGINE=NDBCLUSTER`.

It is also possible to convert existing tables that use other storage engines to `NDBCLUSTER` using one or more `ALTER TABLE` statement. However, the definition of the table must be compatible with the `NDBCLUSTER` storage engine prior to making the conversion. In MySQL 5.6, an additional workaround is also required; see Section 3.6, “Known Limitations of MySQL Cluster”, for details.

B.27: How do MySQL Cluster nodes communicate with one another?

Cluster nodes can communicate through any of three different transport mechanisms: TCP/IP, SHM (shared memory), and SCI (Scalable Coherent Interface). Where available, SHM is used by default between nodes residing on the same cluster host; however, this is considered experimental. SCI is a high-speed (1 gigabit per second and higher), high-availability protocol used in building scalable multi-processor systems; it requires special hardware and drivers. See Section 5.4, “Using High-Speed Interconnects with MySQL Cluster”, for more about using SCI as a transport mechanism for MySQL Cluster.

B.28: What is an arbitrator?

If one or more data nodes in a cluster fail, it is possible that not all cluster data nodes will be able to “see” one another. In fact, it is possible that two sets of data nodes might become isolated from one another in a network partitioning, also known as a “split-brain” scenario. This type of situation is undesirable because each set of data nodes tries to behave as though it is the entire cluster. An arbitrator is required to decide between the competing sets of data nodes.

When all data nodes in at least one node group are alive, network partitioning is not an issue, because no single subset of the cluster can form a functional cluster on its own. The real problem arises when no single node group has all its nodes alive, in which case network partitioning (the “split-brain” scenario) becomes possible. Then an arbitrator is required. All cluster nodes recognize the same node as the arbitrator, which is normally the management server; however, it is possible to configure any of the MySQL Servers in the cluster to act as the arbitrator instead. The arbitrator accepts the first set of cluster nodes to contact it, and tells the remaining set to shut down. Arbitrator selection is controlled by the `ArbitrationRank` configuration parameter for MySQL Server and management server nodes. In MySQL Cluster NDB 7.0.7 and later, you can also use the `ArbitrationRank` configuration parameter to control the arbitrator selection process. For more information about these parameters, see Section 5.3.5, “Defining a MySQL Cluster Management Server”.

The role of arbitrator does not in and of itself impose any heavy demands upon the host so designated, and thus the arbitrator host does not need to be particularly fast or to have extra memory especially for this purpose.

B.29: What data types are supported by MySQL Cluster?

MySQL Cluster supports all of the usual MySQL data types, including those associated with MySQL's spatial extensions; however, the `NDB` storage engine does not support spatial indexes. (Spatial indexes are supported only by `MyISAM`; see Extensions for Spatial Data, for more information.) In addition, there are some differences with regard to indexes when used with `NDB` tables.

> **Note**
>
> MySQL Cluster Disk Data tables (that is, tables created with `TABLESPACE ... STORAGE DISK ENGINE=NDB` or `TABLESPACE ... STORAGE DISK`

> ENGINE=NDBCLUSTER) have only fixed-width rows. This means that (for example) each Disk Data table record containing a VARCHAR(255) column requires space for 255 characters (as required for the character set and collation being used for the table), regardless of the actual number of characters stored therein.

See Section 3.6, “Known Limitations of MySQL Cluster”, for more information about these issues.

B.30: How do I start and stop MySQL Cluster?

It is necessary to start each node in the cluster separately, in the following order:

1. Start the management node, using the ndb_mgmd command.

   You must include the -f or --config-file option to tell the management node where its configuration file can be found.

2. Start each data node with the ndbd command.

   Each data node must be started with the -c or --ndb-connectstring option so that the data node knows how to connect to the management server.

3. Start each MySQL Server (SQL node) using your preferred startup script, such as mysqld_safe.

   Each MySQL Server must be started with the --ndbcluster and --ndb-connectstring options. These options cause mysqld to enable NDBCLUSTER storage engine support and how to connect to the management server.

Each of these commands must be run from a system shell on the machine housing the affected node. (You do not have to be physically present at the machine—a remote login shell can be used for this purpose.) You can verify that the cluster is running by starting the NDB management client ndb_mgm on the machine housing the management node and issuing the SHOW or ALL STATUS command.

To shut down a running cluster, issue the command SHUTDOWN in the management client. Alternatively, you may enter the following command in a system shell:

```
shell> ndb_mgm -e "SHUTDOWN"
```

(The quotation marks in this example are optional, since there are no spaces in the command string following the -e option; in addition, the SHUTDOWN command, like other management client commands, is not case-sensitive.)

Either of these commands causes the ndb_mgm, ndb_mgm, and any ndbd processes to terminate gracefully. MySQL servers running as SQL nodes can be stopped using mysqladmin shutdown.

For more information, see Section 7.2, “Commands in the MySQL Cluster Management Client”, and Section 4.7, “Safe Shutdown and Restart of MySQL Cluster”.

B.31: What happens to MySQL Cluster data when the MySQL Cluster is shut down?

The data that was held in memory by the cluster's data nodes is written to disk, and is reloaded into memory the next time that the cluster is started.

B.32: Is it a good idea to have more than one management node for a MySQL Cluster?

It can be helpful as a fail-safe. Only one management node controls the cluster at any given time, but it is possible to configure one management node as primary, and one or more additional management nodes to take over in the event that the primary management node fails.

See Section 5.3, “MySQL Cluster Configuration Files”, for information on how to configure MySQL Cluster management nodes.

B.33: Can I mix different kinds of hardware and operating systems in one MySQL Cluster?

Yes, as long as all machines and operating systems have the same “endianness” (all big-endian or all little-endian).

It is also possible to use software from different MySQL Cluster releases on different nodes. However, we support this only as part of a rolling upgrade procedure (see Section 7.5, “Performing a Rolling Restart of a MySQL Cluster”).

B.34: Can I run two data nodes on a single host? Two SQL nodes?

Yes, it is possible to do this. In the case of multiple data nodes, it is advisable (but not required) for each node to use a different data directory. If you want to run multiple SQL nodes on one machine, each instance of `mysqld` must use a different TCP/IP port. However, in MySQL 5.6, running more than one cluster node of a given type per machine is generally not encouraged or supported for production use.

We also advise against running data nodes and SQL nodes together on the same host, since the `ndbd` and `mysqld` processes may compete for memory.

B.35: Can I use host names with MySQL Cluster?

Yes, it is possible to use DNS and DHCP for cluster hosts. However, if your application requires “five nines” availability, you should use fixed (numeric) IP addresses, since making communication between Cluster hosts dependent on services such as DNS and DHCP introduces additional potential points of failure.

B.36: Does MySQL Cluster support IPv6?

IPv6 is supported for connections between SQL nodes (MySQL servers), but connections between all other types of MySQL Cluster nodes must use IPv4.

In practical terms, this means that you can use IPv6 for replication between MySQL Clusters, but connections between nodes in the same MySQL Cluster must use IPv4. For more information, see Section 8.3, “Known Issues in MySQL Cluster Replication”.

B.37: How do I handle MySQL users in a MySQL Cluster having multiple MySQL servers?

MySQL user accounts and privileges are normally not automatically propagated between different MySQL servers accessing the same MySQL Cluster. Beginning with MySQL Cluster NDB 7.2, MySQL Cluster provides support for distributed privileges. While privilege distribution is not enabled automatically, you can activate it by following a procedure provided in the MySQL Cluster documentation. See Section 7.14, “Distributed MySQL Privileges for MySQL Cluster”, for more information.

B.38: How do I continue to send queries in the event that one of the SQL nodes fails?

MySQL Cluster does not provide any sort of automatic failover between SQL nodes. Your application must be prepared to handle the loss of SQL nodes and to fail over between them.

B.39: How do I back up and restore a MySQL Cluster?

You can use the NDB native backup and restore functionality in the MySQL Cluster management client and the `ndb_restore` program. See Section 7.3, “Online Backup of MySQL Cluster”, and Section 6.20, “`ndb_restore` — Restore a MySQL Cluster Backup”.

You can also use the traditional functionality provided for this purpose in `mysqldump` and the MySQL server. See `mysqldump` — A Database Backup Program, for more information.

B.40: What is an “angel process”?

This process monitors and, if necessary, attempts to restart the data node process. If you check the list of active processes on your system after starting `ndbd`, you can see that there are actually 2 processes running by that name, as shown here (we omit the output from `ndb_mgmd` and `ndbd` for brevity):

```
shell> ./ndb_mgmd

shell> ps aux | grep ndb
me      23002  0.0  0.0 122948  3104 ?        Ssl  14:14   0:00 ./ndb_mgmd
me      23025  0.0  0.0   5284   820 pts/2    S+   14:14   0:00 grep ndb

shell> ./ndbd -c 127.0.0.1 --initial

shell> ps aux | grep ndb
me      23002  0.0  0.0 123080  3356 ?        Ssl  14:14   0:00 ./ndb_mgmd
me      23096  0.0  0.0  35876  2036 ?        Ss   14:14   0:00 ./ndbd -c 127.0.0.1 --initial
me      23097  1.0  2.4 524116 91096 ?        Sl   14:14   0:00 ./ndbd -c 127.0.0.1 --initial
me      23168  0.0  0.0   5284   812 pts/2    R+   14:15   0:00 grep ndb
```

The `ndbd` process showing 0 memory and CPU usage is the angel process. It actually does use a very small amount of each, of course. It simply checks to see if the main `ndbd` process (the primary data node process that actually handles the data) is running. If permitted to do so (for example, if the `StopOnError` configuration parameter is set to false—see Section 5.2.1, “MySQL Cluster Data Node Configuration Parameters”), the angel process tries to restart the primary data node process.