Altibase Administration

General Reference

Release 5.5.1

January 15, 2013

ALTIBASE
PERFORMANCE SOLUTIONS

Altibase Administration General Reference

Release 5.5.1

Altibase Corporation

10F, Daerung PostTower II, 182-13,

Guro-dong Guro-gu Seoul, 152-847, Korea

Telephone: +82-2-2082-1000   Fax: 82-2-2082-1099

E-mail: support@altibase.com   www: http://www.altibase.com

Contents

Preface

About This Manual

This manual describes the concepts and architecture of ALTIBASE® HDB™. This manual also explains to administrators how to manage their databases.

Audience

This manual has been prepared for the following users of ALTIBASE HDB:

- database administrators
- application developers
- programmers

It is recommended that those reading this manual possess the following background knowledge:

- basic knowledge in the use of computers, operating systems, and operating system utilities
- experience in using relational databases and an understanding of database concepts
- computer programming experience

Software Environment

This manual has been prepared assuming that ALTIBASE HDB 5.5.1 will be used as the database server.

Organization

This manual has been organized as follows:

- Chapter1: Data Types

  This chapter explains the data types that are supported in ALTIBASE HDB.

- Chapter2: ALTIBASE HDB Properties

  This chapter lists the ALTIBASE HDB properties.

- Chapter3: The Data Dictionary

  This chapter describes the specification of the ALTIBASE HDB data dictionary. The data dictionary of ALTIBASE HDB comprises meta tables, in which information about objects is stored, and process tables, in which information about processes is stored.

- Chapter4: The Sample Schema

  This chapter describes the example table information, ER diagrams and sample data.

Documentation Conventions

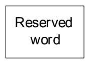
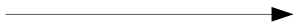
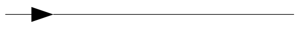
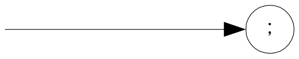
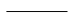
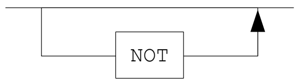
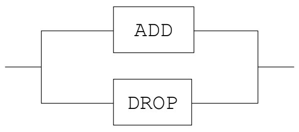
This section describes the conventions used in this manual. Understanding these conventions will make it easier to find information in this manual and other manuals in the series.

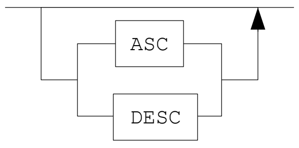
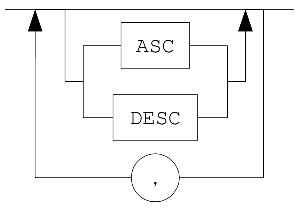
There are two sets of conventions:

- syntax diagram conventions
- sample code conventions

Syntax Diagram Conventions

This manual describes command syntax using diagrams composed of the following elements:

Elements	Meaning
	Indicates the start of a command. If a syntactic element starts with an arrow, it is not a complete command.
	Indicates that the command continues to the next line. If a syntactic element ends with this symbol, it is not a complete command.
	Indicates that the command continues from the previous line. If a syntactic element starts with this symbol, it is not a complete plete command.
	Indicates the end of a statement.
	Indicates a mandatory element.
	Indicates an optional element.
	Indicates a mandatory element comprised of options. One, and only one, option must be specified.

Elements	Meaning
	Indicates an optional element comprised of options.
	Indicates an optional element in which multiple elements may be specified. A comma must precede all but the first element.

Sample Code Conventions

The code examples explain SQL statements, stored procedures, iSQL statements, and other command line syntax.

The following table describes the printing conventions used in the code examples.

Rules	Meaning	Example
[]	Indicates an optional item.	`VARCHAR [(size)] [[FIXED |] VARIABLE]`
{}	Indicates a mandatory field for which one or more items must be selected.	`{ ENABLE | DISABLE | COMPILE }`

|	A delimiter between optional or mandatory arguments.	`{ ENABLE | DISABLE | COMPILE }`
		`[ ENABLE | DISABLE | COMPILE ]`

.
.
.
	Indicates that the previous argument is repeated, or that sample code has been omitted.	`iSQL> select e_lastname from employees;`
		`E_LASTNAME`
		`------------------------`
		`Moon`
		`Davenport`
		`Kobain`
		`.`
		`.`
		`.`
		`20 rows selected.`

Other Symbols	Symbols other than those shown above are part of the actual code.	`EXEC :p1 := 1;`
		`acc NUMBER(11,2);`

Rules	Meaning	Example
Italics	Statement elements in italics indicate variables and special values specified by the user.	`SELECT * FROM table_name;` `CONNECT userID/password;`
Lower case words	Indicate program elements set by the user, such as table names, column names, file names, etc.	`SELECT e_lastname FROM employees;`
Upper case words	Keywords and all elements provided by the system appear in upper case.	`DESC SYSTEM_.SYS_INDEX_;`

Related Documents

For more detailed information, please refer to the following documents:

- ALTIBASE HDB Getting Started Guide
- ALTIBASE HDB SQL Reference
- ALTIBASE HDB Administrator's Manual

Online Manual

Online versions of our manuals (PDF or HTML) are available from the Altibase Download Center (http://atc.altibase.com/).

Altibase Welcomes Your Opinions

Please feel free to send us your comments and suggestions regarding this manual. Your comments and suggestions are important to us, and may be used to improve future versions of the manual. When you send your feedback, please make sure to include the following information:

- The name and version of the manual you are using
- Your comments and suggestions regarding the manual
- Your full name, address, and phone number

Please send your e-mail to the following address:

support@altibase.com

In addition to suggestions, this address may also be used to report any errors or omissions discovered in the manual, which we will address promptly.

If you need immediate assistance with technical issues, please contact the Altibase Customer Support Center.

We always appreciate your comments and suggestions.

1 Data Types

In order to use SQL to store, change, and query the data in a database, it is first necessary to possess a thorough understanding of the available data types. This chapter presents a detailed explanation of the data types supported in ALTIBASE HDB.

1.1 Overview

1.1.1 Data Type Overview

The following data types are supported in ALTIBASE HDB:

1.1.1.1 Character Data Types

M: defined column length
L: the length of the input string

Type	M	Storage Required (bytes)
CHAR(M)	1 ~ 32000	M + 2
VARCHAR(M)	1 ~ 32000	length + 2, where length = L if the input value is stored in a variable area length = M if the input value is stored in a fixed area
NCHAR(M)	1~16000(UTF16) 1~10666(UTF8)	M*2 + 2(UTF16) M*3 + 2(UTF8)
NVARCHAR(M)	1~16000(UTF16) 1~10666(UTF8)	length*2 + 2(UTF16) length*3 + 2(UTF8) where: length = L if the input value is stored in a variable area length = M if the input value is stored in a fixed area

NCHAR and NVARCHAR are Unicode character types. The available maximum length of a UTF16-encoded string is different from that of a UTF8-encoded string.

1.1.1.2 Numerical Data Types

- Non-native

Type	Precision	Scale	Size (bytes)	Remarks
NUMERIC	38	0	3+((precision)+2)/2	* Fixed-Point Numbers * The NUMERIC data type is the same as the DECIMAL datatype.
NUMERIC(p)	1 ~ 38	0		
NUMERIC(p, s)	1 ~ 38	-84 ~ 128		
DECIMAL	38	0		
DECIMAL(p)	1 ~ 38	0		
DECIMAL(p, s)	1 ~ 38	-84 ~ 128		
NUMBER(p)	1 ~ 38	0		
NUMBER(p, s)	1 ~ 38	-84 ~ 128		
NUMBER	38	X	3+((precision)+2)/2	* Floating-Point Numbers
FLOAT	38	X		
FLOAT(p)	1 ~ 38	X		

- Native

Type	Compatible C Type	Size (bytes)	Remarks
DOUBLE	double	8	Floating-Point Numbers
REAL	float	4	
BIGINT	long or long long	8	Integer Type
INTEGER	int	4	
SMALLINT	short	2	

Examples

- Fixed-Point Numbers

  Size Calculation: (3 + ((p) + 2) / 2)

```
Ex) NUMERIC

NUMERIC(38,0)
Size = 3 + 40/2 = 23 bytes

Ex) NUMERIC(p) / NUMERIC(p, 0)

NUMERIC(10)
Size = 3 + 12/2 = 9 bytes

Ex) NUMERIC(p, s)

NUMERIC(10, 9)
Size = 3 + 12/2 = 9 bytes
```

- DECIMAL: the same as NUMERIC
- DECIMAL(p): the same as NUMERIC(p)
- DECIMAL(p,s): the same as NUMERIC(p,s)
- NUMBER(p): the same as NUMERIC(p)
- NUMBER(p,s): the same as NUMERIC(p,s)

• Floating-Point Numbers

Size Calculation: (3 + ((p) + 2) / 2)

```
Ex) FLOAT

FLOAT(38)
Size = 3 + 40/2 = 23 bytes

Ex) FLOAT(p)

FLOAT(20)
Size = 3 + 22/2 = 14 bytes
```

- NUMBER: the same as FLOAT

1.1.1.3 Date Data Type

Type	Size (bytes)
DATE	8

1.1.1.4 Binary Data Types

M: defined column length
L: the length of the input value

Type	M	Size (bytes)
BLOB/CLOB		1~2147483647
BYTE	1~32000	M + 2
NIBBLE	1~254	M/2 + 1
BIT	1~64000	M/8 + 4
VARBIT	1~64000	length/8 + 4, where length = L if the input value is stored in a variable area length = M if the input value is stored in a fixed area

1.1.1.5 Geometry Data Types

Type	Length	Size (bytes)
GEOMETRY	8~104857600	length + 40

The actual record size is the size of each data type as indicated above, plus the size of header information. The size of the header information varies depending on the OS.

1.1.2 NULL

When a row is inserted into a table, the value of a column is set to NULL if the value for that column is not known or has not been determined yet. In other words, NULL indicates that no value exists. Therefore, NULL is not the same as 0 (zero) or blank space, and is handled differently when performing comparison operations or saving data.

If any operation other than the NVL() function or the IS NULL or IS NOT NULL conditions is performed on a NULL value, the final result of the formula containing the operation will be NULL. In other words, comparisons and operations are meaningless when performed on NULL values.

NULL can appear in columns of any data type, as long as they are not restricted by NOT NULL or PRIMARY KEY constraints.

1.1.3 Data Type Conversion

The data type conversions that are possible are shown in matrix form in the following table.

When a comparison operation is to be performed on two values having the same data type, the comparison operation is performed on the values directly without any prior conversion. In contrast, when a comparison operation is to be performed on two values having different data types, the comparison is performed after one of the values is converted into the same type as the other value. Note however that when comparisons are performed, character data types are always converted into the data type of the other comparison operand, not the other way around.

After Before	char	varchar	nchar	nvarchar	clob	bigint	decimal	double	float	integer	number	numeric	real	smallint	date	blob	byte	nibble	bit	varbit	geometry
char	o	o	o	o		o	o	o	o	o	o	o	o	o	o						
varchar	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o						
nchar	o	o	o	o		o	o	o	o	o	o	o	o	o	o						
nvarchar	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o						
clob					o																
bigint	o	o	o	o		o	o	o	o	o	o	o	o	o							

After Before	char	varchar	nchar	nvarchar	clob	bigint	decimal	double	float	integer	number	numeric	real	smallint	date	blob	byte	nibble	bit	varbit	geometry
decimal	o	o	o	o		o	o	o	o	o	o	o	o	o							
double	o	o	o	o		o	o	o	o	o	o	o	o	o							
float	o	o	o	o		o	o	o	o	o	o	o	o	o							
integer	o	o	o	o		o	o	o	o	o	o	o	o	o							
number	o	o	o	o		o	o	o	o	o	o	o	o	o							
numeric	o	o	o	o		o	o	o	o	o	o	o	o	o							
real	o	o	o	o		o	o	o	o	o	o	o	o	o							
smallint	o	o	o	o		o	o	o	o	o	o	o	o	o							
date	o	o	o	o											o						
blob																o					
byte																o	o				
nibble																		o			
bit																			o	o	
varbit		o																	o	o	
geometry																					o

1.1.4 Explicit Data Type Conversion

Data type conversion can be explicitly performed using SQL conversion functions or by typecasting, as shown below.

1.1.4.1 Syntax

```
datatype 'string or constant literal'
```

1.1.4.2 Description

Explicitly converts a numeric value from one data type to another. In the following example, the number 157.27 is converted to the characters "157.27".

```
CHAR '157.27'
```

The SQL functions that are used to explicitly convert a value from one data type to another are explained in the *SQL Reference*.

1.1.5 The FIXED and VARIABLE Options

FIXED or VARIABLE specifies where the data in a column will be stored.

When an entire record is stored in a contiguous space, this is called a 'FIXED' area. When one of the columns is stored in a separate space, rather than being stored in the fixed area contiguous with the rest of the record, this column is said to be stored in a 'VARIABLE' area.

When a column is stored in a variable area, the header information for the column, such as the length of the data and the pointer to the actual data, is stored in the fixed area, whereas the data for that column are stored in the variable area.

When a table is created in disk tablespace, whether the user specifies FIXED or VARIABLE is ignored, and all columns in the table are treated as FIXED. However, when a table is created in memory tablespace, the user-specified value is used.

The exception to this is that all LOB data type columns are always treated as VARIABLE, and the data can thus be stored in a fixed or variable area depending on the value specified using the IN ROW clause.

The following data types can be specified as VARIABLE: CHAR, VARCHAR, NCHAR, NVARCHAR, BYTE, NIBBLE, BIT, VARBIT, BLOB, and CLOB.

1.1.6 The IN ROW clause

This clause pertains only to column data that are to be stored in a variable area. If the FIXED and IN ROW clause are both specified when a table is created, the IN ROW clause is ignored. When data are entered into a VARIABLE column, if the length of the data is less than or equal to the value specified using the IN ROW clause, the data will be stored in the fixed area, whereas if the data length is greater than the value specified using the IN ROW clause, the data will be stored in the variable area.

Here, “data length” does not mean the length of the input data, but the length of the data to be stored in memory or on disk, which will be somewhat larger. For example, when a column is defined as ‘VARCHAR(400) in row 200’, data will be inserted into the fixed area if the length of the data that are input is smaller than or equal to 198, because 2 additional bytes are required when storing the data.

The default size of lob data stored in the fixed area can be specified using the MEMORY_LOB_COLUMN_IN_ROW_SIZE property for memory tables and the DISK_LOB_COLUMN_IN_ROW_SIZE for disk tables. Additionally, the default size for columns containing other types of data with the VARIABLE option can be specified using the MEMORY_VARIABLE_COLUMN_IN_ROW_SIZE property. Setting these properties obviates the need to use the IN ROW clause repeatedly for individual columns.

For more information about these properties, please refer to the property descriptions in the *ALTIBASE HDB General Reference*.

1.2 Character Data Types

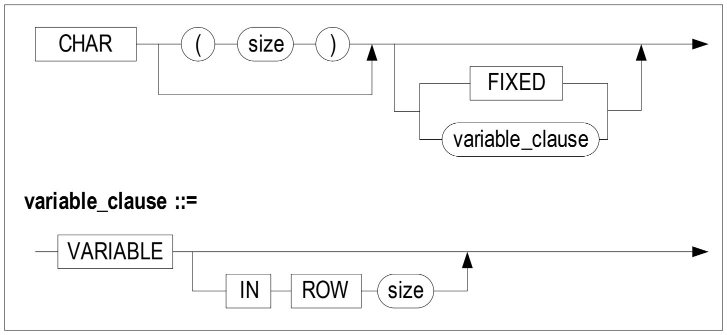
Character data types are used to store character (alphanumeric) data, meaning words or free-form text, in either the database character set or the national character set.

In ALTIBASE HDB, character data types comprise the following types:

- CHAR
- VARCHAR
- NCHAR
- NVARCHAR

1.2.1 CHAR

1.2.1.1 Syntax Diagram



1.2.1.2 Syntax

```
CHAR [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.2.1.3 Description

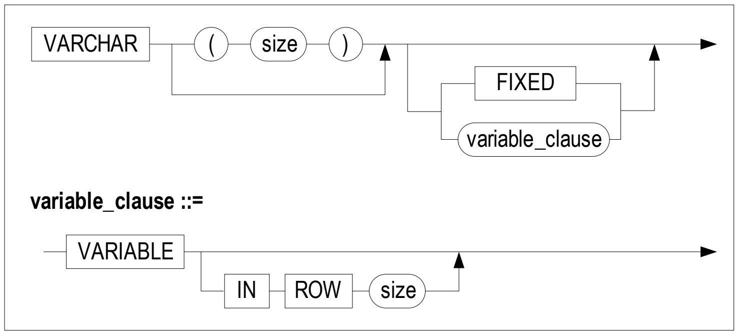
This is a character data type that has a fixed length equal to the specified size. If an input value is shorter than the specified size, the remaining area is filled with blank spaces.

The default size of a CHAR column is 1 byte. The maximum size is 32000 bytes.

For more information on the FIXED and VARIABLE clauses, please refer to the preceding sections, entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.2.2 VARCHAR

1.2.2.1 Syntax Diagram



1.2.2.2 Syntax

```
VARCHAR [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.2.2.3 Description

This is a character data type for storing alphanumeric data that vary in length within a specified size.

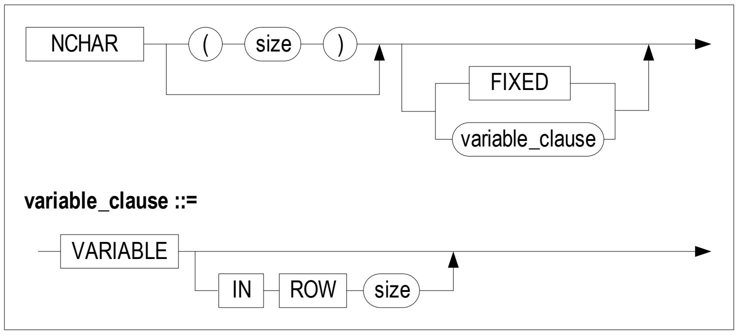
The default size of a VARCHAR column is 1 byte. The maximum size is 32000 bytes.

VARCHAR is a variable length data type; that is, when the length of input data is shorter than the specified column size, only the data that were actually inserted are stored. In contrast, for the CHAR data type, if the length of input data is shorter than the column length, the remaining space in the column is padded with blank spaces. For example, if a column is defined as CHAR(10) and the word "magic" is to be stored, it will be stored as "magic_____", where "_" represents a blank space.

For more information on the FIXED and VARIABLE clauses, please refer to the preceding sections, entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.2.3 NCHAR

1.2.3.1 Syntax Diagram



1.2.3.2 Syntax

```
NCHAR [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.2.3.3 Description

This is a character data type having a specified fixed length. If an input value is shorter than the specified size, the remainder is filled with blank spaces.

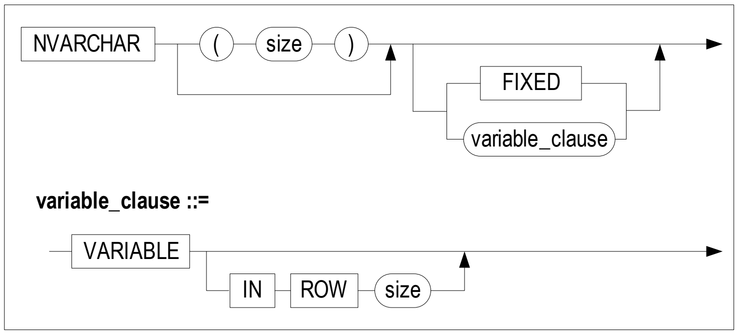
If the national character set is UTF16, the size of one character in an NCHAR column is fixed at 2 bytes, that is, it does not vary in length. In contrast, if the national character set is UTF8, the size of one character in an NCHAR column is not fixed; rather, it varies from 1 to 3 bytes.

The maximum size is 16000 bytes if the national character set is UTF16.

For more information on the FIXED and VARIABLE clauses, please refer to the preceding sections, entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.2.4 NVARCHAR

1.2.4.1 Syntax Diagram



1.2.4.2 Syntax

```
NVARCHAR [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.2.4.3 Description

This is a character data type for storing Unicode alphanumeric data that vary in length within a specified size.

If the national character set is UTF16, the size of one character in an NVARCHAR column is fixed at 2 bytes, that is, it does not vary in length. In contrast, if the national character set is UTF8, the size of one character in an NVARCHAR column is not fixed; rather, it varies from 1 to 3 bytes.

In other aspects, the NVARCHAR type is the same as the VARCHAR type, so for more detailed information please refer to the description of the VARCHAR type.

For more information on the FIXED and VARIABLE clauses, please refer to the preceding sections, entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

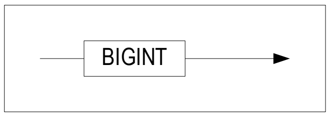
1.3 Numeric Data Types

Numeric data types are used to store zero as well as positive and negative numbers having fixed values. ALTIBASE HDB supports the following numeric types:

- BIGINT
- DECIMAL
- DOUBLE
- FLOAT
- INTEGER
- NUMBER
- NUMERIC
- REAL
- SMALLINT

1.3.1 BIGINT

1.3.1.1 Syntax Diagram



1.3.1.2 Syntax

```
BIGINT
```

1.3.1.3 Description

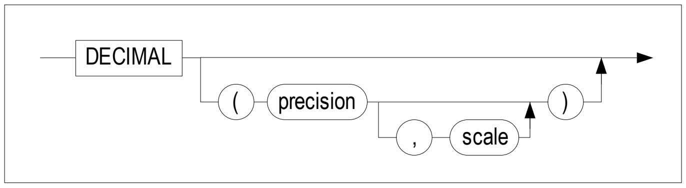
This is an 8-byte integer data type.

It is equivalent to the "long" (on 64-bit systems) and "long long" (on 32-bit systems) types in the C language.

Range: $-2^{63} + 1$(-9223372036854775807) ~ $2^{63} - 1$ (9223372036854775807)

1.3.2 DECIMAL

1.3.2.1 Syntax Diagram



1.3.2.2 Syntax

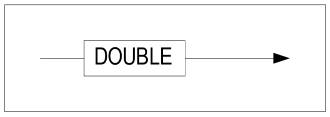
```
DECIMAL [(precision[, scale])]
```

1.3.2.3 Description

This data type is the same as the NUMERIC type.

1.3.3 DOUBLE

1.3.3.1 Syntax Diagram



1.3.3.2 Syntax

```
DOUBLE
```

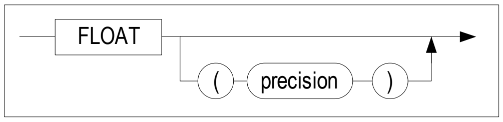
1.3.3.3 Description

This is an 8-byte floating-point numeric data type.

It is the same as the "double" type in the C language.

1.3.4 FLOAT

1.3.4.1 Syntax Diagram



1.3.4.2 Syntax

```
FLOAT [(precision)]
```

1.3.4.3 Description

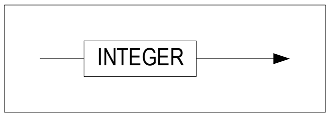
This is a floating-point numeric data type that can store a value ranging from -1E+120 to 1E+120.

Precision is the number of significant digits, that is, the number of digits used to express the mantissa of the floating-point number.

Precision can range from 1 to 38. If it is not expressly specified, the default precision is 38.

1.3.5 INTEGER

1.3.5.1 Syntax Diagram



1.3.5.2 Syntax

```
INTEGER
```

1.3.5.3 Description

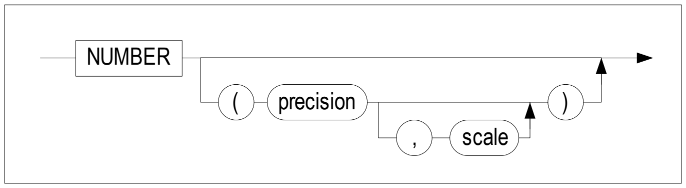
This is an integer data type that is 4 bytes in size.

It is the same as the “int” data type in the C language.

It can have an integer value ranging from -2,147,483,647 to 2,147,483,647.

1.3.6 NUMBER

1.3.6.1 Syntax Diagram



1.3.6.2 Syntax

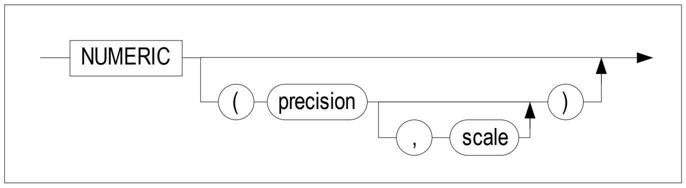
```
NUMBER [(precision, scale)]
```

1.3.6.3 Description

This is an alias of the NUMERIC data type. However, when *precision* and *scale* are not defined, they are the same as for the FLOAT data type.

1.3.7 NUMERIC

1.3.7.1 Syntax Diagram



1.3.7.2 Syntax

```
NUMERIC [(precision, scale)]
```

1.3.7.3 Description

NUMERIC is a fixed decimal data type that can contain a total number of significant digits up to the value specified using *precision* and a number of digits to the right of the decimal place up to the value specified using *scale*. In contrast to the FLOAT data type, which is a floating-point numerical data type used for representing real numbers, when both *precision* and *scale* are omitted from a NUMERIC data type declaration, *precision* defaults to 38 and *scale* to 0, i.e. NUMERIC defaults to a fixed decimal data type that is used to express integer values.

- *Precision* can be specified within the range from 1 to 38.

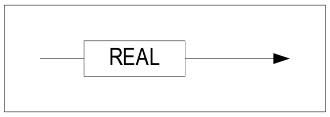
- *Scale* can be specified within the range from -84 to 126.
- If *precision* is omitted, the default is 38.
- If *scale* is omitted, the default is 0.

The following shows the respective values that would result when the input value 1234567.89 is converted to the NUMERIC types defined as shown.

- NUMERIC=> 1234568
- NUMERIC(9)=> 1234568
- NUMERIC(9, 2)=> 1234567.89
- NUMERIC(9, 1)=> 1234567.9
- NUMERIC(6)=> Precision exceeded
- NUMERIC(7, -2)=> 1234500
- NUMERIC(7, 2)=> Precision exceeded

1.3.8 REAL

1.3.8.1 Syntax Diagram



1.3.8.2 Syntax

```
REAL
```

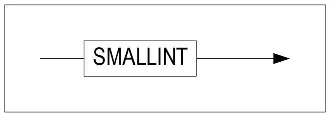
1.3.8.3 Description

This data type is used to store 4-byte floating-point numeric values.

It is the same as the “float” type in the C language.

1.3.9 SMALLINT

1.3.9.1 Syntax Diagram



1.3.9.2 Syntax

```
SMALLINT
```

1.3.9.3 Description

This data type is used to store 2-byte integer values.

It is the same as the "short" type in the C language.

It can be used to store integers ranging from $-2^{15} + 1$(-32,767) to $2^{15} - 1$(32,767) inclusive.

1.3.10 Number Format Model

When data are converted using typecasting functions such as TO_CHAR or TO_NUMBER, numeric data can be specified in the following formats. A number format model consists of one or more elements that represent a number. In this section, each of these elements will be explained with reference to examples showing the related number formats.

1.3.10.1 , (comma)

Description

Outputs a comma at the specified position. More than one comma can be used.

Restrictions

A comma cannot be placed at the end of a number, to the right of a decimal point, or at the very beginning of a number.

Example

```
iSQL> SELECT TO_CHAR (1234, '99,99') FROM dual;
TO_CHAR (1234, '99,99')
---------------------------
 12,34
1 row selected.

iSQL> SELECT TO_NUMBER ( '12,34', '99,99') FROM dual;
TO_NUMBER ( '12,34', '99,99')
--------------------------------
```

```
1234
1 row selected.
```

1.3.10.2 . (decimal point)

Description

Adds a decimal point at the specified position.

Restriction

Only one decimal point can be used within a number.

Example

```
iSQL> SELECT TO_CHAR (1.234, '99.999') FROM dual;
TO_CHAR (1.234, '99.999')
-----------------------------
 1.234
1 row selected.

iSQL> SELECT TO_NUMBER ( '1.234', '99.999') FROM dual;
TO_NUMBER ( '1.234', '99.999')
---------------------------------
1.234
1 row selected.
```

1.3.10.3 $

Description

Prepends the $ sign to a number.

Example

```
iSQL> SELECT TO_CHAR (123, '$9999') FROM dual;
TO_CHAR (123, '$9999')
--------------------------
 $123
1 row selected.
iSQL> SELECT TO_NUMBER ( '$0123', '09$99') FROM dual;
TO_NUMBER ( '$0123', '09$99')
--------------------------------
123
1 row selected.
```

1.3.10.4 0 (numeral 0)

Description

If the number of significant digits to be output exceeds the number of digits in the number that is input, 0's (zeroes) are prepended to the number before it is returned. In all other aspects, this element is the same as the “9” element, which is described below.

Example

```
iSQL> SELECT TO_CHAR (123, '0999') FROM dual;
TO_CHAR (123, '0999')
-------------------------
 0123
```

1.3.10.5 9 (numeral 9)

Description

Uses the numeral 9 to indicate the number of digits to output. If the number of 9's is greater than the number of digits in the number that is input, the space to the left of the number is padded with blank spaces before the number is output. If the number of 9's to the left of the decimal point is less than the number of digits to the left of the decimal point in the input number, the pound sign ("#") is repeatedly output. The number of pound signs that are output is the number of characters in the user-defined format plus one (a sign character). A decimal point placed in between 9's separates the integer and fractional parts of a number.

When there are digits to the right of the decimal point in the first argument, i.e. when the input number has a fractional part, but the user-defined format either has no fractional part or has a fractional part with a smaller number of decimal places than the input number, the input number is rounded off to the number of decimal places in the user-defined format.

Example

```
iSQL> SELECT TO_CHAR (123, '99999') FROM dual;
TO_CHAR (123, '99999')
--------------------------
   123

iSQL> SELECT TO_CHAR (123.55, '999') FROM dual;
TO_CHAR (123.55, '999')
---------------------------
 124
1 row selected.

iSQL> SELECT TO_CHAR (123.4567, '999999') FROM dual;
TO_CHAR (123.4567, '999999')
--------------------------------
    123
1 row selected.

iSQL> SELECT TO_CHAR (1234.578, '9999.99') FROM dual;
TO_CHAR (1234.578, '9999.99')
---------------------------------
 1234.58
1 row selected.

iSQL> SELECT TO_CHAR (1234.578, '999.99999') FROM dual;
TO_CHAR (1234.578, '999.99999')
-----------------------------------
##########
1 row selected.

iSQL> SELECT TO_NUMBER ( '123', '99999') FROM dual;
TO_NUMBER ( '123', '99999')
------------------------------
123
```

```
1 row selected.

iSQL> SELECT TO_NUMBER ( '1234.58', '9999.99') FROM dual;
TO_NUMBER ( '1234.58', '9999.99')
------------------------------------
1234.58
1 row selected.
```

1.3.10.6 B

Description

0's (zeroes) in the integer part of the fixed-point number are replaced with blank spaces.

Example

```
iSQL> SELECT TO_CHAR (0.4, 'B9') FROM T1;
TO_CHAR (0.4, 'B9')
-----------------------

1 row selected.
```

1.3.10.7 EEEE

Description

Display the input number in exponential notation.

Restrictions

EEEE must always be at the rightmost place of the number format. However, it can precede S, PR or MI. It cannot be used with commas, and cannot be used with the TO_NUMBER function.

Example

```
iSQL> SELECT TO_CHAR (1234, '9.9EEEE') FROM dual;
TO_CHAR (1234, '9.9EEEE')
-----------------------------
 1.2E+03
1 row selected.
```

1.3.10.8 MI

Description

When MI is used at the rightmost place in the number format, if the input value is negative, the minus (-) sign is output at the end of the number, rather than at the beginning. If the input value is positive, a blank space is output instead of the minus sign.

Restrictions

MI must always be at the rightmost place in the number format. It cannot be used together with S or PR.

Example

```
iSQL> SELECT TO_CHAR (-123, '999MI') FROM dual;
TO_CHAR (-123, '999MI')
---------------------------
123-
1 row selected.

iSQL> SELECT TO_NUMBER ( '123-', '999MI') FROM dual;
TO_NUMBER ( '123-', '999MI')
-------------------------------
-123
1 row selected.
```

1.3.10.9 PR

Description

When PR is used at the rightmost place in the number format, if the input value is negative, the value is output in the form of "<number>", rather than using the minus ("-") sign.

Restrictions

PR must always be at the rightmost place in the number format. It cannot be used together with S or MI.

Example

```
iSQL> SELECT TO_CHAR (-123, '999PR') FROM dual;
TO_CHAR (-123, '999PR')
---------------------------
<123>
1 row selected.

iSQL> SELECT TO_NUMBER ( '<123>', '999PR') FROM dual;
TO_NUMBER ( '<123>', '999PR')
--------------------------------
-123
1 row selected.
```

1.3.10.10 RN

Description

Converts an input number to Roman numerals. The valid input range is from 1 to 3,999. If the lower-case letters "rn" are used in the number format, lower-case Roman numerals are output.

Restrictions

RN cannot be used with any other number format elements or with the TO_NUMBER function.

Example

```
iSQL> SELECT TO_CHAR (14, 'RN') FROM dual;
TO_CHAR (14, 'RN')
----------------------
```

```
XIV
1 row selected.
```

1.3.10.11 S

Description

When S is placed at the beginning or end of the number format, a plus ("+") or minus ("-") sign is output at the same position, corresponding to the sign of the input number.

Restrictions

S can be placed at the beginning or end of the number format. It cannot be used with MI or PR.

Example

```
iSQL> SELECT TO_CHAR (123, 'S999.99') FROM dual;
TO_CHAR (123, 'S999.99')
----------------------------
+123.00
1 row selected.

iSQL> SELECT TO_CHAR (-123, '999.99S') FROM dual;
TO_CHAR (-123, '999.99S')
-----------------------------
123.00-
1 row selected.

iSQL> SELECT TO_NUMBER ( '+123', 'S999.99') FROM dual;
TO_NUMBER ( '+123', 'S999.99')
---------------------------------
123
1 row selected.

iSQL> SELECT TO_NUMBER ( '123.00-', '999.99S') FROM dual;
TO_NUMBER ( '123.00-', '999.99S')
------------------------------------
-123
1 row selected.
```

1.3.10.12 V

Description

The input number is multiplied by 10 to the power of the number of 9's after V. The number of 9's before V represents the number of significant digits to return from the input number.

Restrictions

V cannot be used with a decimal point, and cannot be used with the TO_NUMBER function.

Example

```
iSQL> SELECT TO_CHAR (12, '99V99') FROM dual;
TO_CHAR (12, '99V99')
-------------------------
```

```
 1200
1 row selected.

iSQL> SELECT TO_CHAR (1200, '99V99') FROM dual;
TO_CHAR (1200, '99V99')
---------------------------
######
1 row selected.

iSQL> SELECT TO_CHAR (-123.456, '999V999EEEEMI') from dual;
TO_CHAR (-123.456, '999V999EEEEMI')
---------------------------------------
 1235E+02-
1 row selected.
```

1.3.10.13 XXXX

Description

Converts the input number to a hexadecimal number. If the input number is not an integer, it is rounded off before being converted to a hexadecimal number. Specifying “xxxx” in lower-case returns the letters in the hexadecimal number in lower-case.

Restrictions

XXXX cannot be used with other number format elements. The number to be converted must be greater than 0 (zero).

Example

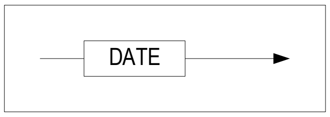
```
iSQL> SELECT TO_CHAR (123, 'XXXX') FROM dual;
TO_CHAR (123, 'XXXX')
-------------------------
7B
1 row selected.

iSQL> SELECT TO_NUMBER ('ABC', 'XXXX') FROM dual;
TO_NUMBER ('ABC', 'XXXX')
----------------------------
2748
1 row selected.
```

1.4 Date Data Types

The DATE type is used to store date and time information. Although date and time information can also be represented using both character and number data types, the DATE data type has special properties. This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND.

1.4.1 DATE

1.4.1.1 Syntax Diagram



1.4.1.2 Syntax

```
DATE
```

1.4.1.3 Description

This data type is used to stores date values in 8 bytes.

The range of dates that can be stored depends on the system. Typically, the dates that can be stored range from 0001/01/01 - 9999/12/31.

The date value can be displayed in various formats using a date format string.

1.4.2 The Datetime Format Model

Date type data are managed as numerical data within a database. However, users can display date data as a string after conversion using the TO_CHAR and TO_DATE conversion functions. When using conversion functions, the user must specify a date data type string in the desired format.

The datetime format model consists of the following basic elements:

- AM, PM
- CC
- D, DD, DDD, DAY,DY
- HH, HH12, HH24
- MM, MON, MONTH
- MI

- Q
- SS, SSSSS, SSSSSS, SSSSSSSS, FF[1..6]
- WW, W
- Y,YYY
- YYYY, YYY, YY, Y, RR, RRRR

Along with these basic elements, the datetime format model also comprises the following punctuation marks and special characters:

- Hyphen (-)
- Slash (/)
- Comma (,)
- Period (.)
- Colon (:)
- Single Quotation (')

The meaning and use of each of these basic elements will be explained below with reference to examples.

1.4.2.1 AM/PM

Description

Returns either "AM" or "PM" depending on whether the input time is before or after noon. This element can be specified as either "AM" or "PM" when input, regardless of whether "AM" or "PM" is output.

Example

```
% export ALTIBASE_DATE_FORMAT="YYYY/MM/DD HH:MI:SS"

iSQL> SELECT TO_CHAR ( TO_DATE( '13', 'HH' ), 'AM' ) FROM dual;
TO_CHAR ( TO_DATE( '13', 'HH' ), 'AM' )
-------------------------------------------
PM
1 row selected.

iSQL> SELECT TO_DATE('1980-12-28 PM', 'YYYY-MM-DD AM') FROM dual;
TO_DATE('1980-12-28 PM', 'YYYY-MM-DD AM')
-------------------------------------------
1980/12/28 12:00:00
1 row selected.
```

1.4.2.2 CC

Description

Represents a century.

- If the last 2 digits of an input 4-digit year are within the range from 01 to 99, the sum of 1 plus the first 2 digits of the 4-digit year is returned.
- If the last 2 digits of an input 4-digit year are 00, the first 2 digits of the 4-digit year are returned unchanged.

CC cannot be used as an argument for the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'CC' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'CC' )
-----------------------------------
20
1 row selected.
```

1.4.2.3 D

Description

Returns the day of the week, represented by a number from 1 to 7. Sunday is represented by the number 1.

D cannot be used as an argument for the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'D' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'D' )
----------------------------------
1
1 row selected.
```

1.4.2.4 DAY

Description

Returns the day of the week in upper-case letters in English (SUNDAY, MONDAY,…).

DAY cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'DAY' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'DAY' )
------------------------------------
SUNDAY
1 row selected.
```

1.4.2.5 DD

Description

Returns the day of the month, represented by a number from 1 to 31.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'DD' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'DD' )
-----------------------------------
28
1 row selected.

iSQL> SELECT TO_DATE( '1980-12-28', 'YYYY-MM-DD') FROM dual;
TO_DATE( '1980-12-28', 'YYYY-MM-DD')
---------------------------------------
1980/12/28 00:00:00
1 row selected.
```

1.4.2.6 DDD

Description

Returns the day of the year, represented by a number from 1 to 366.

DDD cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'DDD' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'DDD' )
------------------------------------
363
1 row selected.
```

1.4.2.7 DY

Description

Returns the day of the week in abbreviated form (SUN, MON, TUE, …).

DY cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'DY' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'DY' )
-----------------------------------
SUN
1 row selected.
```

1.4.2.8 FF [1..6]

Description

Returns the fractional part of a second. The number of decimal places to return is determined by the number input after FF as part of the argument. If this number is omitted (i.e. "FF" is specified with no number following it), the element is handled the same as if "FF6" were specified.

Example

```
iSQL> SELECT TO_CHAR ( SYSDATE, 'FF5' ) FROM dual;
TO_CHAR ( SYSDATE, 'FF5' )
------------------------------
34528
1 row selected.

iSQL> CREATE TABLE T1(C1 DATE);
Create success.

iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.1', 'YYYY-MM-DD
HH:MI:SS.FF1'));
1 row inserted.
iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.12', 'YYYY-MM-DD
HH:MI:SS.FF2'));
1 row inserted.
iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.123', 'YYYY-MM-DD
HH:MI:SS.FF3'));
1 row inserted.
iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.1234', 'YYYY-MM-DD
HH:MI:SS.FF4'));
1 row inserted.
iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.12345', 'YYYY-MM-DD
HH:MI:SS.FF5'));
1 row inserted.
iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.123456', 'YYYY-MM-DD
HH:MI:SS.FF6'));
1 row inserted.
iSQL> INSERT INTO T1 VALUES(TO_DATE('2012-12-31 23:59:59.123456', 'YYYY-MM-DD
HH:MI:SS.FF'));
1 row inserted.

iSQL> SELECT TO_CHAR(C1, 'YYYY-MM-DD HH:MI:SS.FF') FROM T1;
TO_CHAR(C1, 'YYYY-MM-DD HH:MI:SS.FF')
------------------------------------------------------------------------
2012-12-31 23:59:59.100000
2012-12-31 23:59:59.120000
2012-12-31 23:59:59.123000
2012-12-31 23:59:59.123400
2012-12-31 23:59:59.123450
2012-12-31 23:59:59.123456
2012-12-31 23:59:59.123456
7 rows selected.
```

1.4.2.9 HH, HH24

Description

Returns the hour of the day in 24-hour format (i.e. returns a number from 0 to 23).

Example

```
iSQL> SELECT TO_CHAR ( TO_DATE( '2008-12-28 17:30:29', 'YYYY-MM-DD HH:MI:SS'
), 'HH' ) FROM dual;
TO_CHAR ( TO_DATE( '2008-12-28 17:30:29'
--------------------------------------------
17
1 row selected.

iSQL> SELECT TO_CHAR ( TO_DATE( '2008-12-28 17:30:29', 'YYYY-MM-DD
HH24:MI:SS' ), 'YYYY-MM-DD HH24:MI:SS' ) FROM dual;
TO_CHAR ( TO_DATE( '2008-12-28 17:30:29',
------------------------------------------
2008-12-28 17:30:29
1 row selected.
```

1.4.2.10 HH12

Description

Returns the hour of the day in 12-hour format (i.e. returns a number from 1 to 12).

This element cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( TO_DATE( '2008-12-28 17:30:29', 'YYYY-MM-DD HH:MI:SS'
), 'HH12' ) FROM dual;
TO_CHAR ( TO_DATE( '2008-12-28 17:30:29',
---------------------------------------------
05
1 row selected.

iSQL> SELECT TO_CHAR( TO_DATE ( '08-12-28 05:30:29', 'RR-MM-DD HH12:MI:SS' ),
'RR-MM-DD HH12:MI:SS') FROM dual;
TO_CHAR( TO_DATE ( '08-12-28 05:30:29', 'R
--------------------------------------------
08-12-28 05:30:29
1 row selected.
```

1.4.2.11 MI

Description

Returns a number ranging from 0 to 59, indicating the minutes portion of the input date.

Example

```
% export ALTIBASE_DATE_FORMAT="YYYY/MM/DD HH:MI:SS"

iSQL> SELECT TO_CHAR ( TO_DATE( '1980-12-28 17:30:29', 'YYYY-MM-DD HH:MI:SS'
), 'MI' ) FROM dual;
TO_CHAR ( TO_DATE( '1980-12-28 17:30:29'
--------------------------------------------
30
1 row selected.

iSQL> SELECT TO_DATE ( '05-12-28 14:30:29', 'RR-MM-DD HH:MI:SS' ) FROM dual;
```

```
TO_DATE ( '05-12-28 14:30:29', 'RR-MM-DD
------------------------------------------
2005/12/28 14:30:29
1 row selected.
```

1.4.2.12 MM

Description

Returns a number ranging from 01 to 12, indicating the month of the input date.

Example

```
iSQL> SELECT TO_CHAR ( TO_DATE( '1980-12-28 17:30:29', 'YYYY-MM-DD HH:MI:SS'
), 'MM' ) FROM dual;
TO_CHAR ( TO_DATE( '1980-12-28 17:30:29'
-------------------------------------------
12
1 row selected.

iSQL> SELECT TO_DATE ( '05-12-28 14:30:29', 'RR-MM-DD HH:MI:SS' ) FROM dual;
TO_DATE ( '05-12-28 14:30:29', 'RR-MM-DD
------------------------------------------
2005/12/28 14:30:29
1 row selected.
```

1.4.2.13 MON

Description

Returns the name of the month in upper case in abbreviated form (JAN, FEB, MAR, …).

Example

```
SQL> SELECT TO_CHAR (TO_DATE ('1995-12-05', 'YYYY-MM-DD'), 'MON') FROM dual;
TO_
---
DEC
```

1.4.2.14 MONTH

Description

Returns the name of the month in upper case. (JANUARY, FEBRUARY, …)

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'Month' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'Month' )
-------------------------------------
December
1 row selected.

iSQL> SELECT TO_DATE ( '05-APRIL-28 14:30:29', 'RR-MONTH-DD HH:MI:SS' ) FROM
dual;
```

```
TO_DATE ( '05-APRIL-28 14:30:29', 'RR-MO
-------------------------------------------
2005/04/28 14:30:29
1 row selected.
```

1.4.2.15 Q

Description

Returns a number ranging from 1 to 4, indicating the quarter of the year of the input date.

This element cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'Q' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'Q' )
----------------------------------
4
1 row selected.
```

1.4.2.16 RM

Description

Returns the month of the input date in Roman numerals (I, II, III, IV...).

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'RM' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'RM' )
-----------------------------------
XII
1 row selected.

iSQL> SELECT TO_DATE ('28-V-1980', 'DD-RM-YYYY') FROM dual;
TO_DATE ('28-V-1980', 'DD-RM-YYYY')
--------------------------------------
1980/05/28 00:00:00
1 row selected.
```

1.4.2.17 RR

Description

Returns the year of the input date as a 2-digit integer. When the year portion of the input date has 2 digits, if it is less than 50, 2000 is added to it (i.e. the 21st Century is assumed), whereas if it is greater than or equal to 50, 1900 is added to it before it is displayed. Therefore, the range of years that can be displayed is between 1950 – 2049.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-80', 'RR' ) FROM dual;
TO_CHAR ( '28-DEC-80', 'RR' )
---------------------------------
```

```
80
1 row selected.
iSQL> SELECT TO_DATE ( '28-DEC-80', 'DD-MON-RR' ) FROM dual;
TO_DATE ( '28-DEC-80', 'DD-MON-RR' )
---------------------------------------
1980/12/28 00:00:00
1 row selected.
```

1.4.2.18 RRRR

Description

Year (0 - 9999)

Returns the year of the input date as a 4-digit integer. When the year portion of the input date has 2 digits, if it is less than 50, 2000 is added to it (i.e. the 21st Century is assumed), whereas if it is greater than or equal to 50 and less than 100, 1900 is added to it before it is displayed. When the year portion of the input date has 4 digits, it is output without change.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'RRRR' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'RRRR' )
-------------------------------------
1980
1 row selected.

iSQL> select to_date('23-FEB-11', 'DD-MON-RRRR') from dual;
TO_DATE('23-FEB-11', 'DD-MON-RRRR')
--------------------------------------
2011/02/23 00:00:00
1 row selected.

iSQL> select to_date('23-FEB-100', 'DD-MON-RRRR') from dual;
TO_DATE('23-FEB-100', 'DD-MON-RRRR')
---------------------------------------
0100/02/23 00:00:00
1 row selected.
```

1.4.2.19 SS

Description

Returns a number ranging from 0 to 59, indicating the seconds portion of the input date.

Example

```
iSQL> SELECT TO_CHAR ( TO_DATE( '1980-12-28 17:30:29', 'YYYY-MM-DD HH:MI:SS' 
), 'SS' ) FROM dual;
TO_CHAR ( TO_DATE( '1980-12-28 17:30:29'
--------------------------------------------
29
1 row selected.

iSQL> SELECT TO_DATE ( '05-12-28 14:30:29', 'RR-MM-DD HH:MI:SS' ) FROM dual;
TO_DATE ( '05-12-28 14:30:29', 'RR-MM-DD
-------------------------------------------
2005/12/28 14:30:29
```

```
1 row selected.
```

1.4.2.20 SSSSS

Description

Returns a number ranging from 0 to 86399, indicating the number of seconds that have passed since midnight.

Example

```
iSQL> SELECT TO_CHAR ( TO_DATE( '1980-12-28 17:30:29', 'YYYY-MM-DD
HH24:MI:SS' ), 'SSSSS' ) FROM dual;
TO_CHAR ( TO_DATE( '1980-12-28 17:30:29'
--------------------------------------------
62940
1 row selected.

iSQL> SELECT TO_DATE('1980-12-28 12345', 'YYYY-MM-DD SSSSS') FROM dual;
TO_DATE('1980-12-28 12345', 'YYYY-MM-DD
-------------------------------------------
1980/12/28 03:25:45
1 row selected.
```

1.4.2.21 SSSSSS

Description

Returns the fractional part of a second.

Example

```
iSQL> SELECT TO_CHAR (SYSDATE, 'SSSSSS') FROM dual;
TO_CHAR (SYSDATE, 'SSSSSS')
-------------------------------
490927
1 row selected.

iSQL> SELECT TO_CHAR ( TO_DATE('1980-12-28 123456', 'YYYY-MM-DD SSSSSS'),
'SSSSSS' ) FROM dual;
TO_CHAR ( TO_DATE('1980-12-28 123456', '
--------------------------------------------
123456
1 row selected.
```

1.4.2.22 SSSSSSSS

Description

Returns the integer and fractional parts of the number of seconds in the input date, expressed as an 8-digit integer ranging from 0 to 59999999. The first two digits indicate the number of seconds, and the remaining 6 digits represent the fractional part of the second.

Example

```
iSQL> SELECT TO_CHAR (SYSDATE, 'SSSSSSSS') FROM dual;
TO_CHAR (SYSDATE, 'SSSSSSSS')
---------------------------------
48987403
1 row selected.

iSQL> SELECT TO_DATE ( '12.345678', 'SS.SSSSSS') FROM dual;
TO_DATE ( '12.345678', 'SS.SSSSSS')
--------------------------------------
2005/12/01 00:00:12
1 row selected.

iSQL> SELECT TO_CHAR( TO_DATE( '12.345678', 'SS.SSSSSS'), 'SSSSSS') FROM 
dual;
TO_CHAR( TO_DATE( '12.345678', 'SS.SSSSS
--------------------------------------------
345678
1 row selected.
```

1.4.2.23 WW

Description

Returns a number ranging from 1 to 54, indicating the week of the year. The period from January 1 to the first Saturday is considered the first week of the year.

This element cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'WW' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'WW' )
-----------------------------------
53
1 row selected.
```

1.4.2.24 W

Description

Returns a number ranging from 1 to 6, indicating the week of the month. The period from the first day of the month to the first Saturday is considered the first week of the year.

This element cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'W' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'W' )
----------------------------------
5
1 row selected.
```

1.4.2.25 Y,YYY

Description

Returns the year of the input date. A comma can be inserted at any place within a number representing the year, including the very beginning or end.

This element cannot be used with the TO_DATE function.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'Y,YYY' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'Y,YYY' )
-------------------------------------
1,980
1 row selected.
```

1.4.2.26 YYYY

Description

Handles a positive four-digit number ranging from 0 - 9999 as the year.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'YYYY' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'YYYY' )
-------------------------------------
1980
1 row selected.

iSQL> SELECT TO_DATE ( '28-DEC-1980', 'DD-MON-YYYY' ) FROM dual;
TO_DATE ( '28-DEC-1980', 'DD-MON-YYYY' )
-------------------------------------------
1980/12/28 00:00:00
1 row selected.
```

1.4.2.27 YY

Description

Returns the last two digits of the year. The 21st Century is assumed, so 2000 is added to it to obtain the actual year, which can range from 2000 to 2099.

Example

```
iSQL> SELECT TO_CHAR ( '28-DEC-1980', 'YY' ) FROM dual;
TO_CHAR ( '28-DEC-1980', 'YY' )
-----------------------------------
80
1 row selected.

iSQL> SELECT TO_DATE ( '28-DEC-80', 'DD-MON-YY' ) FROM dual;
TO_DATE ( '28-DEC-80', 'DD-MON-YY' )
---------------------------------------
2080/12/28 00:00:00
```

```
1 row selected.
```

Example

```
iSQL> CREATE TABLE timetbl(i1 INTEGER, t1 DATE, etc VARCHAR(10));
Create success.
iSQL> INSERT INTO timetbl VALUES (1, SYSDATE, 'Start');
1 row inserted.

iSQL> INSERT INTO timetbl VALUES (2, TO_DATE('2003-02-20 12:15:50', 'YYYY-MM-
DD HH:MI:SS'), 'The end');
1 row inserted.

iSQL> SELECT TO_CHAR(T1, 'YYYY YY MM MON Mon mon DD HH MI SS SSSSSS D DDD')
Date_format FROM timetbl WHERE I1 = 2;
DATE_FORMAT
------------------------------------------------
2003 03 02 FEB Feb feb 20 12 15 50 000000 5 051
1 row selected.
```

1.4.2.28 The RR, RRRR, YY, and YYYY Date Format Elements Compared

Please refer to the descriptions of the respective format elements.

- [YYYY]: The number is treated as a year, without change.

  '23-FEB-5' = February 23, 0005

  '23-FEB-05' = February 23, 0005

  '23-FEB-2005'= February 23, 2005

  '23-FEB-95' = February 23, 0095

- [YY]: 2000 is added to YY to obtain the year.

  '23-FEB-5' = February 23, 2005

  '23-FEB-05' = February 23, 2005

  '23-FEB-2005'= Error

  '23-FEB-95' = February 23, 2095

  '23-FEB-05' = February 23, 2005

  '23-FEB-2005'= Error

  '23-FEB-95' = February 23, 2095

- [RRRR]: A number greater than 100 is taken as the year without change. If the input number has one or two digits, if it is < 50, 2000 is added to it, and if it is >= 50 and < 100, 1900 is added to it before it is output.

  '23-FEB-5': February 23, 2005

  '23-FEB-05': February 23, 2005

'23-FEB-2005': February 23, 2005

'23-FEB-95': February 23, 1995

'23-FEB-100': February 23, 0100

'23-FEB-0005': February 23, 0005

- [RR]: If the input number is < 50, 2000 is added to it, whereas if the input number is >= 50 and < 100, 1900 is added to it before it is output.

  '23-FEB-5': February 23, 2005

  '23-FEB-05': February 23, 2005

  '23-FEB-2005': Error

  '23-FEB-95': February 23, 1995

1.4.2.29 YYY

Description

The last 3 digits of the year. As the 21st Century is assumed, 2000 is added to it to obtain the actual year, which can range from 2000 to 2099.

1.4.2.30 Y

Description

The final digit of the year. As the 21st Century is assumed, 2000 is added to it to obtain the actual year, which can range from 2000 to 2099.

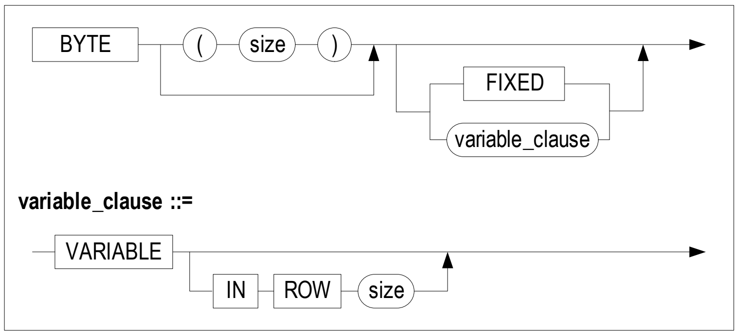
1.5 Binary Types

Large and unstructured data such as text, images, video, and spatial data can be stored as binary data. ALTIBASE HDB supports the following binary types:

- BYTE
- NIBBLE
- BIT
- VARBIT

1.5.1 BYTE

1.5.1.1 Syntax Diagram



1.5.1.2 Syntax

```
BYTE [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.5.1.3 Description

This is a binary data type having a specified fixed length. The default size of a BYTE column is 1 byte. The maximum length of a BYTE column is 32000 bytes. The data can be expressed in hexadecimal format using a combination of alphabet and numeric characters, such as '0FAE13.' The allowable alphanumeric characters are 0 (zero) to 9 and A to F.

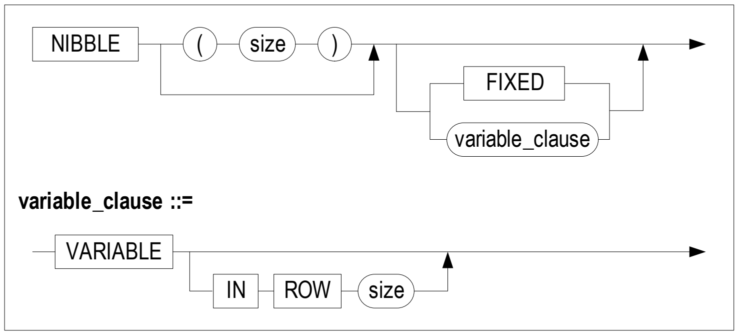
When data are stored in or retrieved from a BYTE column, the specified size of the column must be used. Two characters can be stored in one byte. For example, for a column specified as BYTE(3), a range of values from '000000' to 'FFFFFF' can be input.

When the lower case letters 'a' through 'f' are input, they are converted into upper-case letters.

For more information on the FIXED and VARIABLE clauses, please refer to the sections earlier in this chapter entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause section.

1.5.2 NIBBLE

1.5.2.1 Syntax Diagram



1.5.2.2 Syntax

```
NIBBLE [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.5.2.3 Description

This is a binary data type that varies in length up to the specified size. The default size of a NIBBLE column is that of a single character, and the maximum size is 254nibbles.

The data can be expressed in hexadecimal format using a combination of alphabet and numeric characters. The allowable alphanumeric characters are 0 (zero) to 9 and A to F. Unlike the BYTE type, only one character can be entered into one nibble.

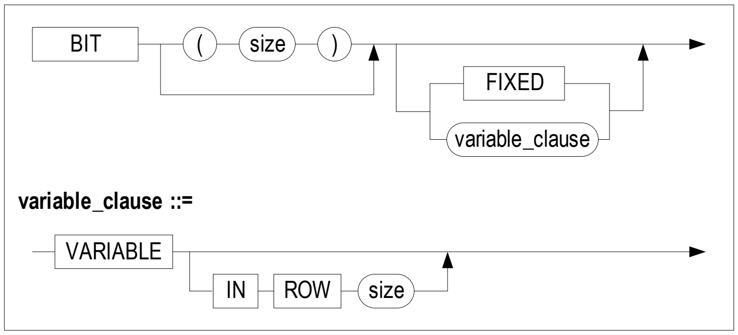
For example, for NIBBLE (6), '000000' to 'FFFFFF' can be inserted.

When the lower case letters 'a' through 'f' are input, they are converted into upper-case letters.

For more information on the FIXED and VARIABLE clauses, please refer to the sections earlier in this chapter entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.5.3 BIT

1.5.3.1 Syntax Diagram



1.5.3.2 Syntax

```
BIT [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.5.3.3 Description

This is a binary data type that has a fixed length and consists only of 0's and 1's. The default size of a BIT column is one bit. Its maximum size is 64000 bits.

If an attempt is made to input a string that is longer than the specified length, an 'Invalid data type length' error will be raised. If a string shorter than the specified length is input, the space to the right of the input data is populated with 0's. If a value other than 0 or 1 is input, an 'Invalid literal' error is raised.

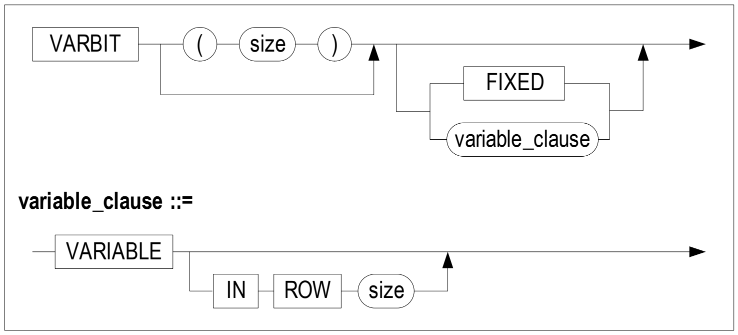
For more information on the FIXED and VARIABLE clauses, please refer to the sections earlier in this chapter entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.5.3.4 Example

```
iSQL> CREATE TABLE T1 ( I1 BIT(1), I2 BIT(5) );
Create success.
iSQL> INSERT INTO T1 VALUES ( BIT'1', BIT'011' );
1 row inserted.

iSQL> SELECT TO_CHAR(I1), TO_CHAR(I2) FROM T1;
TO_CHAR(I1) TO_CHAR(I2)
-----------------------------
1 01100
1 row selected.

iSQL> INSERT INTO T1 VALUES ( BIT'1111', BIT'011' );
[ERR-2100D : Invalid data type length]
iSQL> INSERT INTO T1 VALUES ( BIT'1', BIT'1234' );
[ERR-21011 : Invalid literal]
```

1.5.4 VARBIT

1.5.4.1 Syntax Diagram



1.5.4.2 Syntax

```
VARBIT [(size)] [[FIXED |] VARIABLE ( IN ROW size ) ]
```

1.5.4.3 Description

This is a binary data type that has a variable length and consists only of 0's and 1's. The default size of a BIT column is one bit. Its maximum size is 64000 bits.

If an attempt is made to input a string that is longer than the specified length, an "Invalid data type length" error will be raised. If a string shorter than the specified length is input, the space to the right of the input data is populated with 0's. If a value other than 0 or 1 is input, an 'Invalid literal' error is raised.

For more information on the FIXED and VARIABLE clauses, please refer to the sections earlier in this chapter entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.5.4.4 Example

```
iSQL> CREATE TABLE T1 ( I1 VARBIT(1), I2 VARBIT(5) );
Create success.
iSQL> INSERT INTO T1 VALUES ( VARBIT'1', VARBIT'011' );
1 row inserted.

iSQL> SELECT TO_CHAR(I1), TO_CHAR(I2) FROM T1;
TO_CHAR(I1) TO_CHAR(I2)
-----------------------------
1 011
1 row selected.

iSQL> INSERT INTO T1 VALUES ( VARBIT'1111', VARBIT'011' );
[ERR-2100D : Invalid data type length]
iSQL> INSERT INTO T1 VALUES ( VARBIT'1', VARBIT'1234' );
[ERR-21011 : Invalid literal]
```

1.6 LOB Data Type

1.6.1 Overview

The LOB (which stands for Large OBject) data type is for holding large amounts of data. Up to 2 GB can be stored in one column of LOB data. Unlike other data types, the length of a LOB column does not need to be specified when a table is created. Additionally, more than one LOB type column can be defined in a table.

The LOB data type is divided into the Binary Large Object (BLOB) type, which is for holding binary data such as image and video files, and the Character Large Object (CLOB) type, which is for holding string data.

1.6.2 The Features of LOB

The LOB data type provided with ALTIBASE HDB has the following features:

- Data Storage Functions
- Partial Read
- Disk LOB Partitioning

1.6.2.1 Data Storage Functions

CLOB or BLOB data can be stored using the ODBC SQLPutLob function or using the setBlob or set-Clob methods in JDBC.

1.6.2.2 Partial Read

It is possible to read only a desired portion of LOB data. A specific amount of data, offset a specific distance from the beginning of the file, can be read using the SQLGetLob function in ALTIBASE HDB ODBC.

1.6.2.3 Disk LOB Partitioning

Disk LOB data can be stored in a disk tablespace other than the one in which the table is stored. This tablespace can be configured in a method similar to partitioning. For more information about disk LOB partitioning, please refer to the description of the CREATE TABLE statement in the *SQL Reference*.

1.6.3 Storing LOB Columns

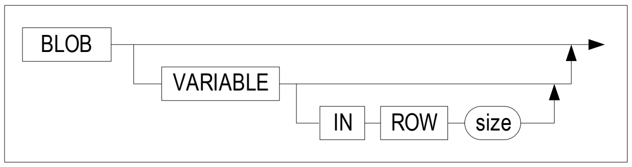
In most cases, LOB data are stored in a variable area, away from the rest of the record. However, in cases where the amount of data stored in the LOB column is not big, the column can be stored in an area that is contiguous with the rest of the record (i.e. in the fixed area) using the 'in row' option. Note that this is possible for memory tables only; regardless of their size, LOB data in disk tables are always stored in a separate, variable area.

Because the amount of LOB column data that is stored in the variable area is typically very large, storing it in the same tablespace as the rest of the record has a negative impact on the efficiency of usage of space.

In a disk table, LOB column data can be stored in a tablespace other than the one containing the table to which the LOB column belongs. However, in a memory table, LOB column data cannot be stored separately, and thus are stored in the same tablespace as the table.

1.6.4 BLOB

1.6.4.1 Syntax Diagram



1.6.4.2 Syntax

BLOB [VARIABLE (IN ROW size)]

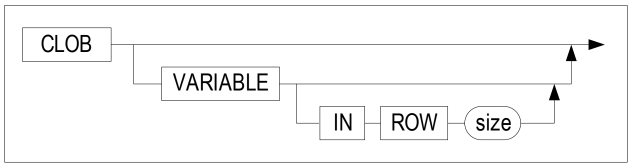
1.6.4.3 Description

BLOB is a binary data type that can vary in length up to 2 GB and is intended for use in storing large amounts of binary data.

For more information on the VARIABLE clause, please refer to the sections earlier in this chapter entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.6.5 CLOB

1.6.5.1 Syntax Diagram



1.6.5.2 Syntax

```
CLOB [ VARIABLE ( IN ROW size ) ]
```

1.6.5.3 Description

CLOB is a character data type that can vary in length up to 2 GB and is intended for use in storing large amounts of character data.

For more information on the VARIABLE clause, please refer to the sections earlier in this chapter entitled 1.1.5 The FIXED and VARIABLE Options and 1.1.6 The IN ROW clause.

1.6.6 Restrictions

- LOB type columns can't be used with stored procedures or triggers.
- LOB type columns can't be used with cursors.
- LOB type columns can't be used in volatile tables or disk temporary tablespaces.
- LOB columns associated with tables in discarded tablespaces cannot be accessed.
- LOB type columns cannot be used for partitioning conditions, because in order to partition a column it must be possible to perform comparisons on the data in the column.
- Indexes cannot be created for LOB columns.
- It is possible to define a NOT NULL constraint for a LOB type column. However, when an insert attempt is made, a constraint violation error may be raised while the ALTIBASE HDB server is internally processing the data. Therefore, it is recommended that the NOT NULL constraint not be used with LOB type columns.

1.7 Spatial Types

The only geometry data type that is defined and supported for use with SQL in ALTIBASE HDB is the GEOMETRY data type. The Geometry data type comprises the following seven subtypes:

- Point
- LineString
- Polygon
- GeomCollection
- MultiPolygon
- MultiLineString
- MultiPoint

For more information about the geometry datatype, please refer to the *ALTIBASE HDB Spatial SQL Reference*.

2 ALTIBASE HDB Properties

ALTIBASE HDB server can be run in various modes. The altibase.properties file is used to make ALTIBASE HDB server environment settings. The altibase.properties file contains all elements related to the operation and adjustment of the ALTIBASE HDB server.

In this chapter, the ALTIBASE HDB properties that must be set and managed in order to configure and use ALTIBASE HDB in the manner that is suitable for the user's purposes will be explained.

This chapter contains the following sections:

2.1 Configuration

There are three ways to make ALTIBASE HDB server environment settings. The first method involves making changes to the ALTIBASE HDB properties file, which is located at $ALTIBASE_HOME/conf/altibase.properties. Because this method of configuration is static, meaning that changes can only be made while ALTIBASE HDB is not running, after setting given variables in the properties file to particular values, it will be necessary to restart the ALTIBASE HDB server in order for the changes to take effect.

The second method is dynamic, meaning that configuration changes of ALTIBASE HDB can be made even while ALTIBASE HDB server is running. Although this method confers the advantage of being able to make and apply changes without shutting down the server, it is not possible for all properties. For properties that can be dynamically changed, the ALTER SYSTEM or ALTER SESSION statements can be used to apply the configuration changes to the entire ALTIBASE HDB server or to individual sessions, respectively.

The third method of configuring the ALTIBASE HDB environment is through the use of operating system environment variables. Like the method involving the altibase.properties file, this configuration method is also static. Properties that are read-only or that can only have a single value can be set in this way. After specifying the environment variable as ALTIBASE_property_name, it will be necessary to reboot the database server in order to implement the changes.

Here is an example:

```
$ export ALTIBASE_DEFAULT_DATE_FORMAT=YYYY/MM/DD
```

The precedence of the property-setting methods is as follows:

1. environment variables settings
2. altibase.properties file settings
3. default system values

As can be seen in the following example, when properties are set, because environment variables take highest precedence, the value of DEFAULT_DATE_FORMAT in the altibase.properties file is ignored, and the value of the environment variable is used.

```
$ export ALTIBASE_DEFAULT_DATE_FORMAT=YYYY-MM-DD
```

altibase.properties

```
DEFAULT_DATE_FORMAT=YYYY-MM-DD
```

Similarly, in the following example, NLS_USE in the altibase.properties file is ignored, and UTF-8, which is specified by the NLS_USE environment variable, is used, because environment variables have the highest priority.

```
$ export ALTIBASE_NLS_USE=UTF8
```

altibase.properties

```
NLS_USE = KO16KSC5601
```

The property file for configuring the ALTIBASE HDB server is called "altibase.properties" and is located in the conf subdirectory of ALTIBASE_HOME. The properties therein are broadly grouped as follows:

- database initialization properties
- performance properties
- session properties
- transaction properties
- backup and recovery properties
- replication properties
- message logging properties
- Database Link properties
- DataPort properties
- other properties

The following table lists all ALTIBASE HDB properties. For reference, each group in the table has the following meaning:

- D: database initialization properties
- P: performance properties
- S: session properties
- T: transaction properties
- B: backup and recovery properties
- R: replication properties
- M: message logging properties
- L: Database Link properties
- O: DataPort properties
- E: other properties

The values in the "Alter Level" column have the following meaning:

- SESSION: the property can be changed using an ALTER SESSION statement.
- SYSTEM: the property can be changed using an ALTER SYSTEM statement.
- BOTH: the property can be changed using either an ALTER SESSION or an ALTER SYSTEM statement.

Group	Class	Property Name	Alter Level
D		BUFFER_AREA_CHUNK_SIZE	None
		BUFFER_AREA_SIZE	SYSTEM
		BUFFER_CHECKPOINT_LIST_CNT	None
		BUFFER_FLUSHER_CNT	None
		BUFFER_FLUSH_LIST_CNT	None
		BUFFER_HASH_BUCKET_DENSITY	None
		BUFFER_HASH_CHAIN_LATCH_DENSITY	None
		BUFFER_LRU_LIST_CNT	None
		BUFFER_PREPARE_LIST_CNT	None
		BULKIO_PAGE_COUNT_FOR_DIRECT_PATH_INSERT	SYSTEM
		COMPRESSION_RESOURCE_GC_SECOND	None
		DB_NAME	None
		DDL_SUPPLEMENTAL_LOG_ENABLE	SYSTEM
		DEFAULT_DISK_DB_DIR	None
		DEFAULT_MEM_DB_FILE_SIZE	None
		DEFAULT_SEGMENT_MANAGEMENT_TYPE	None
		DEFAULT_SEGMENT_STORAGE_INITEXTENTS	None
		DEFAULT_SEGMENT_STORAGE_MAXEXTENTS	None
		DEFAULT_SEGMENT_STORAGE_MINEXTENTS	None
		DEFAULT_SEGMENT_STORAGE_NEXTEXTENTS	None
		DIRECT_PATH_BUFFER_PAGE_COUNT	SYSTEM
		DISK_INDEX_UNBALANCED_SPLIT_RATE	SYSTEM
		DISK_LOB_COLUMN_IN_ROW_SIZE	None
		DOUBLE_WRITE_DIRECTORY	None
		DOUBLE_WRITE_DIRECTORY_COUNT	None
		DRDB_FD_MAX_COUNT_PER_DATAFILE	SYSTEM
		EXPAND_CHUNK_PAGE_COUNT	None
		FULL_SCAN_USE_BUFFER_POOL	SYSTEM
		LOGANCHOR_DIR	None
		LOG_DIR	None
		LOG_FILE_SIZE	None
		MAX_CLIENT	None
		MEM_DB_DIR	None
		MEM_MAX_DB_SIZE	None
		MEMORY_INDEX_BUILD_RUN_SIZE	SYSTEM
		MEMORY_INDEX_BUILD_VALUE_LENGTH_THRESHOLD	SYSTEM
		MEMORY_LOB_COLUMN_IN_ROW_SIZE	None
		MEMORY_VARIABLE_COLUMN_IN_ROW_SIZE	None
		MEM_SIZE_CLASS_COUNT	None
		MIN_COMPRESSION_RESOURCE_COUNT	None
		MIN_LOG_RECORD_SIZE_FOR_COMPRESS	SYSTEM
		MIN_PAGES_ON_DB_FREE_LIST	None
		MIN_PAGES_ON_TABLE_FREE_LIST	SYSTEM
		PCTFREE	None
		PCTUSED	None
		QP_MEMORY_CHUNK_SIZE	None
		SECURITY_ECC_POLICY_NAME	SYSTEM
		SECURITY_MODULE_LIBRARY	SYSTEM
		SECURITY_MODULE_NAME	SYSTEM
		SHM_DB_KEY	SYSTEM
		STARTUP_SHM_CHUNK_SIZE	None
		ST_OBJECT_BUFFER_SIZE	BOTH
		SYS_DATA_FILE_INIT_SIZE	None
		SYS_DATA_FILE_MAX_SIZE	None
		SYS_DATA_FILE_NEXT_SIZE	None
		SYS_DATA_TBS_EXTENT_SIZE	None

Group	Class	Property Name	Alter Level
		SYS_TEMP_FILE_INIT_SIZE	None
		SYS_TEMP_FILE_MAX_SIZE	None
		SYS_TEMP_FILE_NEXT_SIZE	None
		SYS_TEMP_TBS_EXTENT_SIZE	None
		SYS_UNDO_FILE_INIT_SIZE	None
		SYS_UNDO_FILE_MAX_SIZE	None
		SYS_UNDO_FILE_NEXT_SIZE	None
		SYS_UNDO_TBS_EXTENT_SIZE	None
		TABLE_BACKUP_FILE_BUFFER_SIZE	None
		TABLE_COMPACT_AT_SHUTDOWN	SYSTEM
		TEMP_PAGE_CHUNK_COUNT	None
		TRCLOG_DETAIL_SCHEMA	BOTH
		USER_DATA_FILE_INIT_SIZE	None
		USER_DATA_FILE_MAX_SIZE	None
		USER_DATA_FILE_NEXT_SIZE	None
		USER_DATA_TBS_EXTENT_SIZE	None
		USER_TEMP_FILE_INIT_SIZE	None
		USER_TEMP_FILE_MAX_SIZE	None
		USER_TEMP_FILE_NEXT_SIZE	None
		USER_TEMP_TBS_EXTENT_SIZE	None
		VOLATILE_MAX_DB_SIZE	None
P		AGER_WAIT_MAXIMUM	None
		AGER_WAIT_MINIMUM	None
		BUFFER_VICTIM_SEARCH_INTERVAL	SYSTEM
		BUFFER_VICTIM_SEARCH_PCT	SYSTEM
		CHECKPOINT_BULK_SYNC_PAGE_COUNT	SYSTEM
		CHECKPOINT_BULK_WRITE_PAGE_COUNT	SYSTEM
		CHECKPOINT_BULK_WRITE_SLEEP_SEC	SYSTEM
		CHECKPOINT_BULK_WRITE_SLEEP_USEC	SYSTEM
		CHECKPOINT_FLUSH_COUNT	SYSTEM
		CHECKPOINT_FLUSH_MAX_GAP	SYSTEM
		CHECKPOINT_FLUSH_MAX_WAIT_SEC	SYSTEM
		CM_BUFFER_MAX_PENDING_LIST	None
		DATABASE_IO_TYPE	None
		DATAFILE_WRITE_UNIT_SIZE	SYSTEM
		DB_FILE_MULTIPAGE_READ_COUNT	SYSTEM
		DEFAULT_FLUSHER_WAIT_SEC	SYSTEM
		DIRECT_IO_ENABLED	None
		DISK_INDEX_BUILD_MERGE_PAGE_COUNT	SYSTEM
		EXECUTE_STMT_MEMORY_MAXIMUM	SYSTEM
		FAST_START_IO_TARGET	SYSTEM
		FAST_START_LOGFILE_TARGET	SYSTEM
		HIGH_FLUSH_PCT	SYSTEM
		HOT_LIST_PCT	SYSTEM
		HOT_TOUCH_CNT	SYSTEM

Group	Class	Property Name	Alter Level
P		INDEX_BUILD_THREAD_COUNT	SYSTEM
		INDEX_INITRANS	None
		INDEX_MAXTRANS	None
		INSPECTION_LARGE_HEAP_THRESHOLD	SYSTEM
		LFG_GROUP_COMMIT_INTERVAL_USEC	None
		LFG_GROUP_COMMIT_RETRY_USEC	None
		LFG_GROUP_COMMIT_UPDATE_TX_COUNT	None
		LOCK_ESCALATION_MEMORY_SIZE	SYSTEM
		LOG_FILE_GROUP_COUNT	None
		LOG_IO_TYPE	None
		LOW_FLUSH_PCT	SYSTEM
		LOW_PREPARE_PCT	SYSTEM
		MAX_FLUSHER_WAIT_SEC	SYSTEM
		MULTIPLEXING_CHECK_INTERVAL	SYSTEM
		MULTIPLEXING_MAX_THREAD_COUNT	SYSTEM
		MULTIPLEXING_THREAD_COUNT	None
		NORMALFORM_MAXIMUM	BOTH
		OPTIMIZER_MODE	BOTH
		PARALLEL_LOAD_FACTOR	None
		PREPARE_STMT_MEMORY_MAXIMUM	SYSTEM
		REFINE_PAGE_COUNT	None
		SHM_PAGE_COUNT_PER_KEY	SYSTEM
		SORT_AREA_SIZE	SYSTEM
		SQL_PLAN_CACHE_BUCKET_CNT	None
		SQL_PLAN_CACHE_HOT_REGION_LRU_RATIO	SYSTEM
		SQL_PLAN_CACHE_PREPARED_EXECUTION_CONTEXT_CNT	SYSTEM
		SQL_PLAN_CACHE_SIZE	SYSTEM
		STATEMENT_LIST_PARTIAL_SCAN_COUNT	SYSTEM
		TABLE_INITRANS	None
		TABLE_LOCK_ENABLE	SYSTEM
		TABLE_MAXTRANS	None
		TIMER_RUNNING_LEVEL	None
		TIMED_STATISTICS	SYSTEM
		TIMER_THREAD_RESOLUTION	SYSTEM
		TOUCH_TIME_INTERVAL	SYSTEM
		TRANSACTION_SEGMENT_COUNT	SYSTEM
		TRX_UPDATE_MAX_LOGSIZE	BOTH

Group	Class	Property Name	Alter Level
S	Normal	CM_DISCONN_DETECT_TIME	None
		DEFAULT_THREAD_STACK_SIZE	None
		IPC_CHANNEL_COUNT	None
		IPC_PORT_NO	None
		MAX_LISTEN	None
		MAX_STATEMENTS_PER_SESSION	BOTH
		NET_CONN_IP_STACK	None
		NLS_NCHAR_CONV_EXCP	SESSION
		NLS_COMP	None
		NLS_USE	None
		PORT_NO	None
		PSM_FILE_OPEN_LIMIT	SYSTEM
		SERVICE_THREAD_STACK_SIZE	None
		USE_MEMORY_POOL	None
		XA_HEURISTIC_COMPLETE	None
	Time-Out	BLOCK_ALL_TX_TIME_OUT	SYSTEM
		DDL_LOCK_TIMEOUT	SYSTEM
		FETCH_TIMEOUT	BOTH
		IDLE_TIMEOUT	BOTH
		LINKER_CONNECT_TIMEOUT	None
		LINKER_RECEIVE_TIMEOUT	None
		LOGIN_TIMEOUT	SYSTEM
		MULTIPLEXING_POLL_TIMEOUT	SYSTEM
		QUERY_TIMEOUT	BOTH
		REMOTE_SERVER_CONNECT_TIMEOUT	None
		REPLICATION_CONNECT_TIMEOUT	SYSTEM
		REPLICATION_LOCK_TIMEOUT	SYSTEM
		REPLICATION_RECEIVE_TIMEOUT	SYSTEM
		REPLICATION_SENDER_SLEEP_TIMEOUT	SYSTEM
		REPLICATION_SYNC_LOCK_TIMEOUT	SYSTEM
		SHUTDOWN_IMMEDIATE_TIMEOUT	SYSTEM
		UTRANS_TIMEOUT	BOTH
		XA_INDOUBT_TX_TIMEOUT	None
T		AUTO_COMMIT	BOTH
		ISOLATION_LEVEL	None
		TRANSACTION_TABLE_SIZE	SYSTEM
B		ARCHIVE_DIR	None
		ARCHIVE_FULL_ACTION	None
		ARCHIVE_THREAD_AUTOSTART	None
		CHECKPOINT_ENABLED	None
		CHECKPOINT_INTERVAL_IN_LOG	SYSTEM
		CHECKPOINT_INTERVAL_IN_SEC	SYSTEM
		COMMIT_WRITE_WAIT_MODE	BOTH
		LOG_BUFFER_TYPE	None
		PREPARE_LOG_FILE_COUNT	None

Group	Class	Property Name	Alter Level
R		REPLICATION_ACK_XLOG_COUNT	None
		REPLICATION_CONNECT_RECEIVE_TIMEOUT	SYSTEM
		REPLICATION_DDL_ENABLE	SYSTEM
		REPLICATION_FAILBACK_INCREMENTAL_SYNC	None
		REPLICATION_HBT_DETECT_HIGHWATER_MARK	SYSTEM
		REPLICATION_HBT_DETECT_TIME	SYSTEM
		REPLICATION_INSERT_REPLACE	SYSTEM
		REPLICATION_KEEP_ALIVE_CNT	None
		REPLICATION_LOG_BUFFER_SIZE	None
		REPLICATION_MAX_LISTEN	None
		REPLICATION_MAX_LOGFILE	SYSTEM
		REPLICATION_NET_CONN_IP_STACK	None
		REPLICATION_POOL_ELEMENT_COUNT	SYSTEM
		REPLICATION_POOL_ELEMENT_SIZE	SYSTEM
		REPLICATION_PORT_NO	None
		REPLICATION_PREFETCH_LOGFILE_COUNT	SYSTEM
		REPLICATION_RECOVERY_MAX_LOGFILE	None
		REPLICATION_RECOVERY_MAX_TIME	None
		REPLICATION_SENDER_AUTO_START	None
		REPLICATION_SENDER_SLEEP_TIME	None
		REPLICATION_SERVICE_WAIT_MAX_LIMIT	None
		REPLICATION_SYNC_LOG	None
		REPLICATION_SYNC_TUPLE_COUNT	SYSTEM
		REPLICATION_TIMESTAMP_RESOLUTION	SYSTEM
		REPLICATION_UPDATE_REPLACE	SYSTEM
		REPLICATION_EAGER_PARALLEL_FACTOR	None
		REPLICATION_COMMIT_WRITE_WAIT_MODE	SYSTEM
		REPLICATION_SERVER_FAILBACK_MAX_TIME	None

Group	Class	Property Name	Alter Level
M		ALL_MSGLOG_FLUSH	SYSTEM
		DL_MSGLOG_COUNT	None
		DL_MSGLOG_DIR	None
		DL_MSGLOG_FILE	None
		DL_MSGLOG_FLAG	SYSTEM
		DL_MSGLOG_SIZE	None
		LK_MSGLOG_COUNT	None
		LK_MSGLOG_DIR	None
		LK_MSGLOG_FILE	None
		LK_MSGLOG_FLAG	SYSTEM
		LK_MSGLOG_SIZE	None
		NETWORK_ERROR_LOG	SYSTEM
		QP_MSGLOG_COUNT	None
		QP_MSGLOG_DIR	None
		QP_MSGLOG_FILE	None
		QP_MSGLOG_FLAG	SYSTEM
		QP_MSGLOG_SIZE	None
		QUERY_PROF_FLAG	SYSTEM
		RP_MSGLOG_COUNT	None
		RP_MSGLOG_DIR	None
		RP_MSGLOG_FILE	None
		RP_MSGLOG_FLAG	SYSTEM
		RP_MSGLOG_SIZE	None
		SERVER_MSGLOG_COUNT	None
		SERVER_MSGLOG_DIR	None
		SERVER_MSGLOG_FILE	None
		SERVER_MSGLOG_FLAG	SYSTEM
		SERVER_MSGLOG_SIZE	None
		SM_MSGLOG_COUNT	None
		SM_MSGLOG_DIR	None
		SM_MSGLOG_FILE	None
		SM_MSGLOG_FLAG	SYSTEM
		SM_MSGLOG_SIZE	None
		TRCLOG_DETAIL_PREDICATE	SYSTEM
		XA_MSGLOG_DIR	None
		XA_MSGLOG_FILE	None
		XA_MSGLOG_FLAG	SYSTEM
		XA_MSGLOG_SIZE	None
L		AUTO_REMOTE_EXEC	BOTH
		DBLINK_ENABLE	None
		LINKER_LINK_TYPE	None
		LINKER_PORT_NO	None
		LINKER_SQLLEN_SIZE	None
		LINKER_THREAD_COUNT	None
		LINKER_THREAD_SLEEP_TIME	None
		MAX_DBLINK_COUNT	None
O		DATAPORT_FILE_DIRECTORY	SYSTEM
		DATAPORT_IMPORT_COMMIT_UNIT	SYSTEM
		DATAPORT_IMPORT_STATEMENT_UNIT	SYSTEM
E		ACCESS_LIST	None
		ADMIN_MODE	SYSTEM
		CHECK_MUTEX_DURATION_TIME_ENABLE	SYSTEM
		DEFAULT_DATE_FORMAT	None
		EXEC_DDL_DISABLE	SYSTEM
		QUERY_STACK_SIZE	BOTH
		REMOTE_SYSDBA_ENABLE	SYSTEM
		SELECT_HEADER_DISPLAY	BOTH

In this chapter, each property is explained as follows:

- Property Name
- Data Type
- Default Value
- Attributes (e.g. read-only vs. read-write, single vs. multiple values)
- Range (maximum and minimum possible values)
- Description

2.2 Database Initialization Properties

2.2.1 BUFFER_AREA_CHUNK_SIZE

2.2.1.1 Data Type

Unsigned Long

2.2.1.2 Default Value

33554432 (32MB)

2.2.1.3 Attributes

Read-Only, Single Value

2.2.1.4 Range

[8192, 2^{64} - 1]

2.2.1.5 Description

This indicates the unit, in bytes, by which the buffer size is incremented. When the buffer size is increased, it is increased in multiples of this number. This property can't be changed while the server is running.

2.2.2 BUFFER_AREA_SIZE

2.2.2.1 Data Type

Unsigned Long

2.2.2.2 Default Value

134217728 (128MB)

2.2.2.3 Attributes

Read-Write, Single Value

2.2.2.4 Range

[8192, 2^{64} - 1]

2.2.2.5 Description

This indicates the total memory size, in bytes, used by the buffer pool of ALTIBASE HDB. The value specified by the user will be rounded up to the nearest multiple of BUFFER_AREA_CHUNK_SIZE.

2.2.3 BUFFER_CHECKPOINT_LIST_CNT

2.2.3.1 Data Type

Unsigned Integer

2.2.3.2 Default Value

4

2.2.3.3 Attributes

Read-Only, Single Value

2.2.3.4 Range

[1, 64]

2.2.3.5 Description

This indicates the number of checkpoint lists. The greater the number of checkpoint lists, the less lock contention there is among transactions.

2.2.4 BUFFER_FLUSHER_CNT

2.2.4.1 Data Type

Unsigned Integer

2.2.4.2 Default Value

2

2.2.4.3 Attributes

Read-Only, Single Value

2.2.4.4 Range

[1, 16]

2.2.4.5 Description

This indicates the number of buffer flushers. This parameter can't be changed while the server is running.

2.2.5 BUFFER_FLUSH_LIST_CNT

2.2.5.1 Data Type

Unsigned Integer

2.2.5.2 Default Value

1

2.2.5.3 Attributes

Read-Only, Single Value

2.2.5.4 Range

[1, 64]

2.2.5.5 Description

This indicates the number of flush lists. The more flush lists there are, the less lock contention there is among transactions.

2.2.6 BUFFER_HASH_BUCKET_DENSITY

2.2.6.1 Data Type

Unsigned Integer

2.2.6.2 Default Value

1

2.2.6.3 Attributes

Read-Only, Single Value

2.2.6.4 Range

[1, 100]

2.2.6.5 Description

This indicates the percentage of BCBs (Buffer Control Blocks) that can be contained in one bucket. For example, when the number of BCBs is 100, if this value is set to 1, lock contention is minimized because the number of buckets is the same as the number of buffer frames in the buffer pool. If this value is set to 2, the number of buckets is half the number of frames, whereas if this value is set to 100, there is only one bucket. As this value is increased, less memory is used; however, operational costs increase because a single bucket will manage more buffer frames.

2.2.7 BUFFER_HASH_CHAIN_LATCH_DENSITY

2.2.7.1 Data Type

Unsigned Integer

2.2.7.2 Default Value

1

2.2.7.3 Attributes

Read-Only, Single Value

2.2.7.4 Range

[1, 100]

2.2.7.5 Description

This sets the percentage of buckets that correspond to each latch in a hash table. For example, when the number of buckets is 1000, if this value is 1, one latch corresponds to ten buckets. If this value is 2, twenty buckets share a single latch. If this value is 100, only one latch exists for the entire hash table.

This property is used to control concurrency when inserting a BCB (Buffer Control Block) into a hash table or deleting it therefrom. The more latches there are, the less hash chain latch contention will occur.

2.2.8 BUFFER_LRU_LIST_CNT

2.2.8.1 Data Type

Unsigned Integer

2.2.8.2 Default Value

7

2.2.8.3 Attributes

Read-Only, Single Value

2.2.8.4 Range

[1, 64]

2.2.8.5 Description

This indicates the number of LRU lists. LRU list lock contention among transactions decreases as this value is increased.

2.2.9 BUFFER_PREPARE_LIST_CNT

2.2.9.1 Data Type

Unsigned Integer

2.2.9.2 Default Value

7

2.2.9.3 Attributes

Read-Only, Single Value

2.2.9.4 Range

[1, 64]

2.2.9.5 Description

This indicates the number of prepare lists. The greater this value is, the less prepare list lock contention there is among transactions.

2.2.10 BULKIO_PAGE_COUNT_FOR_DIRECT_PATH_INSERT

2.2.10.1 Data Type

Unsigned Integer

2.2.10.2 Default Value

128

2.2.10.3 Attributes

Read-Write, Single Value

2.2.10.4 Range

[128, 12800]

2.2.10.5 Description

This property indicates how many pages can be simultaneously written to disk when entering data using direct-path INSERT. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.2.11 COMPRESSION_RESOURCE_GC_SECOND

2.2.11.1 Data Type

Unsigned Integer

2.2.11.2 Default Value

3600

2.2.11.3 Attributes

Read-Only, Single Value

2.2.11.4 Range

[1, (2^{64} – 1)/1000000]

2.2.11.5 Description

This property specifies the amount of time, in seconds, that unused resources are retained in the log compression resource pool before they are discarded.

2.2.12 DB_NAME

2.2.12.1 Data Type

String

2.2.12.2 Default Value

mydb

2.2.12.3 Attributes

Read-Only, Single Value

2.2.12.4 Range

None

2.2.12.5 Description

This indicates the database name. When a database is created, you must set the database name to the same value as the value in this property.

2.2.13 DDL_SUPPLEMENTAL_LOG_ENABLE

2.2.13.1 Data Type

Unsigned Integer

2.2.13.2 Default Value

0

2.2.13.3 Attributes

Read-Write, Single Value

2.2.13.4 Range

[0, 1]

2.2.13.5 Description

This property determines whether to add a log file when a DDL statement is executed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

0: Disabled (Do not add a log file)

1: Enabled (add a log file)

2.2.14 DEFAULT_DISK_DB_DIR

2.2.14.1 Data Type

String

2.2.14.2 Default Value

$ALTIBASE_HOME/dbs

2.2.14.3 Attributes

Read-Only, single

2.2.14.4 Range

None

2.2.14.5 Description

This property specifies the directory in which to save the disk database files. This property must be set, even if the DRDBMS feature is not used. The default value is $ALTIBASE_HOME/dbs.

2.2.15 DEFAULT_MEM_DB_FILE_SIZE

2.2.15.1 Data Type

Unsigned Long

2.2.15.2 Default Value

1073741824 bytes (1GB)

2.2.15.3 Attributes

Read-Only, Single Value

2.2.15.4 Range

[4194304 (4MB), 2^{64} - 1]

2.2.15.5 Description

This property indicates the default checkpoint image file size, in bytes, for memory tablespaces.

2.2.16 DEFAULT_SEGMENT_MANAGEMENT_TYPE

2.2.16.1 Data Type

Unsigned Integer

2.2.16.2 Default Value

1

2.2.16.3 Attributes

Read-Only, Single Value

2.2.16.4 Range

None

2.2.16.5 Description

This indicates how segments are managed when creating disk tablespaces.

0: MANUAL –segments are created on the basis of a so-called "free list" method of managing available space in the user tablespace

1: AUTO –segments are created on the basis of a bitmap index to manage available space in the user tablespace

2.2.17 DEFAULT_SEGMENT_STORAGE_INITEXTENTS

2.2.17.1 Data Type

Unsigned Integer

2.2.17.2 Default Value

1

2.2.17.3 Attributes

Read-Only, Single Value

2.2.17.4 Range

[1, 2^{32} - 1]

2.2.17.5 Description

This sets the default number of extents that are initially allocated to a segment.

2.2.18 DEFAULT_SEGMENT_STORAGE_MAXEXTENTS

2.2.18.1 Data Type

Unsigned Integer

2.2.18.2 Default Value

2^{32} - 1

2.2.18.3 Attributes

Read-Only, Single Value

2.2.18.4 Range

[1, 2^{32} - 1]

2.2.18.5 Description

This sets the maximum number of extents that can be allocated to a segment.

2.2.19 DEFAULT_SEGMENT_STORAGE_MINEXTENTS

2.2.19.1 Data Type

Unsigned Integer

2.2.19.2 Default Value

1

2.2.19.3 Attributes

Read-Only, Single Value

2.2.19.4 Range

[1, 2^{32} - 1]

2.2.19.5 Description

This sets the minimum number of extents that can be allocated to a segment.

2.2.20 DEFAULT_SEGMENT_STORAGE_NEXTEXTENTS

2.2.20.1 Data Type

Unsigned Integer

2.2.20.2 Default Value

1

2.2.20.3 Attributes

Read-Only, Single Value

2.2.20.4 Range

[1, 2^{32} - 1]

2.2.20.5 Description

This sets the number of extents that can be added to an existing segment.

2.2.21 DIRECT_PATH_BUFFER_PAGE_COUNT

2.2.21.1 Data Type

Unsigned Integer

2.2.21.2 Default Value

1024

2.2.21.3 Attributes

Read-Write, Single Value

2.2.21.4 Range

[1024, 2^{32} - 1]

2.2.21.5 Description

This sets the number of pages in the direct-path INSERT buffer. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.2.22 DISK_INDEX_UNBALANCED_SPLIT_RATE

2.2.22.1 Data Type

Unsigned Integer

2.2.22.2 Default Value

90

2.2.22.3 Attributes

Read-Write, Single Value

2.2.22.4 Range

[50, 99]

2.2.22.5 Description

In a disk B+ tree index, when the last child node of a leaf node in the lowest rank is divided, this property specifies the ratio by which to divide keys between the node to be divided and the created node. When this value is set to 90, which is the default value, the key ratio between the 2 nodes is 90:10. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.2.23 DISK_LOB_COLUMN_IN_ROW_SIZE

2.2.23.1 Data Type

Unsigned Long

2.2.23.2 Default Value

4000

2.2.23.3 Attributes

Read-Only, Single Value

2.2.23.4 Range

[0,4000]

2.2.23.5 Description

Ths property sets the default column size, in bytes, when LOB type data are stored directly in disk tables.

When data are entered into a LOB data type column, if the data length is smaller or the same as the value specified here, they are saved in table segment, whereas if the data are larger than this value, they are saved in LOB segment. This property pertains only to disk tables, and has no effect on how memory tables are managed.

For detailed information on LOB type data, please refer to Chapter1: Data Types.

2.2.24 DOUBLE_WRITE_DIRECTORY

2.2.24.1 Data Type

String

2.2.24.2 Default Value

None

2.2.24.3 Attributes

Read-Only, Multiple Values

2.2.24.4 Range

None

2.2.24.5 Description

This specifies the directory in which double-write files are saved. Multiple values can be saved for this property, according to the value specified in DOUBLE_WRITE_DIRECTORY_COUNT.

2.2.25 DOUBLE_WRITE_DIRECTORY_COUNT

2.2.25.1 Data Type

Unsigned Integer

2.2.25.2 Default Value

2

2.2.25.3 Attributes

Read-Only, Single Value

2.2.25.4 Range

[1, 16]

2.2.25.5 Description

This specifies the number of directories in which double-write files are saved. Double write files can independently be saved on different disks. Because respective double-write files are used for each flusher, better flush performance can be realized when directories on different disks are specified.

2.2.26 DRDB_FD_MAX_COUNT_PER_DATAFILE

2.2.26.1 Data Type

Unsigned Integer

2.2.26.2 Default Value

8

2.2.26.3 Attributes

Read-Write, Single Value

2.2.26.4 Range

[1, 1024]

2.2.26.5 Description

This property specifies the maximum number of FD (File Descriptors) that can be opened for I/O operations on a single disk data file. If the maximum number of FDs specified in this property has been opened, requests to open additional FDs will wait until previous I/O operations are completed.

2.2.27 EXPAND_CHUNK_PAGE_COUNT

2.2.27.1 Data Type

Unsigned Integer

2.2.27.2 Default Value

128

2.2.27.3 Attributes

Read-Only, Single Value

2.2.27.4 Range

[64, 2^{64}- 1]

2.2.27.5 Description

This property specifies the number of pages by which to increase the size of the memory database.

2.2.28 FULL_SCAN_USE_BUFFER_POOL

2.2.28.1 Data Type

Unsigned Integer

2.2.28.2 Default Value

0

2.2.28.3 Attributes

Read-Write, Single Value

2.2.28.4 Range

[0, 1]

2.2.28.5 Description

This property indicates whether disk tables are loaded into a buffer when a full scan is performed on them. It is recommended to specify this property as 0 because there is a low probability that a full scan is performed on already read ones.

2.2.29 LOGANCHOR_DIR

2.2.29.1 Data Type

String

2.2.29.2 Default Value

$ALTIBASE_HOME/logs

2.2.29.3 Attributes

Read-Only, Multiple Values

2.2.29.4 Range

None

2.2.29.5 Description

This property specifies the pathnames for the log anchor files. There must be three log anchor file pathways. They are all set to the same default path.

2.2.30 LOG_DIR

2.2.30.1 Data Type

String

2.2.30.2 Default Value

$ALTIBASE_HOME/logs

2.2.30.3 Attributes

Read-Only, Multiple Values

2.2.30.4 Range

None

2.2.30.5 Description

This property specifies the pathname for log files. When using the log file group functionality, the number of values specified here must be equal to the value specified in LOG_FILE_GROUP_COUNT.

2.2.31 LOG_FILE_SIZE

2.2.31.1 Data Type

Unsigned long

2.2.31.2 Default Value

10 * 1024 * 1024

2.2.31.3 Attributes

Read-Only, Single Value

2.2.31.4 Range

[1024 * 1024, 2^{64}-1]

2.2.31.5 Description

This property specifies the size, in bytes, of a log file. When an active log file fills up, writing continues in a new log file. This property can be set only when creating a database; it can't be changed afterwards. If the user arbitrarily changes this property after a database has been created, abnormal shutdown or other problems can occur.

Restrictions

- In order to perform offline replication, this property must be set the same on the local (active) server and the remote (standby) server.
- On Microsoft Windows (x64), if the DIRECT_IO_ENABLED property is set to 1, LOG_FILE_SIZE must be set lower than 32Mbytes because of operating system-specific buffer size restrictions. In order to set LOG_FILE_SIZE to a value greater than 32Mbytes, DIRECT_IO_ENABLED must be set to 0.

  Please, refer to http://msdn.microsoft.com/en-us/library/aa365747%28VS.85%29.aspx.

2.2.32 MAX_CLIENT

2.2.32.1 Data Type

Unsigned integer

2.2.32.2 Default Value

1000

2.2.32.3 Attributes

Read-Only, Single Value

2.2.32.4 Range

[0, $2^{32} – 1$]

2.2.32.5 Description

This property specifies the maximum number of clients that can connect to an ALTIBASE HDB server.

2.2.33 MEM_DB_DIR

2.2.33.1 Data Type

String

2.2.33.2 Default Value

$ALTIBASE_HOME/dbs

2.2.33.3 Attributes

Read-Only, Multiple Values

2.2.33.4 Range

None

2.2.33.5 Description

This property specifies the pathname for the memory database files.

It is possible to specify a minimum of 1 to a maximum of 8 paths. If multiple paths are specified, the database files are distributed among the paths. All of the paths specified using this parameter must be actual existing paths. The default number of paths is two, and they are both set to $ALTIBASE_HOME/dbs.

This parameter cannot be modified after the database has been created.

2.2.34 MEM_MAX_DB_SIZE

2.2.34.1 Data Type

Unsigned Long

2.2.34.2 Default Value

2^{32}

2.2.34.3 Attributes

Read-Only, Single Value

2.2.34.4 Range

[1024 * 1024, 2^{32}] (32 bits), [1024 * 1024, 2^{64}-1] (64 bits)

2.2.34.5 Description

This property specifies the maximum size, in bytes, to which a memory database can dynamically increase while the server is running. The default value is 4 GB for both 32-bit and 64-bit mode.

If a database expands to a size exceeding MEM_MAX_DB_SIZE, the offending transaction is treated as an error, and all subsequent SQL statements other than SELECT statements are also treated as errors.

2.2.35 MEMORY_INDEX_BUILD_RUN_SIZE

2.2.35.1 Data Type

Unsigned Long

2.2.35.2 Default Value

32768 (bytes)

2.2.35.3 Attributes

Read-Write, Single Value

2.2.35.4 Range

[1024, 2^{64} - 1]

2.2.35.5 Description

This sets the size, in bytes, of the in-memory sorting area for building memory indexes. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.2.36 MEMORY_INDEX_BUILD_VALUE_LENGTH_THRESHOLD

2.2.36.1 Data Type

Unsigned Long

2.2.36.2 Default Value

64

2.2.36.3 Attributes

Read-Write, Single Value

2.2.36.4 Range

[0, 2^{64} - 1]

2.2.36.5 Description

This property sets the maximum length, in bytes, of the key value used for intermediate sorting when building memory indexes.

If the length of the key value is less than this value, the key value is used for intermediate sorting. If this property is set to 0, the index build thread uses a pointer to the record rather than this key value.

This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.2.37 MEMORY_LOB_COLUMN_IN_ROW_SIZE

2.2.37.1 Data Type

Unsigned Long

2.2.37.2 Default Value

64

2.2.37.3 Attributes

Read-Only, Single Value

2.2.37.4 Range

[0,4000]

2.2.37.5 Description

This property sets the default column size, in bytes, when LOB type data are stored directly in memory tables.

When data are entered into a LOB data type column, if the data length is smaller or the same as the value specified here, they are saved in a fixed amount of area, whereas if the data are larger than this value, they are saved in a variable area. This property pertains only to memory tables, and has no effect on how disk tables are managed.

For detailed information on LOB type data, please refer to Chapter1: Data Types.

2.2.38 MEMORY_VARIABLE_COLUMN_IN_ROW_SIZE

2.2.38.1 Data Type

Unsigned Long

2.2.38.2 Default Value

32

2.2.38.3 Attributes

Read-Write, Single Value

2.2.38.4 Range

[0,4000]

2.2.38.5 Description

This property sets the default column size, in bytes, when the variable type data are stored directly in memory tables.

When data are entered into the variable type column, if the data length is smaller or the same as the value specified here, they are saved in a fixed amount of area, whereas if the data are larger than this value, they are saved in a variable area. This property pertains only to memory tables, and has no effect on how disk tables are managed.

For detailed information on IN ROW clause, please refer to Chapter1: Data Types.

2.2.39 MEM_SIZE_CLASS_COUNT

2.2.39.1 Data Type

Unsigned Integer

2.2.39.2 Default Value

4

2.2.39.3 Attributes

Read-Only, Single Value

2.2.39.4 Range

[1, 4]

2.2.39.5 Description

This property determines the number of categories into which memory pages are classified based on the amount of free space in them.

2.2.40 MIN_COMPRESSION_RESOURCE_COUNT

2.2.40.1 Data Type

unsigned integer

2.2.40.2 Default Value

16

2.2.40.3 Attributes

Read-Only, Single Value

2.2.40.4 Range

[1, 10240]

2.2.40.5 Description

This property indicates the minimum number of buffer chunks used by the log manager for log compression. (One compression buffer chunk is about 16kB.)

2.2.41 MIN_LOG_RECORD_SIZE_FOR_COMPRESS

2.2.41.1 Data Type

Unsigned Integer

2.2.41.2 Default Value

512

2.2.41.3 Attributes

Read-Write, Single Value

2.2.41.4 Range

[0, 2^{32} - 1]

2.2.41.5 Description

This property specifies the log size, in bytes, that is used to determine whether to compress logs.

When this property is set to 0, logs are never compressed. If the size of a log exceeds the size specified here, logs will be compressed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.2.42 MIN_PAGES_ON_DB_FREE_LIST

2.2.42.1 Data Type

Unsigned integer

2.2.42.2 Default Value

16

2.2.42.3 Attributes

Read-Only, Single Value

2.2.42.4 Range

[1, 2^{32} – 1]

2.2.42.5 Description

This property specifies the minimum number of free pages that must be available on each list of free pages. These pages are allocated to table free lists as required.

2.2.43 MIN_PAGES_ON_TABLE_FREE_LIST

2.2.43.1 Data Type

Unsigned integer

2.2.43.2 Default Value

1

2.2.43.3 Attributes

Read-Write, Single Value

2.2.43.4 Range

[1, $2^{32} – 1$]

2.2.43.5 Description

This property specifies the minimum number of free pages that each table must maintain on its own list of free pages.

2.2.44 PCTFREE

2.2.44.1 Data Type

Unsigned Integer

2.2.44.2 Default Value

10

2.2.44.3 Attributes

Read-Only, Single Value

2.2.44.4 Range

[0, 99]

2.2.44.5 Description

This property indicates the minimum percentage of space to keep free in each page for the insertion of data. The value specified by PCTFREE indicates the percentage of space that is kept free in order to allow existing records to be updated.

If the total size of the tablespace is 100MB and the value of PCTFREE is 10, up to 90MB of data, or data equivalent in size to 90% of the total space, can be inserted.

If the value of PCTFREE is not set using a CREATE TABLE statement when a disk table is created, the default value is used.

2.2.45 PCTUSED

2.2.45.1 Data Type

Unsigned Integer

2.2.45.2 Default Value

40

2.2.45.3 Attributes

Read-Only, Single Value

2.2.45.4 Range

[0, 99]

2.2.45.5 Description

The PCTUSED property is the minimum percentage of used space for reinsertion when ALTIBASE HDB can only update record.This property indicates the amount of space that is used to convert the state of a tablespace page from one on which only updates can be performed to one on which insert operations can also be performed.

When enough data have been entered that the amount of used page space reaches the value specified in PCTFREE, only update operations can be performed. In this state, if the amount of used space falls below the value of PCTUSED due to update and delete operations, new records can be inserted.

If the value of PCTUSED is not explicitly set using a CREATE TABLE statement when a disk table is created, the default value is used.

2.2.46 QP_MEMORY_CHUNK_SIZE

2.2.46.1 Data Type

Unsigned Long

2.2.46.2 Default Value

65536

2.2.46.3 Attributes

Read-Only, Single Value

2.2.46.4 Range

[1024, 2^{64} – 1]

2.2.46.5 Description

This property specifies the number of additional bytes allocated by the system each time the Query Processor requires additional memory.

2.2.47 SECURITY_ECC_POLICY_NAME

2.2.47.1 Data Type

String

2.2.47.2 Default Value

None

2.2.47.3 Attributes

Read-Write, Single Value

2.2.47.4 Range

None

2.2.47.5 Description

This property indicates the name of ECC (Encrypted Comparison Code) algorithm used when you perform a security module for the encrypted columns.

2.2.48 SECURITY_MODULE_LIBRARY

2.2.48.1 Data Type

String

2.2.48.2 Default Value

None

2.2.48.3 Attributes

Read-Write, Single Value

2.2.48.4 Range

None

2.2.48.5 Description

This property indicates library file name of security module, and is used when you perform a security module.

2.2.49 SECURITY_MODULE_NAME

2.2.49.1 Data Type

String

2.2.49.2 Default Value

None

2.2.49.3 Attributes

Read-Write, Single Value

2.2.49.4 Range

None

2.2.49.5 Description

This property indicates the name of security module, and is used when you perform a security module.

2.2.50 SHM_DB_KEY

2.2.50.1 Data Type

Unsigned integer

2.2.50.2 Default Value

0

2.2.50.3 Attributes

Read-Write, Single Value

2.2.50.4 Range

[0, $2^{32} - 1$]

2.2.50.5 Description

If the database is to be used in virtual memory space, this parameter is set to 0, whereas If shared memory is used, this parameter must be set to the shared memory key value. The shared memory key value can be any arbitrary value not used by the system. Because the process of reading pages from disk is not necessary when the database is located in shared memory rather than on disk, ALTIBASE HDB server starting time can be reduced.

2.2.51 STARTUP_SHM_CHUNK_SIZE

2.2.51.1 Data Type

Unsigned long

2.2.51.2 Default Value

1 G

2.2.51.3 Attributes

Read-Only, Single Value

2.2.51.4 Range

[1024, $2^{64} - 1$]

2.2.51.5 Description

In the state in which a value other than 0 has been set for SHM_DB_KEY, i.e. when the database is to be stored in shared memory, this property sets the maximum size, in bytes, of shared memory chunks that are created when ALTIBASE HDB is started.

2.2.52 ST_OBJECT_BUFFER_SIZE

2.2.52.1 Data Type

Unsigned long

2.2.52.2 Default Value

32000

2.2.52.3 Attributes

Read-Write, Single Value

2.2.52.4 Range

[32000, 104857600]

2.2.52.5 Description

This sets the maximum size, in bytes, of a single geometry object.

2.2.53 SYS_DATA_FILE_INIT_SIZE

2.2.53.1 Data Type

Unsigned long

2.2.53.2 Default Value

100M (100 * 1024 * 1024)

2.2.53.3 Attributes

Read-Only, Single Value

2.2.53.4 Range

[8 * 8kB, 32GB]

2.2.53.5 Description

This specifies the initial size, in bytes, of the data file (system001.dbf) when SYS_TBS_DISK_DATA (system disk tablespace) is created. Moreover, if the initial size is not specified when a data file (that is, a user-specified file other than system001.dbf) is added to SYS_TBS_DISK_DATA, the initial size of that data file also defaults to the value specified here.

2.2.54 SYS_DATA_FILE_MAX_SIZE

2.2.54.1 Data Type

Unsigned long

2.2.54.2 Default Value

2 * 1024 * 1024 * 1024

2.2.54.3 Attributes

Read-Only, Single Value

2.2.54.4 Range

[8 * 8kB, 32GB]

2.2.54.5 Description

This property specifies the maximum size, in bytes, of the allocated data file when SYS_TBS_DISK_DATA (system disk tablespace) is created. It must be equal to or greater than the value of SYS_DATA_FILE_INIT_SIZE. The minimum possible value is 64kB.

Moreover, if no maximum value is set when data files are added to SYS_TBS_DISK_DATA (system disk tablespace), the value specified here will be taken for SYS_DATA_FILE_MAX_SIZE.

2.2.55 SYS_DATA_FILE_NEXT_SIZE

2.2.55.1 Data Type

Unsigned long

2.2.55.2 Default Value

1 * 1024 * 1024 (bytes)

2.2.55.3 Attributes

Read-Only, Single Value

2.2.55.4 Range

[8 * 8kB, 32GB]

2.2.55.5 Description

When the autoextend property of system disk tablespace (SYS_TBS_DISK_DATA) is set to “autoextend on”, data files are automatically incremented in size by the number of bytes specified here in order to accommodate increased amounts of data.

If the size of a data file reaches the value specified in SYS_DATA_FILE_MAX_SIZE , and additionally the amount of valid space in other data files is less than that specified in SYS_DATA_FILE_NEXT_SIZE, an insufficient tablespace error will be raised.

2.2.56 SYS_DATA_TBS_EXTENT_SIZE

2.2.56.1 Data Type

Unsigned long

2.2.56.2 Default Value

512 * 1024

2.2.56.3 Attributes

Read-Only, Single Value

2.2.56.4 Range

[40kB, 32GB]

2.2.56.5 Description

This specifies the size, in bytes, of an extent[1] when SYS_TBS_DISK_DATA (system disk tablespace) is created[2]. In order for an extent to contain at least 5 pages, the minimum value of this property is 40kB (5*8kB).

2.2.57 SYS_TEMP_FILE_INIT_SIZE

2.2.57.1 Data Type

Unsigned long

2.2.57.2 Default Value

100M (100 * 1024 * 1024)

2.2.57.3 Attributes

Read-Only, Single Value

2.2.57.4 Range

[8 * 8kB, 32GB]

1. The initial extent size cannot be changed after the database has been created. The default value is 32 pages.
2. System disk data tablespace: this is the disk tablespace that is created by default when a database is created. The disk table and disk index are the only database objects that are saved.

2.2.57.5 Description

This specifies the initial size, in bytes, of the temporary data file (temp001.dbf) when SYS_TBS_DISK_TEMP is created. Moreover, if the initial size is not specified when a temporary data file is added to SYS_TBS_DISK_TEMP, the value specified here is used.

2.2.58 SYS_TEMP_FILE_MAX_SIZE

2.2.58.1 Data Type

Unsigned long

2.2.58.2 Default Value

2 * 1024 * 1024 * 1024

2.2.58.3 Attributes

Read-Only, Single Value

2.2.58.4 Range

[8 * 8kB, 32GB]

2.2.58.5 Description

This specifies the maximum size, in bytes, of the data file (temp001.dbf) that is allocated when SYS_TBS_DISK_TEMP is created.

The value of this property must be at least as great as that of SYS_TEMP_FILE_INIT_SIZE. The minimum possible value is 64kB. Moreover, if the maximum size is not specified when a temporary data file is added to SYS_TBS_DISK_TEMP, the size specified here is the default maximum size.

2.2.59 SYS_TEMP_FILE_NEXT_SIZE

2.2.59.1 Data Type

Unsigned long

2.2.59.2 Default Value

1 * 1024 * 1024

2.2.59.3 Attributes

Read-Only, Single Value

2.2.59.4 Range

[8 * 8kB, 32GB]

2.2.59.5 Description

If there is not enough space in a data file in the SYS_TBS_DISK_TEMP tablespace, the size of the file is increased by the amount specified here.

2.2.60 SYS_TEMP_TBS_EXTENT_SIZE

2.2.60.1 Data Type

Unsigned long

2.2.60.2 Default Value

256 * 1024

2.2.60.3 Attributes

Read-Only, Single Value

2.2.60.4 Range

[40kB, 32GB]

2.2.60.5 Description

This specifies the size, in bytes, of an extent when the SYS_TBS_DISK_TEMP (system disk temporary tablespace)[1] is created. It must be large enough to contain at least five pages (40kB = 5 * 8kB).

2.2.61 SYS_UNDO_FILE_INIT_SIZE

2.2.61.1 Data Type

Unsigned long

2.2.61.2 Default Value

100 * 1024 * 1024

1. System disk temporary tablespace: This is automatically created by default when a database is created, and is a tablespace for temporary storage related to various kinds of database operations. It is set as the default temporary tablespace for storing objects on disk for all users. The only database objects that are stored here are disk tables and disk indexes.

2.2.61.3 Attributes

Read-Only, Single Value

2.2.61.4 Range

[32 * 8kB, 32GB]

2.2.61.5 Description

This specifies the default size, in bytes, of the data file (undo001.dbf) when SYS_TBS_DISK_UNDO tablespace is created. Additionally, when a data file is added to SYS_TBS_DISK_UNDO without specifying its initial size, the size specified here is used.

2.2.62 SYS_UNDO_FILE_MAX_SIZE

2.2.62.1 Data Type

Unsigned long

2.2.62.2 Default Value

2 * 1024 * 1024 * 1024

2.2.62.3 Attributes

Read-Only, Single Value

2.2.62.4 Range

[32 * 8kB, 32GB]

2.2.62.5 Description

This specifies the maximum size, in bytes, of the data file (undo001.dbf) that is allocated when SYS_TBS_DISK_UNDO is created.

The value of this property must be at least as great as that of SYS_UNDO_FILE_INIT_SIZE. The minimum possible value is 256kB. Moreover, if the maximum size is not specified when a temporary data file is added to SYS_TBS_DISK_UNDO, the value specified here is used as the default maximum size.

2.2.63 SYS_UNDO_FILE_NEXT_SIZE

2.2.63.1 Data Type

Unsigned long

2.2.63.2 Default Value

1 * 1024 * 1024

2.2.63.3 Attributes

Read-Only, Single Value

2.2.63.4 Range

[8 * 8kB, 32GB]

2.2.63.5 Description

When there is not enough space in the SYS_TBS_DISK_UNDO tablespace data file, the size of the data file is incremented by the number of bytes specified here.

2.2.64 SYS_UNDO_TBS_EXTENT_SIZE

2.2.64.1 Data Type

Unsigned long

2.2.64.2 Default Value

256 * 1024

2.2.64.3 Attributes

Read-Only, Single Value

2.2.64.4 Range

[40kB, 32GB]

2.2.64.5 Description

This specifies the size, in bytes, of an extent when SYS_TBS_DISK_UNDO (system disk undo tablespace)[1] is created.

1. System disk undo tablespace: this is automatically created by default when a database is created, and is used only for saving undo information. Only one system disk undo tablespace exists in a database. The user cannot create or delete tables, indexes, or anything else in system disk undo tablespace.

2.2.65 TABLE_BACKUP_FILE_BUFFER_SIZE

2.2.65.1 Data Type

Unsigned Integer

2.2.65.2 Default Value

1024

2.2.65.3 Attributes

Read-Only, Single Value

2.2.65.4 Range

[0, 1048576]

2.2.65.5 Description

This property specifies the table backup buffer size, in bytes, for use when using the ALTER TABLE command to add or delete columns to or from memory tables.

2.2.66 TABLE_COMPACT_AT_SHUTDOWN

2.2.66.1 Data Type

Unsigned Integer

2.2.66.2 Default Value

1

2.2.66.3 Attributes

Read-Write, Single Value

2.2.66.4 Range

None

2.2.66.5 Description

This property indicates whether to compact tables when you shut down database. It is recommended to specify this property as 1 to reduce memory consumption of tables when you restart database up.

2.2.67 TEMP_PAGE_CHUNK_COUNT

2.2.67.1 Data Type

Unsigned integer

2.2.67.2 Default Value

128

2.2.67.3 Attributes

Read-Only, Single Value

2.2.67.4 Range

[1, 2^{32} – 1]

2.2.67.5 Description

This property indicates the number of temporary data pages that can be allocated at one time.

2.2.68 TRCLOG_DETAIL_SCHEMA

2.2.68.1 Data Type

Unsigned integer

2.2.68.2 Default Value

0

2.2.68.3 Attributes

Read-Write, Single Value

2.2.68.4 Range

[0, 1]

2.2.68.5 Description

This property indicates whether to output the names of the owners of tables, indexes, and Database Link-related objects when outputting the execution plan for a SQL statement.

2.2.69 USER_DATA_FILE_INIT_SIZE

2.2.69.1 Data Type

Unsigned long

2.2.69.2 Default Value

100 * 1024 * 1024

2.2.69.3 Attributes

Read-Only, Single Value

2.2.69.4 Range

[8 * 8kB, 32GB]

2.2.69.5 Description

This property sets the initial size, in bytes, of a user-defined data file that is created or added to user disk data tablespace. The default value specified here is used if no initial size is specified.

2.2.70 USER_DATA_FILE_MAX_SIZE

2.2.70.1 Data Type

Unsigned long

2.2.70.2 Default Value

2 * 1024 * 1024 * 1024

2.2.70.3 Attributes

Read-Only, Single Value

2.2.70.4 Range

[8 * 8kB, 32GB]

2.2.70.5 Description

This sets the maximum size, in bytes, of a user-defined data file that is created or added to user disk data tablespace.

The value of this property should be at least as big as that specified in USER_DATA_FILE_INIT_SIZE. The minimum possible value is 64kB. If no maximum size is specified when a data file is created or added, the default value specified here is used.

2.2.71 USER_DATA_FILE_NEXT_SIZE

2.2.71.1 Data Type

Unsigned long

2.2.71.2 Default Value

1 * 1024 * 1024

2.2.71.3 Attributes

Read-Only, Single Value

2.2.71.4 Range

[8 * 8kB, 32GB]

2.2.71.5 Description

When there is not enough data file space in the user-defined data file user disk data tablespace, the size of the data file is incremented by the number of bytes specified here.

2.2.72 USER_DATA_TBS_EXTENT_SIZE

2.2.72.1 Data Type

Unsigned long

2.2.72.2 Default Value

512 * 1024

2.2.72.3 Attributes

Read-Only, Single Value

2.2.72.4 Range

[2 * 8kB, 2^{64} – 1]

2.2.72.5 Description

This specifies the size, in bytes, of an extent when a user disk data tablespace is created.

2.2.73 USER_TEMP_FILE_INIT_SIZE

2.2.73.1 Data Type

Unsigned long

2.2.73.2 Default Value

100 * 1024 * 1024

2.2.73.3 Attributes

Read-Only, Single Value

2.2.73.4 Range

[8 * 8kB, 32GB]

2.2.73.5 Description

This specifies the initial size, in bytes, of a data file when a user-defined temporary data file is created or added to user temporary tablespace. If no initial size is specified, the default value specifed here is used.

2.2.74 USER_TEMP_FILE_MAX_SIZE

2.2.74.1 Data Type

Unsigned long

2.2.74.2 Default Value

2 * 1024 * 1024 * 1024

2.2.74.3 Attributes

Read-Only, Single Value

2.2.74.4 Range

[8 * 8kB, 32GB]

2.2.74.5 Description

This property limits the maximum size, in bytes, of user-defined temporary data files that are created in or added to user temporary tablespace.

This parameter must be at least as great as USER_DATA_FILE_INIT_SIZE. The minimum possible value is 64kB. If no maximum size is specified when temporary data files are created or added, the default value specified here is used.

2.2.75 USER_TEMP_FILE_NEXT_SIZE

2.2.75.1 Data Type

Unsigned long

2.2.75.2 Default Value

1 * 1024 * 1024

2.2.75.3 Attributes

Read-Only, Single Value

2.2.75.4 Range

[8 * 8kB, 32GB]

2.2.75.5 Description

If there is insufficient space in a user-defined temporary data file in user temporary tablespace, the size of the data file is increased by the number of bytes specified here.

2.2.76 USER_TEMP_TBS_EXTENT_SIZE

2.2.76.1 Data Type

Unsigned long

2.2.76.2 Default Value

256 * 1024

2.2.76.3 Attributes

Read-Only, Single Value

2.2.76.4 Range

[2 * 8kB, 2^{64} – 1]

2.2.76.5 Description

This specifies the size, in bytes, of an extent when user temporary tablespace is created. It must be at least 2 pages (16kB = 2 * 8kB).

2.2.77 VOLATILE_MAX_DB_SIZE

2.2.77.1 Data Type

Unsigned long

2.2.77.2 Default Value

2^{32}

2.2.77.3 Attributes

Read-Only, Single Value

2.2.77.4 Range

[2097152, 2^{64} - 1]

2.2.77.5 Description

This property specifies the maximum size, in bytes, of volatile tablespace.

2.3 Performance Properties

2.3.1 AGER_WAIT_MAXIMUM

2.3.1.1 Data Type

Unsigned integer

2.3.1.2 Default Value

100000

2.3.1.3 Attributes

Read-Only, Single Value

2.3.1.4 Range

[0, 2^{32} – 1]

2.3.1.5 Description

This property specifies the maximum waiting time, in microseconds, of the garbage collector (also known as the “Ager”).

This property is intended to prevent deterioration in performance (especially in HP systems) resulting from excessive “sleep” system calls by threads related to the garbage collector while the garbage collector is asleep. This parameter allows the maximum sleep time of the garbage collector to be suitably regulated while the server is running.

2.3.2 AGER_WAIT_MINIMUM

2.3.2.1 Data Type

Unsigned integer

2.3.2.2 Default Value

100

2.3.2.3 Attributes

Read-Only, Single Value

2.3.2.4 Range

[0, $2^{32} - 1$]

2.3.2.5 Description

This property specifies the minimum waiting time, in microseconds, of the garbage collector (also known as the “Ager”).

This property is intended to prevent deterioration in performance (especially in HP systems) resulting from excessive “sleep” system calls by threads related to the garbage collector while the garbage collector is asleep. This parameter allows the minimum sleep time of the garbage collector to be suitably regulated while the server is running.

2.3.3 BUFFER_VICTIM_SEARCH_INTERVAL

2.3.3.1 Data Type

Unsigned Integer

2.3.3.2 Default Value

3000

2.3.3.3 Attributes

Read-Write, Single Value

2.3.3.4 Range

[0, $2^{32} - 1$]

2.3.3.5 Description

After a search for a replacement BCB (“victim”) fails, this property specifies the amount of time, in milliseconds, to wait for the flusher to flush dirty buffer frames before searching again for a victim.

If, as the result of the subsequent search, a replacement BCB still cannot be found, the value of VICTIM_SEARCH_WARP in the V$BUFFPOOL_STAT performance view is increased.

2.3.4 BUFFER_VICTIM_SEARCH_PCT

2.3.4.1 Data Type

Unsigned Integer

2.3.4.2 Default Value

5

2.3.4.3 Attributes

Read-Write, Single Value

2.3.4.4 Range

[0, 100]

2.3.4.5 Description

This property sets how much to explore when searching for replacement buffers in an LRU list. In other words, this property indicates the percentage of an LRU list that is searched, with the least recently accessed records searched first. A value of 100 indicates that the entire list is searched.

2.3.5 CHECKPOINT_BULK_SYNC_PAGE_COUNT

2.3.5.1 Data Type

Unsigned Integer

2.3.5.2 Default Value

3200

2.3.5.3 Attributes

Read-Write, Single Value

2.3.5.4 Range

[0, $2^{32} - 1$]

2.3.5.5 Description

When performing checkpointing between memory and disk tables, this property sets the number of pages that are synchronized at one time. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.6 CHECKPOINT_BULK_WRITE_PAGE_COUNT

2.3.6.1 Data Type

Unsigned Integer

2.3.6.2 Default Value

0

2.3.6.3 Attributes

Read-Write, Single Value

2.3.6.4 Range

[0, 2^{32} - 1]

2.3.6.5 Description

When checkpointing, a given number of dirty pages can be separated and saved to disk. When this happens, this property specifies the number of dirty pages that are saved to disk at one time. If this is set to 0, all of the dirty pages are saved to the disk database at one time. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.7 CHECKPOINT_BULK_WRITE_SLEEP_SEC

2.3.7.1 Data Type

Unsigned Integer

2.3.7.2 Default Value

0

2.3.7.3 Attributes

Read-Write, Single Value

2.3.7.4 Range

[0, 2^{32} - 1]

2.3.7.5 Description

This property specifies the amount of time to wait (in seconds) after saving dirty pages to disk if the value of CHECKPOINT_BULK_WRITE_PAGE_COUNT is not set to 0. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.8 CHECKPOINT_BULK_WRITE_SLEEP_USEC

2.3.8.1 Data Type

Unsigned Integer

2.3.8.2 Default Value

0

2.3.8.3 Attributes

Read-Write, Single Value

2.3.8.4 Range

[0, 2^{32} - 1]

2.3.8.5 Description

This property specifies the amount of time to wait (in microseconds) after saving dirty pages to disk if the value of CHECKPOINT_BULK_WRITE_PAGE_COUNT is not set to 0. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.9 CHECKPOINT_FLUSH_COUNT

2.3.9.1 Data Type

Unsigned Integer

2.3.9.2 Default Value

64

2.3.9.3 Attributes

Read-Write, Single Value

2.3.9.4 Range

[1, 2^{32} – 1]

2.3.9.5 Description

This property specifies the number of buffer pages (frames) that can be flushed in one flusher cycle when checkpoint flushing.

2.3.10 CHECKPOINT_FLUSH_MAX_GAP

2.3.10.1 Data Type

Unsigned Integer

2.3.10.2 Default Value

10

2.3.10.3 Attributes

Read-Write, Single Value

2.3.10.4 Range

[0, 2^{32} – 1]

2.3.10.5 Description

This is one of the conditions for conducting checkpoint processing. Checkpoint flushing is performed when the number of logfiles between the most recent LSN (Log Sequence Number) and the earliest LSN reaches this value.

This property influences the recovery time when the server is restarted. Greater values mean that checkpoint processing is performed less often, and that it takes more time for the server to recover when restarted.

The value of this property can be changed using the ALTER SYSTEM statement while the server is running.

2.3.11 CHECKPOINT_FLUSH_MAX_WAIT_SEC

2.3.11.1 Data Type

Unsigned Integer

2.3.11.2 Default Value

10

2.3.11.3 Attributes

Read-Write, Single Value

2.3.11.4 Range

[0, $2^{32} - 1$]

2.3.11.5 Description

This is one of the conditions for conducting checkpoint processing. Checkpoint flushing is performed when the number of seconds specified by this property has passed since the most recent flush.

2.3.12 CM_BUFFER_MAX_PENDING_LIST

2.3.12.1 Data Type

Unsigned Integer

2.3.12.2 Default Value

128

2.3.12.3 Attributes

Read-Only, Single Value

2.3.12.4 Range

[1, 512]

2.3.12.5 Description

In order to prevent sudden increases in memory usage, this property specifies the maximum number of communication buffer blocks that can be allocated in one session.

2.3.13 DATABASE_IO_TYPE

2.3.13.1 Data Type

Unsigned integer

2.3.13.2 Default Value

0

2.3.13.3 Attributes

Read-Only, Single Value

2.3.13.4 Range

[0, 1]

2.3.13.5 Description

ALTIBASE HDB is a hybrid MMDBMS. Therefore, database-related disk I/O operations occur when data are loaded at the time that the ALTIBASE HDB server is first started, and when checkpointing is conducted while ALTIBASE HDB is running.

ALTIBASE HDB provides two disk I/O methods related to database files:

- Direct I/O
- Buffered I/O

To use direct I/O, set this parameter to 1, or to use buffered I/O, set it to 0.

The advantage of Direct I/O reduces CPU resources during occurance of Disk I/O. On the other hand, since the Buffered I/O uses techniques of read-ahead and asynchronous write, it may not access disk everytime when the disk I/O is required. It means that the Buffered I/O is faster the disk I/O than the Direct I/O in aspect of the application; however the Buffered I/O consumes a higher level of CPU resources when compared to the Direct I/O.

2.3.14 DATAFILE_WRITE_UNIT_SIZE

2.3.14.1 Data Type

Unsigned Long

2.3.14.2 Default Value

1024

2.3.14.3 Attributes

Read-Write, Single Value

2.3.14.4 Range

[1, 1024]

2.3.14.5 Description

This property specifies the default data unit size when a data file is created. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.15 DB_FILE_MULTIPAGE_READ_COUNT

2.3.15.1 Data Type

Unsigned Integer

2.3.15.2 Default Value

8

2.3.15.3 Attributes

Read-Write, Single Value

2.3.15.4 Range

[1, 128]

2.3.15.5 Description

This property determines the number of pages to read at a time when a full scan is performed on a disk table. At this time, if a disk table's extent size, that is, the number of pages in the extent, is a multiple of (and greater than) the value specified here, Multiple Page Read (MPR) is conducted.

However, if the extent size is not a multiple of, or is smaller than, the value specified here, Single Page Read (SPR) is conducted. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.16 DEFAULT_FLUSHER_WAIT_SEC

2.3.16.1 Data Type

Unsigned Integer

2.3.16.2 Default Value

1

2.3.16.3 Attributes

Read-Write, Single Value

2.3.16.4 Range

[1, $2^{32} - 1$]

2.3.16.5 Description

This property sets the minimum number of seconds that the flusher waits. As long as there are no special conditions, flushing is always conducted after waiting this amount of time.

The wait time is repeatedly incremented 1 second at a time if the flusher is removed from the queue or doesn't perform any flushing work.

2.3.17 DIRECT_IO_ENABLED

2.3.17.1 Data Type

Unsigned Integer

2.3.17.2 Default Value

1

2.3.17.3 Attributes

Read-Only, Single Value

2.3.17.4 Range

[0, 1]

2.3.17.5 Description

This property indicates whether database I/O can be performed via direct disk access.

0: disable

1: enable

2.3.18 DISK_INDEX_BUILD_MERGE_PAGE_COUNT

2.3.18.1 Data Type

Unsigned Integer

2.3.18.2 Default Value

128

2.3.18.3 Attributes

Read-Write, Single Value

2.3.18.4 Range

[2, 2^{32}– 1]

2.3.18.5 Description

When a disk index is created, if the keys extracted from data cannot all be sorted in memory at the same time, this property specifies the number of pages to be used for external sorting.

This property can be changed using the ALTER SYSTEM statement during system operation.

2.3.19 EXECUTE_STMT_MEMORY_MAXIMUM

2.3.19.1 Data Type

Unsigned Long

2.3.19.2 Default Value

1G

2.3.19.3 Property

Read-Write, Single Value

2.3.19.4 Range

[1024*1024, 2^{64} – 1]

2.3.19.5 Description

This property limits the number of bytes of memory that can be used to execute a single query statement .

This property can be changed using the ALTER SYSTEM statement during system operation.

2.3.20 FAST_START_IO_TARGET

2.3.20.1 Data Type

Unsigned Long

2.3.20.2 Default Value

10000

2.3.20.3 Attributes

Read-Write, Single Value

2.3.20.4 Range

[1, 2^{64} – 1]

2.3.20.5 Description

This property indicates the number of redo pages that the server reads when performing recovery after being restarted.

When the flusher performs checkpoint flushing while the system is running, if the number of dirty pages remaining in the buffer is greater than the value saved in this property, the oldest dirty pages, equal in number to the difference therebetween, are written to disk.

This value is important in determining the recovery time when the server is restarted. Because the number of pages to be flushed increases as this value is decreased, the recovery time when the server is restarted can be reduced.

The value of this property can be changed using the ALTER SYSTEM statement while the server is running.

2.3.21 FAST_START_LOGFILE_TARGET

2.3.21.1 Data Type

Unsigned Integer

2.3.21.2 Default Value

100

2.3.21.3 Attributes

Read-Write, Single Value

2.3.21.4 Range

[1, 2^{32} – 1]

2.3.21.5 Description

This property indicates the number of log files that the server reads when performing recovery after being restarted.

When the flusher performs checkpoint flushing while the server is running, if the difference between the LogFileNo of the LSN of the current log and the LogFileNo of the LSN of one of the dirty pages in the checkpoint list is greater than the value specified in this property, that page is flushed.

This value is important in determining the recovery time when the server is restarted. Because the number of pages to be flushed increases as this value is decreased, the recovery time when the server is restarted can be reduced.

The value of this property can be changed using the ALTER SYSTEM statement while the server is running.

2.3.22 HIGH_FLUSH_PCT

2.3.22.1 Data Type

Unsigned Integer

2.3.22.2 Default Value

5

2.3.22.3 Attributes

Read-Write, Single Value

2.3.22.4 Range

[0, 100]

2.3.22.5 Description

When the flusher is not in a waiting state, if the flush list is longer than the percentage of the total buffer size specified here, replacement flushing occurs. At this time, all updated buffers in the flush list are flushed sequentially without waiting.

The value of this property can be changed using the ALTER SYSTEM statement while the server is running.

2.3.23 HOT_LIST_PCT

2.3.23.1 Data Type

Unsigned Integer

2.3.23.2 Default Value

0

2.3.23.3 Attributes

Read-Write, Single Value

2.3.23.4 Range

[0, 100]

2.3.23.5 Description

This property specifies the percentage of an LRU list that is a hot area.

The value of this property can be changed using the ALTER SYSTEM statement while the server is running.

2.3.24 HOT_TOUCH_CNT

2.3.24.1 Data Type

Unsigned Integer

2.3.24.2 Default Value

2

2.3.24.3 Attributes

Read-Write, Single Value

2.3.24.4 Range

[1, $2^{32} - 1$]

2.3.24.5 Description

This property defines what constitutes a hot buffer in terms of the number of times the buffer is accessed. If the buffer is accessed more times than the value specified for this property, the buffer is considered hot. Hot buffers are moved to the hot list when replacement buffer searching is performed.

2.3.25 INDEX_BUILD_THREAD_COUNT

2.3.25.1 Data Type

Unsigned integer

2.3.25.2 Default Value

The Number of CPUs

2.3.25.3 Attributes

Read-Write, Single Value

2.3.25.4 Range

[1, 128]

2.3.25.5 Description

This property indicates the number of index-building threads that are created when an index is rebuilt at runtime. If this property is commented out, the default number of parallel threads generated by the system is equal to the number of CPUs.

2.3.26 INDEX_INITRANS

2.3.26.1 Data Type

Unsigned Integer

2.3.26.2 Default Value

8

2.3.26.3 Attributes

Read-Only, Single Value

2.3.26.4 Range

[0, 30]

2.3.26.5 Description

This property indicates the initial number of TTS (Touched Transaction Slots) in an index page.

2.3.27 INDEX_MAXTRANS

2.3.27.1 Data Type

Unsigned Integer

2.3.27.2 Default Value

30

2.3.27.3 Attributes

Read-Only, Single Value

2.3.27.4 Range

[0, 30]

2.3.27.5 Description

This property indicates the maximum number of TTS (Touched Transaction Slots) in an index page.

2.3.28 INSPECTION_LARGE_HEAP_THRESHOLD

2.3.28.1 Data Type

Unsigned integer

2.3.28.2 Default Value

0

2.3.28.3 Attributes

Read-Write, Single Value

2.3.28.4 Range

[0, 2^{32}-1]

2.3.28.5 Description

This property is for showing the user the number of bytes of memory requested by the server for internal use. A call stack log file, which requires a large amount of memory, is output in order to provide the user with information. When this value is set to 0, this information is not output. Call stack information is output to a log file only when the amount of memory that is being used is greater than the value specified here.

2.3.29 LFG_GROUP_COMMIT_INTERVAL_USEC

2.3.29.1 Data Type

Unsigned integer

2.3.29.2 Default Value

1000

2.3.29.3 Attributes

Read-Only, Single Value

2.3.29.4 Range

[0, 2^{32} – 1]

2.3.29.5 Description

This property pertains to group commit.

The last time point at which disk I/O was performed is maintained, for use in writing each log file group (LFG) separately to the log disk. On the basis of this time point, after a number of microseconds equal to the value specified in this property has passed, disk I/O is performed.

In this way, multiple transactions can be collectively committed to disk at the same time, and the requested disk I/O can all be performed at one time.

2.3.30 LFG_GROUP_COMMIT_RETRY_USEC

2.3.30.1 Data Type

Unsigned integer

2.3.30.2 Default Value

100

2.3.30.3 Attributes

Read-Only, Single Value

2.3.30.4 Range

[0, $2^{32} - 1$]

2.3.30.5 Description

This property pertains to group commit.

If the amount of time specified in LFG_GROUP_COMMIT_INTERVAL_USEC has not passed since the last time disk I/O was performed to record logs, a transaction to be committed waits for the number of microseconds specified in this property and then checks again whether sufficient time has passed to perform disk I/O.

2.3.31 LFG_GROUP_COMMIT_UPDATE_TX_COUNT

2.3.31.1 Data Type

Unsigned integer

2.3.31.2 Default Value

80

2.3.31.3 Attributes

Read-Only, Single Value

2.3.31.4 Range

[0, $2^{32} - 1$]

2.3.31.5 Description

This property pertains to group commit.

When the number of database update transactions (internally viewable as UPDATE_TX_COUNT of V$LFG) of an individual log file group (LFG) is greater than the value of this property, group commit is activated.

If this property is set to 0, group commit is disabled.

2.3.32 LOCK_ESCALATION_MEMORY_SIZE

2.3.32.1 Data Type

Unsigned Integer

2.3.32.2 Default Value

100M

2.3.32.3 Attributes

Read-Write, Single Value

2.3.32.4 Range

[0, 1000MB]

2.3.32.5 Description

This property is used to prevent abnormal increases in memory usage due to versioning when large-volume UPDATE batch tasks are performed on memory tables. If the amount of memory that is used increases beyond the value specified in this property, so-called "in-place update"[1] is performed without versioning in order to prevent increased memory usage.

When using versioning while updating records, an X lock is placed on the record, and an IX lock is placed on the table. However, when in-place update is performed, an X lock, that is, an exclusive lock, is placed on the table. Therefore, care must be taken when setting this value as it can degrade the scalability of the corresponding table if the value is set too low. This property value can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.33 LOG_FILE_GROUP_COUNT

2.3.33.1 Data Type

Unsigned integer

2.3.33.2 Default Value

1

2.3.33.3 Attributes

Read-Only, Single Value

2.3.33.4 Range

[1,32]

1. "In-place update" means directly updating the value of a column in an original record without creating another version of the record.

2.3.33.5 Description

This property is related to Log File Group (LFG) functionality. The database administrator (DBA) uses this property to set the number of log file groups used by the system. Log file groups can be defined to increase log writing performance. If multiple log file groups are specified, log files of ALTIBASE HDB are distributed among the multiple locations.

This property requires that the number of paths specified in the LOG_DIR property and the ARCHIVE_DIR property be the same. Regardless of how many paths are specified for LOG_DIR and ARCHIVE_DIR, no two paths can be the same. This parameter cannot be changed after the database has been created.

2.3.34 LOG_IO_TYPE

2.3.34.1 Data Type

Unsigned Integer

2.3.34.2 Default Value

1

2.3.34.3 Attributes

Read-Only, Single Value

2.3.34.4 Range

[0, 1]

2.3.34.5 Description

This indicates the I/O mode used to write logs.

0: buffered I/O

1: direct I/O

2.3.35 LOW_FLUSH_PCT

2.3.35.1 Data Type

Unsigned Integer

2.3.35.2 Default Value

1

2.3.35.3 Attributes

Read-Write, Single Value

2.3.35.4 Range

[0, 100]

2.3.35.5 Description

If the length of the flush list becomes equal to or greater than the percentage of the total buffer size specified by this value, replacement flushing occurs. At this time, all update buffers in the flush list are flushed.

2.3.36 LOW_PREPARE_PCT

2.3.36.1 Data Type

Unsigned Integer

2.3.36.2 Default Value

1

2.3.36.3 Attributes

Read-Write, Single Value

2.3.36.4 Range

[0, 100]

2.3.36.5 Description

When the flusher awakes from a waiting state, if the length of the Prepare list is less than or equal to the percentage of the total buffer size specified by this value, replacement flushing occurs. At this time, all update buffers in the flush list are flushed.

2.3.37 MAX_FLUSHER_WAIT_SEC

2.3.37.1 Data Type

Unsigned Integer

2.3.37.2 Default Value

10

2.3.37.3 Attributes

Read-Write, Single Value

2.3.37.4 Range

[1, 2^{32} – 1]

2.3.37.5 Description

This property specifies the maximum number of seconds that the flusher waits. The flusher wait time can increase depending on the frequency with which a task is conducted, but cannot exceed this value.

2.3.38 MULTIPLEXING_CHECK_INTERVAL

2.3.38.1 Data Type

Unsigned Integer

2.3.38.2 Default Value

200000

2.3.38.3 Attributes

Read-Write, Single Value

2.3.38.4 Range

[100000, 10000000]

2.3.38.5 Description

This property indicates the interval at which sessions are checked, so that the thread manager service thread can be distributed. It is expressed in units of microseconds.

The thread manger periodically checks the status of threads, updates statistical data, and adds and deletes service threads.

2.3.39 MULTIPLEXING_MAX_THREAD_COUNT

2.3.39.1 Data Type

Unsigned Integer

2.3.39.2 Default Value

1024

2.3.39.3 Attributes

Read-Write, Single Value

2.3.39.4 Range

[1, 1024]

2.3.39.5 Description

This is the maximum number of multiplex threads.

If the capacity of existing threads is exceeded, new threads are automatically added. However, because performance can suffer if new threads are continually created, care must be taken to set this property appropriately.

Nevertheless, when queuing (QUEUE) is used, a number of threads exceeding the value specified by this property can be created.

2.3.40 MULTIPLEXING_THREAD_COUNT

2.3.40.1 Data Type

Unsigned Integer

2.3.40.2 Default Value

The number of CPUs in the host

2.3.40.3 Attributes

Read-Only, Single Value

2.3.40.4 Range

[1, 1024]

2.3.40.5 Description

This is the minimum number of shared service threads that ALTIBASE HDB keeps running. The default is the number of CPUs. This parameter cannot be changed after the server has been started.

2.3.41 NORMALFORM_MAXIMUM

2.3.41.1 Data Type

Unsigned integer

2.3.41.2 Default Value

128

2.3.41.3 Attributes

Read-Write, Single Value

2.3.41.4 Range

[1, 2^{32} – 1]

2.3.41.5 Description

This property specifies the maximum number of normal form nodes when normalizing a condition clause. When the predicates in a WHERE statement of a SELECT query are complicated by the use of logical operators (AND, OR), ALTIBASE HDB normalizes the predicates so that the table(s) can be searched more quickly.

There are two normalization methods: Conjunctive Normal Form (CNF) and Disjunctive Normal Form (DNF). If the use of either of these normal forms results in the number of nodes specified here being exceeded, no attempt to perform normalization using that normal form is made.

If both of the normal forms exceed the number specified here, execution proceeds without the condition clause being normalized. In this case, because the condition clause has not been normalized, an index cannot be used. On the other hand, if the value specified here is exceeded, the process of normalizing the complicated condition clause can use vast amounts of memory, thus the normalizing process itself becomes so expensive that it results in a decrease in performance.

Therefore, it is important to avoid the excessive use of logical operators when writing condition clauses, and to write condition clauses in normal forms.

Similarly, this rule also applies to an ON predicate joined to an ON condition.

2.3.42 OPTIMIZER_MODE

2.3.42.1 Data Type

Unsigned integer

2.3.42.2 Default Value

0

2.3.42.3 Attributes

Read-Write, Single Value

2.3.42.4 Range

[0, 1]

2.3.42.5 Description

If this property is set to 0, cost-based optimization will be used to optimize query statements, whereas if it is set to 1, rule-based optimization will be used. This property can be changed using the ALTER SYSTEM or ALTER SESSION statement while ALTIBASE HDB is running.

2.3.43 PARALLEL_LOAD_FACTOR

2.3.43.1 Data Type

Unsigned integer

2.3.43.2 Default Value

The Number of CPUs

2.3.43.3 Attributes

Read-Only, Single Value

2.3.43.4 Range

[1, 128]

2.3.43.5 Description

This property controls the number of database refinement and index rebuilding threads that are created to refine the database or rebuild indexes when an ALTIBASE HDB server is restarted.

If this property is commented out, the default system behavior is to generate a number of parallel threads equal to the number of CPUs.

2.3.44 PREPARE_STMT_MEMORY_MAXIMUM

2.3.44.1 Data Type

Unsigned Long

2.3.44.2 Default Value

100M

2.3.44.3 Attributes

Read-Write, Single Value

2.3.44.4 Range

[1024*1024, 2^{64} – 1]

2.3.44.5 Description

This property indicates the maximum amount of memory, in bytes, that can be used to prepare a query statement . This property may be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.45 REFINE_PAGE_COUNT

2.3.45.1 Data Type

Unsigned integer

2.3.45.2 Default Value

50

2.3.45.3 Attributes

Read-Only, Single Value

2.3.45.4 Range

[0, 2^{32} – 1]

2.3.45.5 Description

One of the ALTIBASE HDB startup steps handles database refinement. When the ALTIBASE HDB server was shut down the previous time, some so-called “versioning records” created by transactions are not handled by the garbage collector, and thus unneeded records may exist in the database, and furthermore, other versioning records created by recovery processes when the server is started up may also exist. The database refining step is conducted so that these records can be reused. Because this process can be time-consuming when many records are to be refined, it is conducted in parallel by multiple threads. This property specifies the number of pages handled by each thread.

2.3.46 SHM_PAGE_COUNT_PER_KEY

2.3.46.1 Data Type

Unsigned integer

2.3.46.2 Default Value

3200

2.3.46.3 Attributes

Read-Write, Single Value

2.3.46.4 Range

[320, $2^{32} - 1$]

2.3.46.5 Description

This property determines how many pages are allocated to each shared memory key. This property is relevant when a database is a shared memory type database.

For a database that uses shared memory, when the amount of memory is insufficient and thus needs to be increased, the shared memory area is allocated by the OS. This property indicates the number of pages by which to increase the size of the database. A new shared memory key is needed.

However, if this value is too small, a large number of shared memory chunks will be assigned, each having its own shared memory key. Consequently, because shared memory keys are a limited resource, the problem can arise in which the database needs to be shut down, shared memory cleared, and the database started up again. To prevent this, the initial value of this property should be set to a suitable size.

2.3.47 SORT_AREA_SIZE

2.3.47.1 Data Type

Unsigned long

2.3.47.2 Default Value

1048576

2.3.47.3 Attributes

Read-Write, Single Value

2.3.47.4 Range

[8192, 2^{64} – 1]

2.3.47.5 Description

This property indicates the amount of memory, in bytes, that will be used when keys extracted from data are sorted while a disk index is created.

This property can be changed using the ALTER SYSTEM statement while the system is running.

2.3.48 SQL_PLAN_CACHE_BUCKET_CNT

2.3.48.1 Data Type

Unsigned Integer

2.3.48.2 Default Value

127

2.3.48.3 Attributes

Read-Only, Single Value

2.3.48.4 Range

[5, 4096]

2.3.48.5 Description

This property inidicates the number of hash table buckets in a SQL plan cache.

2.3.49 SQL_PLAN_CACHE_HOT_REGION_LRU_RATIO

2.3.49.1 Data Type

Unsigned Integer

2.3.49.2 Default Value

50

2.3.49.3 Attributes

Read-Write, Single Value

2.3.49.4 Range

[10, 90]

2.3.49.5 Description

This property inidicates the percentage of a hot area in an LRU list in a SQL plan cache. A HOT area in an LRU list is a separate portion of an LRU list in a SQL plan cache in which plans that are referred to frequently are saved.

This property can be changed using the ALTER SYSTEM statement while the system is running.

2.3.50 SQL_PLAN_CACHE_PREPARED_EXECUTION_CONTEXT_CNT

2.3.50.1 Data Type

Unsigned Integer

2.3.50.2 Default Value

1

2.3.50.3 Attributes

Read-Write, Single Value

2.3.50.4 Range

[0, 1024]

2.3.50.5 Description

This property indicates the number of execution contexts that are initially created when plans are generated.

The initial number of execution contexts is specified before plans are created, however, this only determines the initial number. The number of execution contexts increases or decreases automatically as required during runtime.

Increasing this value can help realize better performance when only one plan is executed at a time, however, in other cases the plan size is merely increased, without realizing improved performance.

2.3.51 SQL_PLAN_CACHE_SIZE

2.3.51.1 Data Type

Unsigned long

2.3.51.2 Default Value

64 M

2.3.51.3 Attributes

Read-Write, Single Value

2.3.51.4 Range

[0, $2^{64} - 1$]

2.3.51.5 Description

This property indicates the maximum size, in bytes, of the SQL plan cache. If set to 0, the cache can't be used. This property can be checked by viewing the value of MAX_CACHE_SIZE of V$SQL_PLAN_CACHE.

This property can be changed using the ALTER SYSTEM statement while the system is running.

2.3.52 STATEMENT_LIST_PARTIAL_SCAN_COUNT

2.3.52.1 Data Type

Unsigned Integer

2.3.52.2 Default Value

0

2.3.52.3 Attributes

Read-Write, Single Value

2.3.52.4 Range

[0, $2^{32} - 1$]

2.3.52.5 Description

This property indicates the maximum number of statements to return to the application in response to a SELECT query executed on V$STATEMENT, V$SQLTEXT, or V$PLANTEXT. If this property is set to 0, all rows pertaining to all statements are returned.

This property can be changed using the ALTER SYSTEM statement while the system is running.

2.3.53 TABLE_INITRANS

2.3.53.1 Data Type

Unsigned Integer

2.3.53.2 Default Value

2

2.3.53.3 Attributes

Read-Only, Single Value

2.3.53.4 Range

[0, 120]

2.3.53.5 Description

This property indicates the initial number of TTS (Touched Transaction Slots) to be maintained in a table page.

2.3.54 TABLE_LOCK_ENABLE

2.3.54.1 Data Type

Unsigned integer

2.3.54.2 Default Value

1

2.3.54.3 Attributes

Read-Write, Single Value

2.3.54.4 Range

[0, 1]

2.3.54.5 Description

This property controls the lock level.

If this parameter is set to 1, which is the default, both table-level locks and record-level locks are enabled. If the parameter is set to 0, table locks are disabled, and only record-level locks are enabled, which realizes the benefit of improved performance of simple DML statements.

However, when this property is set to 0, the following restrictions apply:

- DDL statements cannot be executed.
- CREATE DATABASE cannot be executed.
- When performing replication, parallel SYNC cannot be used.

This property can be changed using the ALTER SYSTEM statement.

2.3.55 TABLE_MAXTRANS

2.3.55.1 Data Type

Unsigned Integer

2.3.55.2 Default Value

120

2.3.55.3 Attributes

Read-Only, Single Value

2.3.55.4 Range

[0, 120]

2.3.55.5 Description

This property indicates the maximum size of the ITL (Interested Transaction List) that is maintained for one table.

2.3.56 TIMER_RUNNING_LEVEL

2.3.56.1 Data Type

Unsigned Integer

2.3.56.2 Default Value

The default value for this property differs depending on the platform as follows:

1: all platforms not listed below

2: sparc-solaris, X86-solaris, IBM-AIX, PA-RISC-HP-64, IA64-HP

3: x86-linux, Amd64-linux

2.3.56.3 Attributes

Read-Write, Single Value

2.3.56.4 Range

[1, 3]

2.3.56.5 Description

This property specifies how to measure the wait time for wait events and the time required for SQL operations.

1: The time measurement thread measures the time at regular intervals specified in the property TIMER_THREAD_RESOLUTION.

2: The time is measured using the library functions provided with respective platforms.

3: This method is similar to #1, but the time is measured using the system clock. Therefore, this method doesn't hinder performance as much as the other methods.

2.3.57 TIMED_STATISTICS

2.3.57.1 Data Type

Unsigned Integer

2.3.57.2 Default Value

0

2.3.57.3 Attributes

Read-Write, Single Value

2.3.57.4 Range

[0, 1]

2.3.57.5 Description

This property determines whether to measure the wait time for wait events and the time required for SQL operations. Using this property to specify that the time is to be measured can negatively impact performance.

0: do not measure the time

1: measure the time

2.3.58 TIMER_THREAD_RESOLUTION

2.3.58.1 Data Type

Unsigned Integer

2.3.58.2 Default Value

1000

2.3.58.3 Attributes

Read-Write, Single Value

2.3.58.4 Range

[50, 10000000]

2.3.58.5 Description

If the TIMER_RUNNING_LEVEL property is set to 1, this property indicates the interval, in microseconds, at which to conduct measurements.

2.3.59 TOUCH_TIME_INTERVAL

2.3.59.1 Data Type

Unsigned Integer

2.3.59.2 Default Value

3

2.3.59.3 Attributes

Read-Write, Single Value

2.3.59.4 Range

[0, 100]

2.3.59.5 Description

This property specifies the minimum time interval, in seconds, at which to increase the buffer access count. After the value specified in this property has passed since the last time the buffer was accessed, the access count is increased.

If this property is set to 3, which is the default value, the access count is not updated if a particular buffer is accessed again less than 3 seconds since it was previously accessed.

2.3.60 TRANSACTION_SEGMENT_COUNT

2.3.60.1 Data Type

Unsigned Integer

2.3.60.2 Default Value

256

2.3.60.3 Attributes

Read-Write, Single Value

2.3.60.4 Range

[1, 512]

2.3.60.5 Description

This property specifies the number of transaction segments (Undo segments and TTS segments) created when the server is started. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.3.61 TRX_UPDATE_MAX_LOGSIZE

2.3.61.1 Data Type

Unsigned Integer

2.3.61.2 Default Value

10M

2.3.61.3 Attributes

Read-Write, Single Value

2.3.61.4 Range

[0, $2^{64} - 1$]

2.3.61.5 Description

If the size of a log created by a DML statement becomes greater than the number of bytes specified in this property, the corresponding transaction is aborted and an error is returned. This property is used to prevent unusual increases in system load attributable to large volume batch tasks that result from the user's carelessness.

Because the log size has no limit if this property is set to 0, logs can be used without limit when records are updated. This property can be changed using the ALTER SYSTEM or ALTER SESSION statement when ALTIBASE HDB is running.

2.4 Session Properties

Session-related properties define the rules for communication between clients and the database server when ALTIBASE HDB is run in a client-server configuration. They are as follows:

2.4.1 CM_DISCONN_DETECT_TIME

2.4.1.1 Data Type

Unsigned integer

2.4.1.2 Default Value

3

2.4.1.3 Attributes

Read-Only, Single Value

2.4.1.4 Range

[1, $2^{32} - 1$]

2.4.1.5 Description

ALTIBASE HDB server provides a session management thread ("cm detector") for checking whether the connection between a client and a server has been interrupted. This property specifies the interval, in seconds, at which the session management thread operates. Usually, when a client process is abnormally terminated, the server to which the client is connected can immediately detect this.

However, when a session has an unfinished task, and furthermore if the task is an internal ALTIBASE HDB server operation that is not directly related to the client session, and it is taking a long time, the server cannot check whether the client has terminated abnormally. That is to say, because the server cannot check whether the connection with the client has ended abnormally, such abnormal termination would be disregarded and ALTIBASE HDB would continue to process the task. Such sessions must be detected, and the corresponding transactions must be rolled back. For this purpose, the session management thread regularly checks the status of all sessions.

2.4.2 DEFAULT_THREAD_STACK_SIZE

2.4.2.1 Data Type

Unsigned Integer

2.4.2.2 Default Value

1048576

2.4.2.3 Attributes

Read-Only, Single Value

2.4.2.4 Range

[8192, 10485760]

2.4.2.5 Description

This property specifies the stack size, in bytes, for all system threads other than service threads. The service thread stack size is set using the SERVICE_THREAD_STACK_SIZE property.

2.4.3 IPC_CHANNEL_COUNT

2.4.3.1 Data Type

Unsigned integer

2.4.3.2 Default Value

0

2.4.3.3 Attributes

Read-Only, Single Value

2.4.3.4 Range

[0, 65535]

2.4.3.5 Description

This property, which specifies the maximum number of IPC communication channels between a client and an ALTIBASE HDB server, must be set. Because shared memory and semaphore(s) are allocated in proportion to the channel count, it is important to set the maximum number of IPC connections that can be simultaneously established with the server.

2.4.4 IPC_PORT_NO

2.4.4.1 Data Type

Unsigned Integer

2.4.4.2 Default Value

20350

2.4.4.3 Attributes

Read-Only, Single Value

2.4.4.4 Range

[1024, 65535]

2.4.4.5 Description

This property specifies the TCP port number for use in establishing client-server IPC connections in a Windows environment. In a Unix environment, Unix domain sockets can be used for IPC connections, but as they cannot be used in Windows, this port number is necessary.

The client receives the shared memory name, semaphore and mutex name via a TCP connection, and then uses that information to connect via IPC.

2.4.5 MAX_LISTEN

2.4.5.1 Data Type

Unsigned integer

2.4.5.2 Default Value

128

2.4.5.3 Attributes

Read-Only, Single Value

2.4.5.4 Range

[0, 512]

2.4.5.5 Description

This property specifies the maximum size of the “listen queue” when TCP/IP or UNIX domain protocol is used for communication between a client and ALTIBASE HDB.

2.4.6 MAX_STATEMENTS_PER_SESSION

2.4.6.1 Data Type

Unsigned Integer

2.4.6.2 Default Value

1024

2.4.6.3 Attributes

Read-Write, Single Value

2.4.6.4 Range

[1, 2^{32}-1]

2.4.6.5 Description

This property specifies the maximum number of statements that can be executed in a session.

2.4.7 NET_CONN_IP_STACK

2.4.7.1 Data Type

Unsigned Integer

2.4.7.2 Default Value

0

2.4.7.3 Attributes

Read-Only, Single Value

2.4.7.4 Range

[0, 1, 2]

2.4.7.5 Description

This property specifies the Internet Protocol Stack to be used when creating sockets on the server side for communication between the client and the server via TCP/IP.

0: An Internet Protocol Stack supporting only IPv4 will be used.

1: A dual stack (Internet Protocol Stack supporting both IPv4 and IPv6) will be used.

2: An Internet Protocol Stack supporting only IPv6 will be used.

2.4.8 NLS_NCHAR_CONV_EXCP

2.4.8.1 Data Type

Unsigned Integer

2.4.8.2 Default Value

0

2.4.8.3 Attributes

Read-Write, Single Value

2.4.8.4 Range

[0, 1]

2.4.8.5 Description

When an NCHAR data type is converted to another character set, data loss can occur. In such cases, this property determines whether to raise an error or to continue converting the data despite the possibility of data loss.

In order to ensure that this property raises an error only when the server performs data type conversion from other character sets to NCHAR, this property doesn't apply to conversion performed on the client. This property can be changed using the ALTER SESSION statement while ALTIBASE HDB is running.

0: FALSE (Do not raise an error.)

1: TRUE

2.4.9 NLS_COMP

2.4.9.1 Data Type

Unsigned Integer

2.4.9.2 Default Value

0

2.4.9.3 Attributes

Read-Only, Single Value

2.4.9.4 Range

[0, 1]

2.4.9.5 Description

When a database is created, it cannot be guaranteed that the sequence of characters in the character set specified by NLS_USE is the same as in a dictionary for the language of the country in question.

If this property is set to 1, each character set is compared in the order in which words in that language appear in a dictionary. This is supported only when the database character set is set to Korean (KSC-5601 complete and MS extended complete) because the system currently supports Korean only.

2.4.10 PORT_NO

2.4.10.1 Data Type

Unsigned integer

2.4.10.2 Default Value

20300

2.4.10.3 Attributes

Read-Only, Single Value

2.4.10.4 Range

[1024, 65535]

2.4.10.5 Description

This property specifies the port number for communication between the client and the server via TCP/IP. The user can set this port number to any number not being used by another application within the range of port numbers (up to number 65535) excluding the so-called “well-known TCP port numbers” (from 1 to 1023). Application programs of ALTIBASE HDB connect to the server via this port number. ALTIBASE HDB obtains the value of PORT_NO from the $ALTIBASE_HOME/conf/

altibase.properties file. This property must be set even when the client and the server are on different computers.

2.4.11 PSM_FILE_OPEN_LIMIT

2.4.11.1 Data Type

Unsigned integer

2.4.11.2 Default Value

16

2.4.11.3 Attributes

Read-Write, Single Value

2.4.11.4 Range

[0,128]

2.4.11.5 Description

This property specifies the maximum number of stored procedure file handlers that can be opened for a session.

2.4.12 SERVICE_THREAD_STACK_SIZE

2.4.12.1 Data Type

Unsigned Integer

2.4.12.2 Default Value

1048576

2.4.12.3 Attributes

Read-Only, Single Value

2.4.12.4 Range

[8192, 10485760]

2.4.12.5 Description

This property specifies the stack size, in bytes, for the service thread of ALTIBASE HDB. The thread stack size is limited by the OS on which ALTIBASE HDB is installed. Please note that the stack size for all system threads other than service threads is set using DEFAULT_THREAD_STACK_SIZE.

2.4.13 USE_MEMORY_POOL

2.4.13.1 Data Type

Unsigned Integer

2.4.13.2 Default Value

1

2.4.13.3 Attributes

Read-Only, Single Value

2.4.13.4 Range

[0,1]

2.4.13.5 Description

This property specifies whether memory pooling is used. “Memory pooling” means assigning server memory in advance.

When this function is used, because server memory is allocated in advance, memory use is increased.

0: do not use memory pooling

1: use memory pooling

2.4.14 XA_HEURISTIC_COMPLETE

2.4.14.1 Data Type

Unsigned integer

2.4.14.2 Default Value

0

2.4.14.3 Attributes

Read-Only, Single Value

2.4.14.4 Range

[0, 2]

2.4.14.5 Description

In a distributed transaction environment, Two-Phase Commit Protocol (XA) is used. While a transaction is underway, after a Prepare command has been received from the global transaction coordinator, if for some reason a COMMIT or ROLLBACK command does not arrive for a long time, ALTIBASE HDB will keep the transaction active for a long time, which will negatively affect database performance.

To prevent this, ALTIBASE HDB terminates the entire transaction if it has been in a PREPARE (or IN_DOUBT) state beyond a certain period of time. In such cases, this property determines whether to use COMMIT or ROLLBACK to terminate the transaction.

ALTIBASE HDB waits for the amount of time specified with the XA_INDOUBT_TX_TIMEOUT property before cancelling a transaction in this way. If the value of XA_HEURISTIC_COMPLETE is 0, which is the default, nothing will be done; if it is 1, the transaction will be committed, and if it is 2, the transaction will be rolled back.

2.5 Time-Out Properties

2.5.1 BLOCK_ALL_TX_TIME_OUT

2.5.1.1 Data Type

Unsigned Integer

2.5.1.2 Default Value

3 (seconds)

2.5.1.3 Attributes

Read-Write, Single Value

2.5.1.4 Range

[0, 2^{32} - 1]

2.5.1.5 Description

This property restricts transactions' access to the hash table when the buffer manager resizes the hash table. The minimum value of 0 specifies that error handling is to be performed without any wait time. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.2 DDL_LOCK_TIMEOUT

2.5.2.1 Data Type

Short integer

2.5.2.2 Default Value

0

2.5.2.3 Attributes

Read-Write, Single Value

2.5.2.4 Range

[-1, 65535]

2.5.2.5 Description

When DDL query statements are executed, this property sets how long to wait to establish a lock when the target table has already been locked by another transaction. In cases where a transaction cannot immediately gain write access to the table, If this parameter is set to -1, the transaction will wait indefinitely, whereas if this parameter is set to a positive value, the transaction will wait for that number of seconds before trying again.

The default value of this parameter is 0, which tells ALTIBASE HDB to return an error code if it cannot obtain a lock immediately at the time of executing a DDL statement. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.3 FETCH_TIMEOUT

2.5.3.1 Data Type

Unsigned integer

2.5.3.2 Default Value

60

2.5.3.3 Attributes

Read-Write, Single Value

2.5.3.4 Range

[0, 2^{32}- 1]

2.5.3.5 Description

This property prevents abnormal increases in database memory consumption when SELECT statements executed by client applications take an excessive amount of time. In cases where the query execution time exceeds the number of seconds specified using this property, the session will be disconnected and the transaction will be rolled back. This property can be changed using the ALTER SYSTEM or ALTER SESSION statement while ALTIBASE HDB is running.

2.5.4 IDLE_TIMEOUT

2.5.4.1 Data Type

Unsigned integer

2.5.4.2 Default Value

0

2.5.4.3 Attributes

Read-Write, Single Value

2.5.4.4 Range

[0, 2^{32} – 1]

2.5.4.5 Description

If a large number of clients are connected to a server for an excessive period of time due to some abnormality, the number of available connections will significantly decrease, ultimately leading to failure to provide service.

This property functions to preemptively prevent this situation. If the number of seconds that a session is idle exceeds this value, the session will be disconnected and any associated transactions will be rolled back. The value of this property can be changed using the ALTER SYSTEM or ALTER SESSION statement while ALTIBASE HDB is running.

2.5.5 LINKER_CONNECT_TIMEOUT

2.5.5.1 Data Type

Unsigned Integer

2.5.5.2 Default Value

225

2.5.5.3 Attributes

Read-Only, Single Value

2.5.5.4 Range

[0, 2^{32} - 1]

2.5.5.5 Description

This property specifies the connection timeout, in seconds, when the ALTIBASE HDB server attempts to establish a connection to another server using AltiLinker.

2.5.6 LINKER_RECEIVE_TIMEOUT

2.5.6.1 Data Type

Unsigned Integer

2.5.6.2 Default Value

300

2.5.6.3 Attributes

Read-Only, Single Value

2.5.6.4 Range

[0, 2^{32}- 1]

2.5.6.5 Description

This property specifies the wait time, in seconds, when an ALTIBASE HDB server is exchanging data with AltiLinker.

2.5.7 LOGIN_TIMEOUT

2.5.7.1 Data Type

Unsigned Integer

2.5.7.2 Default Value

0

2.5.7.3 Attributes

Read-Write, Single Value

2.5.7.4 Range

[0, 2^{32} - 1]

2.5.7.5 Description

This property specifies the permitted amount of time, in seconds, to wait for authorization to be completed after a connection has been made to an ALTIBASE HDB port. If authorization is not completed within this time, the server disconnects.

2.5.8 MULTIPLEXING_POLL_TIMEOUT

2.5.8.1 Data Type

Unsigned Integer

2.5.8.2 Default Value

10000

2.5.8.3 Attributes

Read-Write, Single Value

2.5.8.4 Range

[1000, 1000000]

2.5.8.5 Description

This property specifies the interval, in microseconds, at which the multiplexed thread running service detects sessions.

2.5.9 QUERY_TIMEOUT

2.5.9.1 Data Type

Unsigned integer

2.5.9.2 Default Value

600

2.5.9.3 Attributes

Read-Write, Single Value

2.5.9.4 Range

[0, $2^{32} – 1$]

2.5.9.5 Description

This property is set to prevent abnormal increases in database size when particular kinds of queries (especially those involving sort operations or joins) are executed. If the query execution time exceeds the number of seconds specified here, the transaction is partially rolled back. This property can be changed using the ALTER SYSTEM or ALTER SESSION statement while ALTIBASE HDB is running.

2.5.10 REMOTE_SERVER_CONNECT_TIMEOUT

2.5.10.1 Data Type

Unsigned Integer

2.5.10.2 Default Value

5

2.5.10.3 Attributes

Read-Only, Single Value

2.5.10.4 Range

[0, 2^{32} - 1]

2.5.10.5 Description

This property specifies the amount of time, in seconds, to wait for AltiLinker to connect to a remote server.

2.5.11 REPLICATION_CONNECT_TIMEOUT

2.5.11.1 Data Type

Unsigned integer

2.5.11.2 Default Value

10

2.5.11.3 Attributes

Read-Write, Single Value

2.5.11.4 Range

[0, 2^{32} - 1]

2.5.11.5 Description

When attempting to connect to a target host to perform replication, if there is no response within the number of seconds specified in this property, no further connection attempts are made. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.12 REPLICATION_LOCK_TIMEOUT

2.5.12.1 Data Type

Unsigned integer

2.5.12.2 Default Value

5

2.5.12.3 Attributes

Read-Write, Single Value

2.5.12.4 Range

[0, 3600]

2.5.12.5 Description

When a replication deadlock occurs, the Receiver thread will wait indefinitely to establish a lock, which may result in a service interruption. To prevent this, when the Receiver thread requests a lock to perform this kind of operation, it will only wait for the number of seconds specified using this property.

If a lock cannot be acquired within the given time, the corresponding operation will be rolled back.

2.5.13 REPLICATION_RECEIVE_TIMEOUT

2.5.13.1 Data Type

Unsigned integer

2.5.13.2 Default Value

300

2.5.13.3 Attributes

Read-Write, Single Value

2.5.13.4 Range

[0, 2^{32} - 1]

2.5.13.5 Description

This property, which is used by both the Sender thread and the Receiver thread, specifies the maximum amount of time, in seconds, to wait for a message from the Receiver or Sender thread, respectively.

In the case where the Sender thread has waited for a response from the Receiver thread for the maximum amount of time specified here, the Sender thread will enter into sleep mode for the amount of time specified using the REPLICATION_SENDER_SLEEP_TIMEOUT property before again attempting to connect to the Receiver thread. In this case, the existing socket is closed and a new socket is created for the new connection attempt.

This property also specifies the maximum time that the Receiver thread waits for a message from a Sender thread. If the specified amount of time has passed, the Receiver thread is automatically terminated, and a new Receiver thread will be created when the Sender thread sends a message. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.14 REPLICATION_SENDER_SLEEP_TIMEOUT

2.5.14.1 Data Type

Unsigned integer

2.5.14.2 Default Value

10 (microseconds)

2.5.14.3 Attributes

Read-Write, Single Value

2.5.14.4 Range

[0, 2^{32} - 1]

2.5.14.5 Description

This property specifies the number of microseconds that a replication Sender thread that is in an error state must sleep before trying again. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.15 REPLICATION_SYNC_LOCK_TIMEOUT

2.5.15.1 Data Type

Unsigned integer

2.5.15.2 Default Value

30

2.5.15.3 Attributes

Read-Write, Single Value

2.5.15.4 Range

[1, 2^{32} - 1]

2.5.15.5 Description

When replication synchronization is performed, the Replication Sender Thread determines the current position in the log at which replication will start after synchronization. In order to prevent another transaction from changing the data in the table on which synchronization is to be performed right at the time of this determination, the Replication Sender Thread obtains an S Lock on the table on which synchronization is to be performed for a short time before synchronization. This property specifies the amount of time, in seconds, to wait to establish a lock when a table to be synchronized has been locked by another transaction. If a lock is requested but cannot be obtained immediately, the replication process will wait for the amount of time specified here. If a lock cannot be obtained within the amount of time specified here, the synchronization attempt will be handled as an error. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.16 SHUTDOWN_IMMEDIATE_TIMEOUT

2.5.16.1 Data Type

Unsigned integer

2.5.16.2 Default Value

60

2.5.16.3 Attributes

Read-Write, Single Value

2.5.16.4 Range

[0, $2^{32} - 1$]

2.5.16.5 Description

When shutting down ALTIBASE HDB with the IMMEDIATE option, ALTIBASE HDB is shut down after uncommitted transactions are rolled back. This property specifies the amount of time, in seconds, to wait for the transactions to be rolled back. If the elapsed time exceeds the specified value, ALTIBASE HDB is shut down forcibly and uncommitted transactions are not rolled back. If this property is set to 0, ALTIBASE HDB waits until all transactions are rolled back. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.5.17 UTRANS_TIMEOUT

2.5.17.1 Data Type

Unsigned integer

2.5.17.2 Default Value

3600

2.5.17.3 Attributes

Read-Write, Single Value

2.5.17.4 Range

[0, $2^{32} - 1$]

2.5.17.5 Description

This property is set to prevent the number of log files from abnormally increasing when write operations (UPDATE, DELETE, INSERT) take a long time. If such a transaction takes longer than the number of seconds specified here, the session will be disconnected and the transaction in question will be rolled back. This property can be changed using the ALTER SYSTEM or ALTER SESSION statement while ALTIBASE HDB is running.

2.5.18 XA_INDOUBT_TX_TIMEOUT

2.5.18.1 Data Type

Unsigned integer

2.5.18.2 Default Value

60

2.5.18.3 Attributes

Read-Only, Single Value

2.5.18.4 Range

[0, $2^{32} - 1$]

2.5.18.5 Description

When using the Two-Phase Commit Protocol, this property specifies the number of seconds to wait before terminating an entire transaction that has taken a long time and is thus in IN_DOUBT state.

2.6 Transaction Properties

2.6.1 AUTO_COMMIT

2.6.1.1 Data Type

Unsigned integer

2.6.1.2 Default Value

1

2.6.1.3 Attributes

Read-Write, Single Value

2.6.1.4 Range

[0, 1]

2.6.1.5 Description

This property determines whether to handle each individual SQL statement as a separate transaction and commit it when SQL statements are executed in a session. A value of 1 indicates auto-commit mode, while a value of 0 indicates non-autocommit mode. When using non-autocommit mode, the client application must explicitly indicate the beginning and end of a transaction.

Even if this value is set to 1, indicating auto-commit, when the server is started, this property can be changed for individual sessions. For example, if ALTER SESSION SET AUTOCOMMIT = FALSE (non-autocommit) is executed from a client, the user must explicitly specify whether to commit or rollback any transactions that occur for the remainder of the session. This property can be changed using the ALTER SYSTEM and ALTER SESSION statement while ALTIBASE HDB is running.

2.6.2 ISOLATION_LEVEL

2.6.2.1 Data Type

Unsigned integer

2.6.2.2 Default Value

0

2.6.2.3 Attributes

Read-Only, Single Value

2.6.2.4 Range

[0, 3]

2.6.2.5 Description

This property specifies the transaction isolation level. When a single transaction searches the same table multiple times, the result varies depending on the isolation level. For more information about transaction isolation levels, please refer to the *ALTIBASE HDB Administrator's Manual*.

Isolation Level	Characteristics
0 (Committed Read)	This is default mode of ALTIBASE HDB. This isolation level guarantees that previously read data that have been modified by another transaction will reflect the changes of that other transaction.When a SELECT transaction reads data one time and then reads the data again, if another transaction simultaneously executes and commits an INSERT or DELETE statement, due to this change, it is possible for a new row to be found, or for a previously found row to have disappeared.
1 (Repeatable Read)	This isolation level guarantees that the contents of a row will be the same upon repeated reads by the same transaction.This isolation level places a lock on a row once it has been read. Therefore, when the table is subsequently read, previously read rows will not change or disappear, but it is possible for new rows to appear.
2 (No Phantom)	This isolation level guarantees identical results for repeated reads.

2.6.3 TRANSACTION_TABLE_SIZE

2.6.3.1 Data Type

Unsigned integer

2.6.3.2 Default Value

1024

2.6.3.3 Attributes

Read-Write, Single Value

2.6.3.4 Range

[16, 1024 * 10]

2.6.3.5 Description

This property specifies the maximum number of concurrent transactions while ALTIBASE HDB is running, for which memory is allocated in advance.

2.7 Backup and Recovery Properties

These properties are related to the management of change logs, which are maintained in response to database changes.

2.7.1 ARCHIVE_DIR

2.7.1.1 Data Type

String

2.7.1.2 Default Value

$ALTIBASE_HOME/arch_logs

2.7.1.3 Attributes

Read-Only, Multiple Values

2.7.1.4 Range

None

2.7.1.5 Description

This property specifies the directory or directories in which to store archive log files when performing an archive log backup. If this value is not expressly specified by the user, the default location is $ALTIBASE_HOME/arch_logs.

The number of directories specified in this property must be the same as the number specified in the LOG_DIR property. Furthermore, when multiple values are specified in the LOG_DIR property, the ARCHIVE_DIR property values and the LOG_DIR property values must be specified in sequence, such that they are individually mapped 1:1. The user can explicitly specify the value(s), but the specified directories must be created first. If not, an error message will be output, and ALTIBASE HDB will not start.

2.7.2 ARCHIVE_FULL_ACTION

2.7.2.1 Data Type

Unsigned integer

2.7.2.2 Default Value

0

2.7.2.3 Attributes

Read-Only, Single Value

2.7.2.4 Range

[0, 1]

2.7.2.5 Description

This property controls the action of the archivelog thread, which conducts archive log backup, when there is not enough disk space in the archive log destination (specified using ARCHIVE_DIR).

If this parameter is set to 0, the archivelog thread will output an error message and stop the archive log file backup. Even if enough disk space can subsequently be secured, archive log backup will not resume until the user explicitly issues a command to do so. In such cases, if checkpointing takes place, unnecessary log files will be deleted even if no archive log file backup has been conducted, therefore care must be taken when using this mode.

If this parameter is set to 1, the archivelog thread waits until enough disk space can be secured to perform the archive log file backup. Because the archive log files have not been backed up, care must be taken to prevent the log files from being deleted if checkpointing takes place during this waiting period.

2.7.3 ARCHIVE_THREAD_AUTOSTART

2.7.3.1 Data Type

Unsigned integer

2.7.3.2 Default Value

1

2.7.3.3 Attributes

Read-Only, Single Value

2.7.3.4 Range

[0, 1]

2.7.3.5 Description

This property specifies whether to activate the archivelog thread, which periodically performs archive log file backups. If this property is 1, the archivelog thread is activated.

After the archivelog thread has been suspended due to insufficient disk space in the backup directory, this property is used to restart the thread automatically after sufficient disk space is secured.

2.7.4 CHECKPOINT_ENABLED

2.7.4.1 Data Type

Unsigned integer

2.7.4.2 Default Value

1

2.7.4.3 Attributes

Read-Only, Single Value

2.7.4.4 Range

[0, 1]

2.7.4.5 Description

0: OFF

1: ON

This property specifies whether checkpointing is enabled (“ON”) or disabled (“OFF”).

When this value is 0 (“OFF”), the checkpoint thread cannot be started, and additionally, the user cannot perform checkpointing manually.

2.7.5 CHECKPOINT_INTERVAL_IN_LOG

2.7.5.1 Data Type

Unsigned integer

2.7.5.2 Default Value

100

2.7.5.3 Attributes

Read-Write, Single Value

2.7.5.4 Range

[1, $2^{32} - 1$]

2.7.5.5 Description

This property defines the checkpoint interval based on the log file creation count. In other words, after the log files have been replaced the number of times specified using this property, checkpointing will be automatically executed. When checkpointing is requested based on this property, it may be impossible to execute, either because checkpointing is already underway, or for some other reason.

In such cases, checkpointing is not initiated immediately again after the checkpointing that is already underway has finished; instead, the current checkpointing request is canceled. Therefore, the next checkpointing request will occur when the number of log files reaches the value set in this property. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.7.6 CHECKPOINT_INTERVAL_IN_SEC

2.7.6.1 Data Type

Unsigned integer

2.7.6.2 Default Value

6000

2.7.6.3 Attributes

Read-Write, Single Value

2.7.6.4 Range

[3, $2^{32} - 1$]

2.7.6.5 Description

This property specifies the checkpoint interval in seconds. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.7.7 COMMIT_WRITE_WAIT_MODE

2.7.7.1 Data Type

Unsigned Integer

2.7.7.2 Default Value

0

2.7.7.3 Attributes

Read-Write, Single Value

2.7.7.4 Range

[0, 1]

2.7.7.5 Description

This property specifies whether to wait until logs have been written to log files when committing transactions. In ALTIBASE HDB, the default is not to wait, in the interests of better performance. This property can be set for the entire system or for individual user sessions, and thus this property can be changed using either the ALTER SYSTEM or ALTER SESSION statement while ALTIBASE HDB is running.

0 : Do Not Wait

1 : Wait

2.7.8 LOG_BUFFER_TYPE

2.7.8.1 Data Type

Unsigned Integer

2.7.8.2 Default Value

0

2.7.8.3 Attributes

Read-Only, Single Value

2.7.8.4 Range

[0, 1]

2.7.8.5 Description

This property determines the log buffer type. If it is set to 0, the OS kernel log buffer is used. If it is set to 1, the process memory log buffer is used.

This property cannot be changed while the system is running.

2.7.9 PREPARE_LOG_FILE_COUNT

2.7.9.1 Data Type

Unsigned integer

2.7.9.2 Default Value

5

2.7.9.3 Attributes

Read-Only, Single Value

2.7.9.4 Range

[0, 2^{32} – 1]

2.7.9.5 Description

If there is not enough space in the log file when logs are written, a new log file is created, which can increase the transaction response time. To prevent such delays in transaction execution caused by the creation of log files, ALTIBASE HDB creates extra log files (“prepare log files”) in advance. This parameter specifies the number of such log files.

2.8 Replication Properties

The following parameters pertain to database replication. For more information about database replication, please refer to the *Getting Started Guide* and to the *Replication Manual*.

2.8.1 REPLICATION_ACK_XLOG_COUNT

2.8.1.1 Data Type

Unsigned Integer

2.8.1.2 Default Value

100

2.8.1.3 Attributes

Read-Only, Single Value

2.8.1.4 Range

[0, $2^{32} - 1$]

2.8.1.5 Description

This property indicates the frequency with which the Receiver thread sends ACK to the Sender thread.

The Receiver thread receives XLogs and replays them one by one. When the number of replayed XLogs exceeds the value specified here, the Receiver thread sends ACK to the Sender thread.

If this value is set too low, the Receiver thread sends ACK too often, leading to reduced performance.

If it is set too high, the amount of time that the Sender thread waits for ACK can increase excessively, and may be treated as a network fault. In addition, if the Sender thread does not receive ACK for an extended time, the replication restart SN is not updated, and thus the Sender thread will start over from the most recent log record if checkpointing occurs, resulting in the deletion of unreplicated logs.

2.8.2 REPLICATION_COMMIT_WRITE_WAIT_MODE

2.8.2.1 Data Type

Unsigned integer

2.8.2.2 Default Value

0

2.8.2.3 Attributes

Changeable, Single Value

2.8.2.4 Range

[0, 1]

2.8.2.5 Description

This property determines whether the replication Receiver checks whether XLOGs have been applied to disk after the replication Receiver has completed executing all of the transactions that are necessary in order to apply the contents of XLOGs to disk. If this property is set to 0, the replication Receiver doesn't wait to ensure that the contents of XLOGs have been applied to disk. If the value of this property is set to 1, the replication Receiver ensures that the contents of XLOGs have been applied to disk.

2.8.3 REPLICATION_CONNECT_RECEIVE_TIMEOUT

2.8.3.1 Data Type

Unsigned integer

2.8.3.2 Default Value

60

2.8.3.3 Attributes

Read-Write, Single Value

2.8.3.4 Range

[0, 2^{32} – 1]

2.8.3.5 Description

This property specifies the amount of time, in seconds, to wait after attempting to connect to the target host at the start of replication. This parameter value must be slightly greater than REPLICATION_HBT_DETECT_TIMEOUT. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.4 REPLICATION_DDL_ENABLE

2.8.4.1 Data Type

Unsigned Integer

2.8.4.2 Default Value

0

2.8.4.3 Attributes

Read-Write, Single Value

2.8.4.4 Range

[0, 1]

2.8.4.5 Description

This property specifies whether or not to allow DDL statements to be executed on replication target tables. If this property is set to 1, DDL statements can be executed on replication target tables.

Before executing DDL statements, if the replication property of a transaction in the current session is set to a value other than NONE, the Sender thread can be made aware of the execution of DDL statements.

For a list of DDL statements permitted during replication and other restrictions, please refer to the *ALTIBASE HDB Replication Manual*. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.5 REPLICATION_EAGER_PARALLEL_FACTOR

2.8.5.1 Data Type

Unsigned integer

2.8.5.2 Default Value

the lower of the number of CPUs and 512

2.8.5.3 Attributes

Read-Only, Single Value

2.8.5.4 Range

[1 - 512]

2.8.5.5 Description

When replication is running in EAGER mode, multiple sender threads can work in parallel. The number of sender threads that work in parallel must be specified using this property. If this property is not set, the default value is either the number of CPUs or 512, whichever is lower.

2.8.6 REPLICATION_FAILBACK_INCREMENTAL_SYNC

2.8.6.1 Data Type

Unsigned integer

2.8.6.2 Default Value

1

2.8.6.3 Attributes

Read-Only, Single Value

2.8.6.4 Range

[0, 1]

2.8.6.5 Description

When an ALTIBASE HDB server is started with replication in EAGER mode, service starts after the data are synchronized between the database servers. This property specifies how the data are synchronized between the database servers.

0: The data are synchronized using LAZY mode by eliminating the replication gap. One server does not wait for data to be synchronized on the other server. Therefore, it is recommended that you confirm that the data have been synchronized. If the data to be updated are completely divided between replicated systems in an Active-Active replication environment, 0 should be specified.

1: One of the two database servers is the basis for data synchronization. If both servers have been providing service in an Active-Active replication environment since the occurrence of a network failure, changes that have been made to data on one server during that time will be removed during synchronization. If the data to be updated are the same on replicated systems in an Active-Active replication environment, 1 should be specified.

This property must be set to the same value on both servers.

2.8.7 REPLICATION_HBT_DETECT_HIGHWATER_MARK

2.8.7.1 Data Type

Unsigned integer

2.8.7.2 Default Value

10

2.8.7.3 Attributes

Read-Write, Single Value

2.8.7.4 Range

[0, $2^{32} - 1$]

2.8.7.5 Description

This property specifies the number of failed connection attempts to make before determining that a failure has occurred in a replication environment. Thus, the maximum time that can pass before it is determined that a host has failed can be calculated by multiplying REPLICATION_HBT_DETECT_TIME * REPLICATION_HBT_DETECT_HIGHWATER_MARK.

In other words, if the HeartBeat thread (see below) fails to connect for 30 seconds (i.e. 10 attempts * 3 seconds, the default values for each of the above properties), it will be handled as a failure. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.8 REPLICATION_HBT_DETECT_TIME

2.8.8.1 Data Type

Unsigned integer

2.8.8.2 Default Value

3

2.8.8.3 Attributes

Read-Write, Single Value

2.8.8.4 Range

[0, $2^{32} - 1$]

2.8.8.5 Description

This property specifies the interval, in seconds, at which to check the HeartBeat thread[1]. The HeartBeat thread checks the host for a fault every 3 seconds (the default value). This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.9 REPLICATION_INSERT_REPLACE

2.8.9.1 Data Type

Unsigned integer

2.8.9.2 Default Value

0

2.8.9.3 Attributes

Read-Write, Single Value

2.8.9.4 Range

[0, 1]

2.8.9.5 Description

This property specifies whether to keep inserted contents if an insert conflict occurs during replication. If this value has been set to 0, the insert will not be committed, and the data conflict will be handled as an error, whereas if this value has been set to 1, the data conflict will be ignored and the insert will be committed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.10 REPLICATION_KEEP_ALIVE_CNT

2.8.10.1 Data Type

Unsigned integer

2.8.10.2 Default Value

600

1. HeartBeat thread: In a replication environment of ALTIBASE HDB, in order to allow physical faults to be detected as quickly as possible while data are being exchanged between a Sender thread and a Receiver thread, a HeartBeat Thread is used to allow each host to regularly check the condition of the other host.

2.8.10.3 Attributes

Read-Only, Single Value

2.8.10.4 Range

[0, 2^{32}-1]

2.8.10.5 Description

A KEEP_ALIVE signal is sent when a Sender thread has not sent a packet and has slept for REPLICATION_SENDER_SLEEP_TIME * REPLICATION_KEEP_ALIVE_CNT.

2.8.11 REPLICATION_LOG_BUFFER_SIZE

2.8.11.1 Data Type

Unsigned Integer

2.8.11.2 Default Value

30 (MB)

2.8.11.3 Attributes

Read-Only, Single Value

2.8.11.4 Range

[0, 212-1]

2.8.11.5 Description

This property is set in order to improve replication performance using a dedicated replication log buffer. The dedicated replication log buffer filters and stores only replication logs.

The Sender thread can read logs from the log buffer or from disk. However, when reading logs from disk, the processing speed of the Sender thread may be greatly reduced. Furthermore, the additional burden of reading unnecessary logs is imposed. The dedicated replication log buffer mitigates this burden.

However, when there is more than one Log File Group (LFG), the dedicated replication log buffer cannot be used, and the value of this property is ignored.

When multiple replication Sender threads are working, replication and overall service performance can suffer. This is because there is only one replication log buffer, so if it is accessed by more than one Sender thread, synchronization overhead is more likely to occur.

When the REPLICATION_SYNC_LOG value is set to 1, this property must be set to 0. Otherwise, the

ALTIBASE HDB server will fail to start. If the value of this property is set too small, it may lead to worse performance than when it is not used at all (i.e. when it is set to 0).

2.8.12 REPLICATION_MAX_LISTEN

2.8.12.1 Data Type

Unsigned integer

2.8.12.2 Default Value

32

2.8.12.3 Attributes

Read-Only, Single Value

2.8.12.4 Range

[0, 512]

2.8.12.5 Description

This property specifies the maximum size of the "listen queue" when TCP/IP is used for communication between a Sender thread and an ALTIBASE HDB server that maintains a Receiver thread.

2.8.13 REPLICATION_MAX_LOGFILE

2.8.13.1 Data Type

Unsigned Integer

2.8.13.2 Default Value

0

2.8.13.3 Attributes

Read-Write, Single Value

2.8.13.4 Range

[0, 65535]

2.8.13.5 Description

This property specifies the maximum number of log files preceding the Restart Redo Point that are to be prevented from being deleted, for use in replication.

If, after replication starts, changes to a local server are not also made on a remote server for some reason, such as reduced network speed between the local and remote servers, replication will prevent log files from being deleted, even after checkpointing has taken place. Under such circumstances, the number of log files on the local server will continue to increase, which can ultimately lead to a disk full error.

Therefore, when checkpointing occurs, if the number of accumulated log files preceding the Restart Redo Point exceeds the number specified using this property, the replication Restart SN is set to the highest SN in the current log file, and the log files preceding the Restart Redo Point are deleted. Then, replication is performed starting from the new Restart SN.

If this property is set to 0, or if replication is running in EAGER mode, this function is disabled. Please note that because log files are erased when checkpointing is carried out, the values of the CHECKPOINT_INTERVAL_IN_SEC and CHECKPOINT_IN_LOG properties should be considered when setting the value of this property.

2.8.14 REPLICATION_NET_CONN_IP_STACK

2.8.14.1 Data Type

Unsigned Integer

2.8.14.2 Default Value

The default value for this property is the same as the value set for the NET_CONN_IP_STACK property.

2.8.14.3 Attributes

Read-Only, Single Value

2.8.14.4 Range

[0, 1, 2]

2.8.14.5 Description

This property specifies the Internet Protocol Stack to be used when creating sockets on the Replication Receiver side for communication between the Receiver and the Sender via TCP/IP.

0: An Internet Protocol Stack supporting only IPv4 will be used.

1: A dual stack (Internet Protocol Stack supporting both IPv4 and IPv6) will be used.

2: An Internet Protocol Stack supporting only IPv6 will be used.

2.8.15 REPLICATION_POOL_ELEMENT_COUNT

2.8.15.1 Data Type

Unsigned Integer

2.8.15.2 Default Value

10

2.8.15.3 Attributes

Read-Write, Single Value

2.8.15.4 Range

[1, 1024]

2.8.15.5 Description

This is the amount of memory (number of elements) used when a Sender thread analyzes a log and copies column values. Memory elements are pre-allocated from the memory pool, and their size is specified by REPLICATION_POOL_ELEMENT_SIZE. The value of this property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.16 REPLICATION_POOL_ELEMENT_SIZE

2.8.16.1 Data Type

Unsigned Integer

2.8.16.2 Default Value

256

2.8.16.3 Attributes

Read-Write, Single Value

2.8.16.4 Range

[128, 65536]

2.8.16.5 Description

This is the size of a memory element, in bytes, that is used when the sender thread analyzes a log and copies column values. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.17 REPLICATION_PORT_NO

2.8.17.1 Data Type

Unsigned integer

2.8.17.2 Default Value

0

2.8.17.3 Attributes

Read-Only, Single Value

2.8.17.4 Range

[0, 65535]

2.8.17.5 Description

This property specifies the replication port number on the local server, to be used when a replication connection is established. Set this property to 0 to disable replication.

2.8.18 REPLICATION_PREFETCH_LOGFILE_COUNT

2.8.18.1 Data Type

Unsigned integer

2.8.18.2 Default Value

0

2.8.18.3 Attributes

Read-Write, Single Value

2.8.18.4 Range

[0, 2^{32} – 1]

2.8.18.5 Description

This property specifies the number of prefetch log files, that is, the number of log files in each log file group that are read in advance. Pre-reading and caching log files allows the Sender thread to read logs from log files more quickly.

2.8.19 REPLICATION_RECOVERY_MAX_LOGFILE

2.8.19.1 Data Type

Unsigned Integer

2.8.19.2 Default Value

0

2.8.19.3 Attributes

Read-Write, Single Value

2.8.19.4 Range

[0, 65535]

2.8.19.5 Description

This property specifies the maximum number of log files that are not deleted, based on a Restart Redo Point, for data recovery using replication.

In order to recover data at the time of replication, the local server does not delete logs that have not been flushed to disk on remote servers. Even if checkpointing takes place at this time, because the log files cannot be deleted, the number of log files on the local server will continue to increase, which can ultimately lead to a disk full error.

Thus, if the maximum log file count in the recovery options is exceeded when checkpointing occurs, replication-based recovery is aborted and the log files are deleted. Then, replication starts over.

If this property is set to 0 or replication runs in eager mode, this function is not used. Because log files are deleted when checkpointing occurs, the values of CHECKPOINT_INTERVAL_IN_SEC and CHECKPOINT_IN_LOG should be considered together.

2.8.20 REPLICATION_RECOVERY_MAX_TIME

2.8.20.1 Data Type

Unsigned Integer

2.8.20.2 Default Value

$2^{32} - 1$ (seconds)

2.8.20.3 Attributes

Read-Only, Single Value

2.8.20.4 Range

$[0, 2^{32} - 1]$

2.8.20.5 Description

If the number of seconds specified using this property is exceeded while the replication module is performing recovery, recovery is stopped and service is provided in the state in which recovery has been performed up to that point.

If this property is set to 0, replication-based recovery is not performed. Before replication-based data recovery is completed, ALTIBASE HDB will not be able to proceed to the service stage, and service may be delayed.

2.8.21 REPLICATION_SENDER_AUTO_START

2.8.21.1 Data Type

Unsigned integer

2.8.21.2 Default Value

1

2.8.21.3 Attributes

Read-Only, Single Value

2.8.21.4 Range

[0, 1]

2.8.21.5 Description

If a replication Sender thread is still active when the server is restarted, ALTIBASE HDB automatically restarts the thread. If this value is set to 0, the user can prevent the Sender thread from being restarted.

2.8.22 REPLICATION_SENDER_SLEEP_TIME

2.8.22.1 Data Type

Unsigned Integer

2.8.22.2 Default Value

10000

2.8.22.3 Attributes

Read-Only, Single Value

2.8.22.4 Range

[0, 2^{32} – 1]

2.8.22.5 Description

This property indicates the sleep time, in microseconds, when there are no more logs to be read by the Sender thread. Because certain platforms ignore short Sleep time values, a suitable value must be specified. The value specified here is used in conjunction with REPLICATION_KEEP_ALIVE_CNT to determine when to send KEEP_ALIVE.

2.8.23 REPLICATION_SERVER_FAILBACK_MAX_TIME

2.8.23.1 Data Type

Unsigned integer

2.8.23.2 Default Value

2^{32}-1

2.8.23.3 Attributes

Read-Only, Single Value

2.8.23.4 Range

[0, 2^{32}-1]

2.8.23.5 Description

In EAGER mode replication, when a server that was terminated abnormally is restarted, it resumes providing service only after it has synchronized its data with the data on another (i.e. the remote) server. At this time, if the process of applying the logs from the other server on the server that experienced the fault takes longer than the number of seconds specified using this property, the server that experienced the fault gives up waiting for synchronization to complete.

2.8.24 REPLICATION_SYNC_LOG

2.8.24.1 Data Type

Unsigned Integer

2.8.24.2 Default Value

0

2.8.24.3 Attributes

Read-Only, Single Value

2.8.24.4 Range

[0, 1]

2.8.24.5 Description

When performing replication, because the Sender thread sends logs that are in memory regardless of whether they have been committed to disk, data inconsistency or other problems may occur in the event of system or media failure.

To prevent this problem, setting this value to 1 ensures that the Sender thread only sends logs that have already been committed to disk.

2.8.25 REPLICATION_SYNC_TUPLE_COUNT

2.8.25.1 Data Type

Unsigned long

2.8.25.2 Default Value

30000

2.8.25.3 Attributes

Read-Write, Single Value

2.8.25.4 Range

[0, $2^{64} - 1$]

2.8.25.5 Description

This property specifies the maximum number of records that each Sender thread can read and handle during parallel synchronization. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.26 REPLICATION_TIMESTAMP_RESOLUTION

2.8.26.1 Data Type

Unsigned integer

2.8.26.2 Default Value

0

2.8.26.3 Attributes

Read-Write, Single Value

2.8.26.4 Range

[0, 1]

2.8.26.5 Description

In an Active-Active replication environment, if this property is set to 1 and a TIMESTAMP column exists in a given replication target table, then the TIMESTAMP-based resolution scheme is used to resolve any data conflicts that occur in that table. However, even if a TIMESTAMP column exists in a replication target table, if this value has been set to 0, some other conflict resolution scheme is used.

For more about TIMESTAMP-based resolution and data conflicts, please refer to the *ALTIBASE HDB Replication Manual*. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.8.27 REPLICATION_UPDATE_REPLACE

2.8.27.1 Data Type

Unsigned integer

2.8.27.2 Default Value

0

2.8.27.3 Attributes

Read-Write, Single Value

2.8.27.4 Range

[0, 1]

2.8.27.5 Description

This property specifies whether to keep updated contents if an update conflict occurs during replication.

If this value has been set to 0, the update will not be committed, and the data conflict will be handled as an error, whereas if this value has been set to 1, the data conflict will be ignored and the update will be committed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.9 Message Logging Properties

2.9.1 ALL_MSGLOG_FLUSH

2.9.1.1 Data Type

Unsigned integer

2.9.1.2 Default Value

1

2.9.1.3 Attributes

Read-Write, Single Value

2.9.1.4 Range

[0, 1]

2.9.1.5 Description

If this property is set to 1, all database messages are written immediately to disk, whereas if it is set to 0, ALTIBASE HDB writes the messages all at once at regularly scheduled intervals. In order to prevent reduced performance attributable to excessive logging, it is recommended that this value be set to 0 for normal operations, and that it be set to 1 when troubleshooting.

2.9.2 DL_MSGLOG_COUNT

2.9.2.1 Data Type

Unsigned Integer

2.9.2.2 Default Value

10

2.9.2.3 Attributes

Read-Only, Single Value

2.9.2.4 Range

[0, 2^{32} – 1]

2.9.2.5 Description

This sets the maximum number of Database Link message files.

2.9.3 DL_MSGLOG_DIR

2.9.3.1 Data Type

String

2.9.3.2 Default Value

$ALTIBASE_HOME/trc

2.9.3.3 Attributes

Read-Only, Single Value

2.9.3.4 Range

None

2.9.3.5 Description

This property sets the directory in which the Database Link module maintains message files.

2.9.4 DL_MSGLOG_FILE

2.9.4.1 Data Type

String

2.9.4.2 Default Value

altibase_dl.log

2.9.4.3 Attributes

Read-Only, Single Value

2.9.4.4 Range

None

2.9.4.5 Description

This property specifies the file in which to write messages that arise during Database Link processing.

2.9.5 DL_MSGLOG_FLAG

2.9.5.1 Data Type

Unsigned Integer

2.9.5.2 Default

6

2.9.5.3 Attributes

Read-Write, Single Value

2.9.5.4 Range

[0, 2^{32} – 1]

2.9.5.5 Description

This is a flag value that indicates whether to write trace messages generated by the database Link module to DL_MSGLOG_FILE.

If this property is set to 0, no messages are written, whereas if it is set to a value greater than 0, the messages are written.

2.9.6 DL_MSGLOG_SIZE

2.9.6.1 Data Type

Unsigned Integer

2.9.6.2 Default Value

10 * 1024 * 1024

2.9.6.3 Attributes

Read-Only, Single Value

2.9.6.4 Range

[0, $2^{32} - 1$]

2.9.6.5 Description

This property sets the maximum size of Database Link message files.

2.9.7 LK_MSGLOG_COUNT

2.9.7.1 Data Type

Unsigned Integer

2.9.7.2 Default

10

2.9.7.3 Attributes

Read-Only, Single Value

2.9.7.4 Range

[0, $2^{32} - 1$]

2.9.7.5 Description

This property sets the maximum number of message files that the Database Link connecting processing module can create.

2.9.8 LK_MSGLOG_DIR

2.9.8.1 Data Type

String

2.9.8.2 Default

$ALTIBASE_HOME/trc

2.9.8.3 Attributes

Read-Only, Single Value

2.9.8.4 Range

None

2.9.8.5 Description

This property specifies the directory in which the Database Link connection processing module stores message files.

2.9.9 LK_MSGLOG_FILE

2.9.9.1 Data Type

String

2.9.9.2 Default

altibase_lk.log

2.9.9.3 Attributes

Read-Only, Single Value

2.9.9.4 Range

None

2.9.9.5 Description

This property specifies the prefix of the file(s) in which the Database Link connection processing module writes messages.

2.9.10 LK_MSGLOG_FLAG

2.9.10.1 Data Type

Unsigned Integer

2.9.10.2 Default

6

2.9.10.3 Attributes

Read-Write, Single Value

2.9.10.4 Range

[0, $2^{32} - 1$]

2.9.10.5 Description

This is a flag value that indicates whether to write messages generated by the Database Link connection processing module to LK_MSGLOG_FILE.

If this property is set to 0, no messages are written, whereas if it is set to a value greater than 0, the messages are written.

2.9.11 LK_MSGLOG_SIZE

2.9.11.1 Data Type

Unsigned Integer

2.9.11.2 Default

10 * 1024 * 1024

2.9.11.3 Attributes

Read-Only, Single Value

2.9.11.4 Range

[0, $2^{32} - 1$]

2.9.11.5 Description

This property sets the maximum size of the Database Link connection processing module message files.

2.9.12 MM_MSGLOG_COUNT

2.9.12.1 Data Type

Unsigned Integer

2.9.12.2 Default Value

10

2.9.12.3 Attributes

Read-Only, Single Value

2.9.12.4 Range

[0, $2^{32} - 1$]

2.9.12.5 Description

This sets the maximum number of message files for the Main module.

2.9.13 MM_MSGLOG_DIR

2.9.13.1 Data Type

String

2.9.13.2 Default Value

$ALTIBASE_HOME/trc

2.9.13.3 Attributes

Read-Only, Single Value

2.9.13.4 Range

None

2.9.13.5 Description

This property sets the directory in which the Main module maintains message files.

2.9.14 MM_MSGLOG_FILE

2.9.14.1 Data Type

String

2.9.14.2 Default Value

altibase_mm.log

2.9.14.3 Attributes

Read-Only, Single Value

2.9.14.4 Range

None

2.9.14.5 Description

This property specifies the file in which to write messages that arise during Main module processing.

2.9.15 MM_SESSION_LOGGING

2.9.15.1 Data Type

Unsigned Integer

2.9.15.2 Default

0

2.9.15.3 Attributes

Read-Write, Single Value

2.9.15.4 Range

[0, 1]

2.9.15.5 Description

This is a flag value that indicates whether to write session information regarding all database logon and logoff events to MM_MSGLOG_FILE. Session information includes session ID, user name, IP address, client program PID and other details about the client program.

If this property is set to 0, no messages are written, whereas if it is set to 1, the messages are written.

2.9.16 MM_MSGLOG_SIZE

2.9.16.1 Data Type

Unsigned Integer

2.9.16.2 Default Value

10 * 1024 * 1024

2.9.16.3 Attributes

Read-Only, Single Value

2.9.16.4 Range

[0, 2^{32} – 1]

2.9.16.5 Description

This property sets the maximum size of the Main module message files.

2.9.17 NETWORK_ERROR_LOG

2.9.17.1 Data Type

Unsigned Integer

2.9.17.2 Default

1

2.9.17.3 Attributes

Read-Write, Single Value

2.9.17.4 Range

[0, 1]

2.9.17.5 Description

This property specifies whether to write network-related error messages in the server message file.

In an unstable network environment, in which error messages are frequently output, setting this value to 0 prevents network-related error messages from being output.

2.9.18 QP_MSGLOG_COUNT

2.9.18.1 Data Type

Unsigned integer

2.9.18.2 Default Value

10

2.9.18.3 Attributes

Read-Only, Single Value

2.9.18.4 Range

[0, 2^{32} – 1]

2.9.18.5 Description

This property sets the maximum number of message log files for the Query Processor.

2.9.19 QP_MSGLOG_DIR

2.9.19.1 Data Type

String

2.9.19.2 Default Value

$ALTIBASE_HOME/trc

2.9.19.3 Attributes

Read-Only, Single Value

2.9.19.4 Range

None

2.9.19.5 Description

This property specifies the directory name in which the Query Processor writes message log files.

2.9.20 QP_MSGLOG_FILE

2.9.20.1 Data Type

String

2.9.20.2 Default Value

altibase_qp.log

2.9.20.3 Attributes

Read-Only, Single Value

2.9.20.4 Range

None

2.9.20.5 Description

This property specifies the name of the file in which to write messages when processing queries.

2.9.21 QP_MSGLOG_FLAG

2.9.21.1 Data Type

Unsigned Integer

2.9.21.2 Default Value

0

2.9.21.3 Attributes

Read-Write, Single Value

2.9.21.4 Range

[0, 2^{32} – 1]

2.9.21.5 Description

This is a flag value that indicates whether to write trace messages generated by the Query Processor in QP_MSGLOG_FILE.

If this property is set to 0, the messages are not written, whereas if it is set to a value greater than 0, the messages are written.

2.9.22 QP_MSGLOG_SIZE

2.9.22.1 Data Type

Unsigned integer

2.9.22.2 Default Value

10 * 1024 * 1024

2.9.22.3 Attributes

Read-Only, Single Value

2.9.22.4 Range

[0, $2^{32} - 1$]

2.9.22.5 Description

This property specifies the maximum size, in bytes, of the Query Processor message log files.

2.9.23 QUERY_PROF_FLAG

2.9.23.1 Data Type

Integer

2.9.23.2 Default Value

0

2.9.23.3 Attributes

Read-Write, Single Value

2.9.23.4 Range

[0, $2^{6} - 1$]

2.9.23.5 Description

This property enables information about the work being conducted by a server and the overall state of the server to be written to a file for later analysis. The user can specify that information is written as desired by suitably combining the following values:

0: write nothing

1: every time a SQL statement is executed, write the executed SQL statement, execution time, execution information, and information about index and disk access

2: every time a SQL statement is executed, write the BIND parameter(s)

4: every time a SQL statement is executed, write the execution plan

8: write session information (i.e. the data in V$SESSTAT) every 3 seconds

16: write system information (i.e. the data in V$SYSSTAT) every 3 seconds

32: write information about memory (i.e. the data in V$MEMSTAT) every 3 seconds

For example, if this property is set to 1+4+32=37, then whenever a SQL statement is executed, the execution information and execution plan for the SQL statement is written, and additionally, information about memory is written every 3 seconds.

This file can be converted to a form suitable for analysis using the `altiprofile` utility. For more information, please refer to the portion of the *Utilities Manual* pertaining to the `altiprofile` utility. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.9.24 RP_MSGLOG_COUNT

2.9.24.1 Data Type

Unsigned integer

2.9.24.2 Default Value

10

2.9.24.3 Attributes

Read-Only, Single Value

2.9.24.4 Range

[0, $2^{32} - 1$]

2.9.24.5 Description

This property specifies the maximum number of replication message log files.

2.9.25 RP_MSGLOG_DIR

2.9.25.1 Data Type

String

2.9.25.2 Default Value

$ALTIBASE_HOME/trc

2.9.25.3 Attributes

Read-Only, Single Value

2.9.25.4 Range

None

2.9.25.5 Description

This property specifies the directory name in which the replication module writes message log files.

2.9.26 RP_MSGLOG_FILE

2.9.26.1 Data Type

String

2.9.26.2 Default Value

altibase_rp.log

2.9.26.3 Attributes

Read-Only, Single Value

2.9.26.4 Range

None

2.9.26.5 Description

This property specifies the name of the file in which to write messages output from the Replication Manager.

2.9.27 RP_MSGLOG_FLAG

2.9.27.1 Data Type

Unsigned Integer

2.9.27.2 Default Value

2

2.9.27.3 Attributes

Read-Write, Single Value

2.9.27.4 Range

[0, 2^{32} – 1]

2.9.27.5 Description

This is a flag value that indicates whether to write trace messages generated by the Replication Manager module in RP_MSGLOG_FILE.

If this property is set to 0, no messages are written, whereas if it is set to a value greater than 0, the messages are written.

2.9.28 RP_MSGLOG_SIZE

2.9.28.1 Data Type

Unsigned integer

2.9.28.2 Default Value

10 * 1024 * 1024

2.9.28.3 Attributes

Read-Only, Single Value

2.9.28.4 Range

[0, 2^{32} – 1]

2.9.28.5 Description

This property specifies the maximum size, in bytes, of the replication message log file.

2.9.29 SERVER_MSGLOG_COUNT

2.9.29.1 Data Type

Unsigned integer

2.9.29.2 Default Value

10

2.9.29.3 Attributes

Read-Only, Single Value

2.9.29.4 Range

[0, $2^{32} - 1$]

2.9.29.5 Description

This property specifies the maximum number of server message log files.

2.9.30 SERVER_MSGLOG_DIR

2.9.30.1 Data Type

String

2.9.30.2 Default Value

$ALTIBASE_HOME/trc

2.9.30.3 Attributes

Read-Only, Single Value

2.9.30.4 Range

None

2.9.30.5 Description

This property specifies the path in which altibase.lock, which is an internally used server maintenance file, and SERVER_MSGLOG_FILE, which is the server module message file in which information about the server startup, shutdown etc. are written, are located.

This directory can also serve as the default directory for individual modules when default values have not been individually set for their corresponding properties, such as SM_MSGLOG_DIR, QP_MSGLOG_DIR, RP_MSGLOG_DIR and the like.

2.9.31 SERVER_MSGLOG_FILE

2.9.31.1 Data Type

String

2.9.31.2 Default Value

altibase_boot.log

2.9.31.3 Attributes

Read-Only, Single Value

2.9.31.4 Range

None

2.9.31.5 Description

This property specifies the file name for messages left by the server module. Messages pertaining to ALTIBASE HDB startup, warnings, and abnormal termination are written to the server message log file.

2.9.32 SERVER_MSGLOG_FLAG

2.9.32.1 Data Type

Unsigned Integer

2.9.32.2 Default Value

7

2.9.32.3 Attributes

Read-Write, Single Value

2.9.32.4 Range

[0, $2^{32} - 1$]

2.9.32.5 Description

This is a flag value that indicates whether to write trace messages generated by the server module in SERVER_MSGLOG_FILE.

If this property is set to 0, no messages are written, whereas if it is set to a value greater than 0, the messages are written.

2.9.33 SERVER_MSGLOG_SIZE

2.9.33.1 Data Type

Unsigned integer

2.9.33.2 Default Value

10 * 1024 * 1024

2.9.33.3 Attributes

Read-Only, Single Value

2.9.33.4 Range

[0, $2^{32} - 1$]

2.9.33.5 Description

This property specifies the maximum size, in bytes, of server message log files.

2.9.34 SM_MSGLOG_COUNT

2.9.34.1 Data Type

Unsigned integer

2.9.34.2 Default Value

10

2.9.34.3 Attributes

Read-Only, Single Value

2.9.34.4 Range

[0, $2^{32} – 1$]

2.9.34.5 Description

This property specifies the maximum number of Storage Manager message log files.

2.9.35 SM_MSGLOG_DIR

2.9.35.1 Data Type

String

2.9.35.2 Default Value

$ALTIBASE_HOME/trc

2.9.35.3 Attributes

Read-Only, Single Value

2.9.35.4 Range

None

2.9.35.5 Description

This property specifies the directory name in which to write the Storage Manager message log files.

2.9.36 SM_MSGLOG_FILE

2.9.36.1 Data Type

String

2.9.36.2 Default Value

altibase_sm.log

2.9.36.3 Attributes

Read-Only, Single Value

2.9.36.4 Range

None

2.9.36.5 Description

This property specifies the prefix of the name of the message file(s) in which the Storage Manager writes messages.

2.9.37 SM_MSGLOG_FLAG

2.9.37.1 Data Type

Unsigned Integer

2.9.37.2 Default Value

2147483647

2.9.37.3 Attributes

Read-Write, Single Value

2.9.37.4 Range

[0, 2^{32} – 1]

2.9.37.5 Description

This is a flag value that indicates whether to write trace messages generated by the Storage Manager module in the file(s) specified in SM_MSGLOG_FILE.

If this property is set to 0, no messages are written, whereas if it is set to a value greater than 0, the messages are written.

2.9.38 SM_MSGLOG_SIZE

2.9.38.1 Data Type

Unsigned integer

2.9.38.2 Default Value

10 * 1024 * 1024

2.9.38.3 Attributes

Read-Only, Single Value

2.9.38.4 Range

[0, 2^{32} – 1]

2.9.38.5 Description

This property specifies the maximum size, in bytes, of the Storage Manager message log files.

2.9.39 TRCLOG_DETAIL_PREDICATE

2.9.39.1 Data Type

Unsigned integer

2.9.39.2 Default Value

0

2.9.39.3 Attributes

Read-Write, Single Value

2.9.39.4 Range

[0, 1]

2.9.39.5 Description

When Explain Plan mode is being used in iSQL, this property specifies whether to display the status of a predicate portion of a WHERE clause. To use this trace log, set this parameter to 1. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.9.40 XA_MSGLOG_COUNT

2.9.40.1 Data Type

Unsigned Integer

2.9.40.2 Default Value

10

2.9.40.3 Attributes

Read-Only, Single Value

2.9.40.4 Range

[0, 2^{32}–1]

2.9.40.5 Description

This property specifies the maximum number of XA message files used by the server.

2.9.41 XA_MSGLOG_DIR

2.9.41.1 Data Type

String

2.9.41.2 Default Value

$ALTIBASE_HOME/trc

2.9.41.3 Attributes

Read-Only, Single Value

2.9.41.4 Range

None

2.9.41.5 Description

This property specifies the directory in which XA message files used by the server are stored.

2.9.42 XA_MSGLOG_FILE

2.9.42.1 Data Type

String

2.9.42.2 Default Value

altibase_xa.log

2.9.42.3 Attributes

Read-Only, Single Value

2.9.42.4 Range

None

2.9.42.5 Description

This property specifies the prefix of the name of the file(s) in which XA message logs from the server are written.

2.9.43 XA_MSGLOG_FLAG

2.9.43.1 Data Type

Unsigned Integer

2.9.43.2 Default Value

3

2.9.43.3 Attributes

Read-Write, Single Value

2.9.43.4 Range

[0, 3]

2.9.43.5 Description

This property determines which of the server XA messages to write to disk. The possible values are as follows:

0: write only critical XA-related messages

1: write messages pertaining to XA calls

2: write messages when XIDs are allocated, freed, etc.

3: write all message logs related to XA

2.9.44 XA_MSGLOG_SIZE

2.9.44.1 Data Type

Unsigned Integer

2.9.44.2 Default Value

10 * 1024 * 1024

2.9.44.3 Attributes

Read-Only, Single Value

2.9.44.4 Range

[0, 2^{32}–1]

2.9.44.5 Description

This property specifies the maximum size of XA message files used by the server.

2.10 Database Link Related Properties

2.10.1 AUTO_REMOTE_EXEC

2.10.1.1 Data Type

Unsigned Integer

2.10.1.2 Default Value

0

2.10.1.3 Attributes

Read-Write, Single Value

2.10.1.4 Range

[0, 1]

2.10.1.5 Description

When using Database Link, this property specifies that only results of search targets are to be retrieved from a remote server, even if EXEC_REMOTE hints are not used directly in SQL statements.

0 : Default Action

1 : Forward queries to a remote server. (REMOTE hint option)

The value of this property can be changed using the ALTER SYSTEM or ALTER SESSION statements while ALTIBASE HDB is running.

2.10.2 DBLINK_ENABLE

2.10.2.1 Data Type

Unsigned Integer

2.10.2.2 Default Value

0

2.10.2.3 Attributes

Read-Only, Single Value

2.10.2.4 Range

[0, 1]

2.10.2.5 Description

This property determines whether to use Database Link. Set this value to 1 to use Database Link. If this value is set to 0 (zero), Database Link cannot be used.

2.10.3 LINKER_LINK_TYPE

2.10.3.1 Data Type

Unsigned Integer

2.10.3.2 Default Value

0

2.10.3.3 Attributes

Read-Only, Single Value

2.10.3.4 Range

[0, 2]

2.10.3.5 Description

This property determines the method of communication between an ALTIBASE HDB server and AltiLinker. If the value of this property is set to 0, communication is conducted using TCP. If it is set to 1, communication is conducted using the UNIX domain protocol. If it is set to 2, communication is conducted using IPC. (At present, only TCP and the Unix domain protocol are supported.)

2.10.4 LINKER_PORT_NO

2.10.4.1 Data Type

Unsigned Integer

2.10.4.2 Default Value

0

2.10.4.3 Attributes

Read-Only, Single Value

2.10.4.4 Range

[0, 65535]

2.10.4.5 Description

When TCP is used for communication with AltiLinker, this property specifies the port number at which AltiLinker listens.

2.10.5 LINKER_SQLLEN_SIZE

2.10.5.1 Data Type

Unsigned Integer

2.10.5.2 Default Value

0

2.10.5.3 Attributes

Read-Only, Single Value

2.10.5.4 Range

[0, 2^{32}-1]

2.10.5.5 Description

This property specifies the size of SQLLEN, used by UNIXODBC, in units of bytes or bits. If this property is set to 4 or 32, the size of SQLLEN is specified as 4 bytes, or 32 bits. If this property is set to 8 or 64, the size of SQLLEN is specified as 64 bits. If you are not sure how to specify this property, you should set it to sizeof(SQLLEN).

2.10.6 LINKER_THREAD_COUNT

2.10.6.1 Data Type

Unsigned Integer

2.10.6.2 Default Value

16

2.10.6.3 Attributes

Read-Only, Single Value

2.10.6.4 Range

[0, 100]

2.10.6.5 Description

This property specifies the number of Linker threads that are launched by AltiLinker.

2.10.7 LINKER_THREAD_SLEEP_TIME

2.10.7.1 Data Type

Unsigned Integer

2.10.7.2 Default Value

200 (1000 on Windows platforms)

2.10.7.3 Attributes

Read-Only, Single Value

2.10.7.4 Range

[0, 2^{32} – 1]

2.10.7.5 Description

This property specifies the wait time, in microseconds, when there are no tasks to be processed by the Linker thread. For normal system operation, the default on Unix platforms is 200, but on Windows platforms the default value is 1000.

2.10.8 MAX_DBLINK_COUNT

2.10.8.1 Data Type

Unsigned Integer

2.10.8.2 Default Value

10

2.10.8.3 Attributes

Read-Only, Single Value

2.10.8.4 Range

[0, $2^{32} - 1$]

2.10.8.5 Description

This property specifies the number of caches that will be used by Database Link. It has nothing to do with the number of instances of Database Link that can be created. More instances of Database Link can be created than the number specified here.

However, if the number of Database Link instances is greater than the number of caches, this may cause frequent cache changes, resulting in reduced performance.

2.11 DataPort Properties

2.11.1 DATAPORT_FILE_DIRECTORY

2.11.1.1 Data Type

String

2.11.1.2 Default Value

$ALTIBASE_HOME/dbs

2.11.1.3 Attributes

Read-Write, Single Value

2.11.1.4 Range

None

2.11.1.5 Description

This property specifies the default directory in which the dataport files are located. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.11.2 DATAPORT_IMPORT_COMMIT_UNIT

2.11.2.1 Data Type

Signed Integer

2.11.2.2 Default Value

10

2.11.2.3 Attributes

Read-Write, Single Value

2.11.2.4 Range

[1, 2^{31} - 1]

2.11.2.5 Description

When importing data, this property determines how many statements are committed at one time after being executed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.11.3 DATAPORT_IMPORT_STATEMENT_UNIT

2.11.3.1 Data Type

Signed Integer

2.11.3.2 Default Value

50000

2.11.3.3 Attributes

Read-Write, Single Value

2.11.3.4 Range

[1, 2^{31} - 1]

2.11.3.5 Description

This property indicates how many rows are inserted per statement when importing data. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.12 Other Properties

2.12.1 ACCESS_LIST

2.12.1.1 Format

```
ACCESS_LIST = operation, address, mask
```

2.12.1.2 Range

- operation ::= [PERMIT|DENY]

  Indicates whether to allow or deny access by an IP packet that matches a validation rule.

- address

  Indicates the IP address of the packet to validate. It can be in IPv4 or IPv6 address notation.

- mask

  If the specified `address` is in IPv4 address notation, `mask` specifies that only part of the IP address of a packet, the subnet mask, is to be validated.

  If the specified `address` is in IPv6 address notation, `mask` gives the length of prefix bits to be compared. An IPv6 address is matched if the specified `mask` bits of the specified `address` are equal to the specified `mask` bits of the originating address of an incoming IP packet.

2.12.1.3 Validation Rule

```
IF
BITXOR ( BITAND ( IP_Packet, mask ), BITAND ( address, mask ) ) = 0
THEN valid
ELSE invalid
```

2.12.1.4 Description

Packets that attempt to access an Altibase database can be allowed or blocked based on the IP address from which they originate. The address of IP packets is checked based on a validation rule, and if the address satisfies the condition in the validation rule, the packet is allowed or blocked as specified by “operation”, whereas if it does not satisfy the validation condition, it is ignored and execution proceeds to the next item on the list.

If more than one IP packet address is specified, validation is performed in the order that they are specified. If none of the conditions are satisfied, access is granted. If more than one validation rule of a single IP address is specified, a "PERMIT" rule will take priority.

2.12.1.5 Example

Block packets with the IP address 192.168.1.55 and allow all other packets.

```
ACCESS_LIST = deny, 192.168.1.55, 255.255.255.255
```

Allow access to packets from the addresses 192.168.3.* and 219.211.253.*, and block all other packets.

```
ACCESS_LIST = permit, 192.168.3.0, 255.255.255.0
ACCESS_LIST = permit, 219.211.253.0, 255.255.255.0
ACCESS_LIST = deny ,0.0.0.0, 0.0.0.0
```

Block all Ipv4 and IPv6 address except for localhost.

```
ACCESS_LIST = deny, 0.0.0.0, 0.0.0.0
ACCESS_LIST = deny, ::1, 1
ACCESS_LIST = deny, fe80::, 1
```

2.12.2 ADMIN_MODE

2.12.2.1 Data Type

Unsigned integer

2.12.2.2 Default Value

0

2.12.2.3 Attributes

Read-Write, Single Value

2.12.2.4 Range

[0, 1]

2.12.2.5 Description

ADMIN_MODE limits the database connection to administrators only.

- 0: OFF
- 1: ON

When this property is set to 1, administrator mode is activated, and only the SYS and SYSTEM_ users can connect to the server using the SYSDBA option, and other users will be unable to establish a connection. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.12.3 CHECK_MUTEX_DURATION_TIME_ENABLE

2.12.3.1 Data Type

Unsigned Integer

2.12.3.2 Default Value

0

2.12.3.3 Attributes

Read-Write, Single Value

2.12.3.4 Range

[0, 1]

2.12.3.5 Description

This property specifies whether to check MUTEX_DURATION_TIME. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

0: disable checking

1: enable checking

2.12.4 DEFAULT_DATE_FORMAT

2.12.4.1 Data Type

String

2.12.4.2 Default Value

DD-MON-RRRR

2.12.4.3 Attributes

Read-Only, Single Value

2.12.4.4 Range

None

2.12.4.5 Description

This property sets the default format of DATE type data table columns. If not specified otherwise when SQL statements are executed, DATE type data are input or output according to this setting. This type must specify the formats in which both dates and times are saved. It is also possible to use blanks within double quotation marks, such as "DD MON RRRR".

```
Ex) DEFAULT_DATE_FORMAT = YYYY/MM/DD
```

```
iSQL> SELECT sysdate FROM dual;
SYSDATE
--------------
2000/01/01
1 row selected.
```

2.12.5 EXEC_DDL_DISABLE

2.12.5.1 Data Type

Unsigned integer

2.12.5.2 Default Value

0

2.12.5.3 Attributes

Read-Write, Single Value

2.12.5.4 Range

[0, 1]

2.12.5.5 Description

Typically, after a database is initially created, DML statements are executed much more frequently than DDL statements. Because DDL statements change existing database schema, they must be executed with caution.

The administrator can thus use this property to prevent the execution of DDL statements. When this property is set to 1, DDL statements cannot be executed while ALTIBASE HDB is running, whereas if it is set to 0, DDL statements can be executed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB is running.

2.12.6 QUERY_STACK_SIZE

2.12.6.1 Data Type

Unsigned integer

2.12.6.2 Default Value

1024

2.12.6.3 Attributes

Read-Write, Single Value

2.12.6.4 Range

[8, 65536]

2.12.6.5 Description

This property specifies the size of the stack internally used in the system to process query operations such as comparisons and other operations.

When complicated calculations or stored procedures are used, a stack overflow error may occur. In such cases, the property must be changed to a bigger value.

This parameter must be set according to the application environment. If it is set to a value higher than necessary, memory space will be wasted, so this parameter must be set carefully.

This property can be set in the altibase.properties file, and can be changed using the ALTER SYSTEM or ALTER SESSION statements.

This property can be changed using the ALTER SESSION statement as follows:

```
ALTER SESSION SET STACK SIZE = n;
```

2.12.7 REMOTE_SYSDBA_ENABLE

2.12.7.1 Data Type

Unsigned Integer

2.12.7.2 Default Value

1

2.12.7.3 Attributes

Read-Write, Single Value

2.12.7.4 Range

[0, 1]

2.12.7.5 Description

This property specifies whether the SYS user can access the database with SYSDBA privileges from a remote location. Its value can be changed using the ALTER SYSTEM statement.

0 : deny remote database access with SYSDBA privileges

1 : allow remote database access with SYSDBA privileges (default)

2.12.8 SELECT_HEADER_DISPLAY

2.12.8.1 Data Type

Unsigned integer

2.12.8.2 Default Value

1

2.12.8.3 Attributes

Read-Write, Single Value

2.12.8.4 Range

[0, 1]

2.12.8.5 Description

When the results of a SELECT query are output over iSQL, this system property determines whether only the column names are output, or whether the table names are output along with the column names. This property can be set in the altibase.properties file, and can be changed using the ALTER SYSTEM or ALTER SESSION statements. If this parameter is set to 0, the table names are displayed along with the column names when the results of SQL statements are output using iSQL.

3 The Data Dictionary

The data dictionary of ALTIBASE HDB comprises meta tables, in which information about objects is stored, and process tables, in which information about processes is stored. Process tables comprise fixed tables and performance views. This chapter describes the ALTIBASE HDB data dictionary, which is the basis of all database objects and all system information of ALTIBASE HDB.

3.1 Meta Tables

Meta tables are system-defined tables that contain all information about database objects.

This section describes the types of meta tables and their structure, and explains how to read and update the information in meta tables.

3.1.1 Structure and Function

Meta tables are defined by the system for the purpose of managing database objects. They use the same data types and store records in the same way as user-defined tables.

When ALTIBASE HDB starts up, it loads information about database objects, and when DDL statements are executed, meta tables are used to read, store, and update this information.

The owner of meta tables is the system user (user name: SYSTEM_), so normal users have limited access to meta tables.

3.1.2 Retrieving Information from Meta Tables

When a database object is created, deleted or modified using a DDL statement, the system creates, deletes, or updates records in one or more meta tables.

After a DDL statement is executed, the resultant changes to database objects can be confirmed by checking meta tables. This is accomplished using a SELECT statement, just as with a regular database table.

3.1.3 Modifying Data in Meta Tables

It is possible to use DML statements to explicitly make changes to the data in meta tables. However, only the system-defined system user (SYSTEM_) can make such changes to meta tables. Additionally, when the information in meta tables is changed, the system may become impossible to start, information about database objects may be lost, or the system may be critically damaged. Therefore, users must avoid making changes to meta tables whenever possible. When it is inevitable that a user must change meta table information, it is imperative that the database first be backed up, and it must be understood that the user is completely responsible for any damage resulting from directly making changes to meta table information.

3.1.4 Modifying Meta Table Schema

The meta table schema may be modified when a new kind of DDL statement is introduced, or when the functionality of an existing statement is changed. Depending on the characteristics of the changes to meta table schema, one of two cases may arise: either the database might need to be migrated, or the meta table schema will simply be automatically modified when ALTIBASE HDB is restarted. This should be kept in mind when upgrading ALTIBASE HDB to a newer version.

3.1.5 The Kinds of Meta Tables

This table shows the list of meta tables. Their names start with SYS_.

Meta Table Name	Description
SYS_COLUMNS_	This table contains information about columns.
SYS_COMMENTS_	This table contains information about explanatory comments.
SYS_CONSTRAINTS_	This table contains information about constraints.
SYS_CONSTRAINT_COLUMNS_	This table contains information about columns having constraints.
SYS_DATABASE_	This table contains information about the name and version of the database.
SYS_DIRECTORIES_	This table contains information about directories used by stored procedures for managing files.
SYS_DN_USERS_	This table is reserved for future use.
SYS_DUMMY_	This table is for internal use only.
SYS_ENCRYPTED_COLUMNS_	This table contains additional security information for individual columns.
SYS_GRANT_OBJECT_	This table contains information about object privileges.
SYS_GRANT_SYSTEM_	This table contains information about system privileges.
SYS_INDEX_COLUMNS_	This table contains information about index key columns.
SYS_INDEX_PARTITIONS_	This table contains information about index partitions.
SYS_INDICES_	This table contains information about indexes.
SYS_LOBS_	This table contains information about LOB columns.
SYS_PART_INDICES_	This table contains information about partitioned indexes.
SYS_PART_KEY_COLUMNS_	This table contains information about partitioning keys.
SYS_PART_LOBS_	This table contains information about LOB columns for respective partitions.
SYS_PART_TABLES_	This table contains information about partitioned tables.
SYS_PRIVILEGES_	This table contains information about privileges.
SYS_PROCEDURES_	This table contains information about stored procedures and functions.

Meta Table Name	Description
SYS_PROC_PARAS_	This table contains information about the parameters for stored procedures and functions.
SYS_PROC_PARSE_	This table contains the actual text of stored procedures and stored functions.
SYS_PROC_RELATED_	This table contains information about tables accessed by stored procedures and functions.
SYS_REPLICATIONS_	This table contains general information about replication.
SYS_REPL_HOSTS_	This table contains information about replication hosts.
SYS_REPL_ITEMS_	This table contains information about tables to be replicated.
SYS_REPL_OFFLINE_DIR_	This table contains information about the log directory related to the replication offline option.
SYS_REPL_OLD_COLUMNS_	This table contains information about columns replicated by the replication sender thread.
SYS_REPL_OLD_INDEX_COLUMNS_	This table contains information about index columns replicated by the replication sender thread.
SYS_REPL_OLD_INDICES_	This table contains information about indexes replicated by the replication sender thread.
SYS_REPL_OLD_ITEMS_	This table contains information about the tables replicated by the replication sender thread.
SYS_REPL_RECOVERY_INFOS_	This table contains information about logs used by replication for recovery of a remote server.
SYS_SECURITY_	This table contains information about the state of the security module.
SYS_SYNONYMS_	This table contains information about synonyms.
SYS_TABLES_	This table contains information about all kinds of tables.
SYS_TABLE_PARTITIONS_	This table contains information about table partitions.
SYS_TBS_USERS_	This table contains information about users' access to user-defined tablespaces.
SYS_TRIGGERS_	This table contains information about triggers.
SYS_TRIGGER_DML_TABLES_	This table contains information about tables accessed by triggers.
SYS_TRIGGER_STRINGS_	This table contains the actual text of trigger commands.

Meta Table Name	Description
SYS_TRIGGER_UPDATE_COLUMNS_	This table contains information about columns that cause triggers to fire whenever their contents are changed.
SYS_USERS_	This table contains information about users.
SYS_VIEWS_	This table contains information about views.
SYS_VIEW_PARSE_	This table contains the actual text of statements used to create views.
SYS_VIEW_RELATED_	This table contains information about objects accessed by views.
SYS_XA_HEURISTIC_TRANS_	This table contains information about global transactions.

3.1.5.1 Unsupported Meta Tables

ALTIBASE HDB provides the following GIS-related meta tables. Their names begin with STO_. They aren't used at present.

STO_COLUMNS_

STO_DATUMS_

STO_ELLIPSOIDS_

STO_GEOCCS_

STO_GEOGCS_

STO_PRIMEMS_

STO_PROJCS_

STO_PROJECTIONS_

STO_SRS_

STO_USER_COLUMNS_

3.1.6 SYS_COLUMNS_

Information about columns defined in all tables, virtual columns in all views, and virtual columns in all sequences is stored in this meta table.

Column	Data Type	Description
COLUMN_ID	INTEGER	The column identifier

Column	Data Type	Description
DATA_TYPE	INTEGER	The data type
LANG_ID	INTEGER	The language identifier
OFFSET	INTEGER	The offset of the column within the record
SIZE	INTEGER	The physical length of the column within the record
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
PRECISION	INTEGER	The specified precision of the column
SCALE	INTEGER	The specified scale of the column
COLUMN_ORDER	INTEGER	The position of the column in the table
COLUMN_NAME	VARCHAR(40)	The name of the column
IS_NULLABLE	CHAR(1)	Whether NULL is permitted. T: can be NULL F: cannot be NULL
DEFAULT_VAL	VARCHAR(4000)	The default value for the column
STORE_TYPE	CHAR(1)	The column storage type V: variable type F: fixed type L: LOB column
IN_ROW_SIZE	INTEGER	The length of data that can be saved in a fixed area when data are saved in a variable-length column in a memory table
REPL_CONDITION	INTEGER	The number of replication conditions for a column

3.1.6.1 Column Information

COLUMN_ID

This is the column identifier, which is assigned automatically by the system sequence.

DATA_TYPE

This is the data type identifier. The identifiers for each data type are as follows:

Data Type	Value
CHAR	1

Data Type	Value
VARCHAR	12
NCHAR	-8
NVARCHAR	-9
NUMERIC	2
DECIMAL	2
FLOAT	6
NUMBER	6
DOUBLE	8
REAL	7
BIGINT	-5
INTEGER	4
SMALLINT	5
DATE	9
BLOB	30
CLOB	40
BYTE	20001
NIBBLE	20002
BIT	-7
VARBIT	-100
GEOMETRY	10003

For more information about data types, please refer to Chapter1: Data Types.

LANG_ID

A column that contains the language properties for character data types (CHAR, VARCHAR).

OFFSET

This indicates the physical starting point of a column within a record. The offset and size of a column are used to calculate the physical storage size of a record.

SIZE

This is the physical storage size of the column in a record, calculated by the system based on the column type, user-defined precision, etc.

USER_ID

This corresponds to a USER_ID value in the SYS_USERS_ meta table, and identifies the owner of the table to which the column belongs.

TABLE_ID

This corresponds to a TABLE_ID value in the SYS_TABLES_ meta table, and identifies the table to which the column belongs.

PRECISION

This is the precision of the data type, and is either defined by the user or corresponds to the default value for the system. In the case of a character data type, it corresponds to the length of the character data type set by the user.

SCALE

This is the scale of the data type, and is either defined by the user or corresponds to the default value for the system. This value is not used with some data types.

COLUMN_ORDER

This is the order in which columns appear in a table.

The order in which the columns are stated in a CREATE TABLE statement determines the order in which they are created, and thus their position in the table. If a column is added using an ALTER TABLE statement, the newly created column will be the last column in the table.

COLUMN_NAME

This is the name specified when a user creates a table or adds a column to the table.

IS_NULLABLE

This indicates whether NULL values are permitted for a column.

When a column is created, the user can explicitly state whether to allow NULL values for the column. If not explicitly set by the user, NULL values are allowed by default.

DEFAULT_VAL

If no column value is specified when inserting a record, this default value is used for the column. In order to disallow NULL values, a default value must be specified by the user when creating the column. If no default value is specified, NULL values will be allowed.

STORE_TYPE

When physically storing a column, it can either be written as part of a record, or it can be saved on another page, in which case only the location of the data is stored in the record.

If the physical storage size of a column is too big, or if the size of the column varies frequently for individual records, the column can be stored on another page by using the VARIABLE option when defining the column. This option is generally used for VARCHAR types where the character strings in a column are long.

This column indicates whether the VARIABLE option is used.

IN_ROW_SIZE

This is the default IN_ROW_SIZE when data are stored in variable-length columns in memory tables. When data are inserted into a variable-length column, if the length of the data is equal to or smaller than the value specified by IN_ROW_SIZE, the data are stored in the fixed space, whereas if the data are longer than this value, they are stored in a variable space. For disk tables, this value is always 0.

For more information about variable-length columns and the IN ROW clause, please refer to Chapter1: Data Types.

REPL_CONDITION

This indicates the number of columns that are associated with replication conditions when condition clauses are used in replication.

3.1.6.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_USER_COLUMNS_

3.1.7 SYS_COMMENTS_

This meta table is for storing comments such as descriptions of user-defined tables, views and associated columns.

Column name	Type	Description
USER_NAME	VARCHAR(40)	The name of the user
TABLE_NAME	VARCHAR(40)	The name of the table
COLUMN_NAME	VARCHAR(40)	The name of the column
COMMENTS	VARCHAR(4000)	The actual comment

3.1.7.1 Column Information

USER_NAME

This is the name of the table owner. Its value corresponds to one of the USER_NAME values in the SYS_USERS_ meta table.

TABLE_NAME

This is the name of the table (or view). Its value is the same as one of the TABLE_NAME values appearing in SYS_TABLES_.

COLUMN_NAME

This is the name of a column in the table (or view). Its value is equal to a COLUMN_NAME value in the SYS_COLUMNS_ meta table.

However, if the comment pertains to an entire table (or view), the value for COLUMN_NAME will be NULL.

COMMENTS

This is the actual comment written by the user.

3.1.7.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_COLUMNS_

3.1.8 SYS_CONSTRAINTS_

This meta table contains information about table constraints.

Column	Data Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
CONSTRAINT_ID	INTEGER	The constraint identifier
CONSTRAINT_NAME	VARCHAR(40)	The name of the constraint
CONSTRAINT_TYPE	INTEGER	The type of the constraint
INDEX_ID	INTEGER	The identifier of the index used by the constraint
COLUMN_CNT	INTEGER	The number of columns that are associated with the constraint
REFERENCED_TABLE_ID	INTEGER	The identifier of a table referenced in a FOREIGN KEY constraint
REFERENCED_INDEX_ID	INTEGER	The identifier of an index referenced in a FOREIGN KEY constraint

Column	Data Type	Description
DELETE_RULE	INTEGER	Whether to perform cascade delete for a FOREIGN KEY constraint 0: Do not perform cascade delete 1: perform cascade delete
VALIDATED	CHAR(1)	Whether all data conform to the constraint

3.1.8.1 Column Information

USER_ID

This is the user identifier, and corresponds to a USER_ID in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier for the table associated with the constraint, and will correspond to a TABLE_ID value in the SYS_TABLES_ meta table.

CONSTRAINT_ID

This is a constraint identifier. It is automatically assigned by the system sequence.

CONSTRAINT_NAME

This is the name of the constraint.

CONSTRAINT_TYPE

This indicates the type of the constraint. The possible types are as follows:

- 0: FOREIGN KEY
- 1: NOT NULL
- 2: UNIQUE
- 3: PRIMARY KEY
- 4: NULL
- 5: TIMESTAMP
- 6: LOCAL UNIQUE

For additional information on each type of constraint, please refer to the description of column constraints in the explanation of the CREATE TABLE statement in the *SQL Reference*.

INDEX_ID

If an index must be created in order to define constraints such as UNIQUE or PRIMARY KEY constraints, the system creates an index internally. This is the identifier of that index, and will correspond to an INDEX_ID in the SYS_INDICES_ meta table.

COLUMN_CNT

This is the number of columns associated with the constraint. For example, for a constraint such as UNIQUE (i1, i2, i3), this value would be 3.

REFERENCED_TABLE_ID

This is the identifier of a table referenced in a FOREIGN KEY constraint (not the table for which the constraint is defined). This identifier will correspond to a TABLE_ID value in the SYS_TABLES_ meta table.

REFERENCED_INDEX_ID

This indicates a UNIQUE or PRIMARY KEY constraint that must exist in a table referenced by a FOREIGN KEY constraint. The identifier of this constraint will be the same as a CONSTRAINT_ID value in the SYS_CONSTRAINTS_ meta table.

VALIDATED

This indicates whether all data conform to the constraint.

T: Validated

F: Not Validated

3.1.8.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_INDICES_

3.1.9 SYS_CONSTRAINT_COLUMNS_

This meta table contains information about columns related to all constraints defined in user tables.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
CONSTRAINT_ID	INTEGER	The constraint identifier
CONSTRAINT_COL_ORDER	INTEGER	The position of the column in the constraint
COLUMN_ID	INTEGER	The column Identifier

3.1.9.1 Column Information

USER_ID

This is the user identifier, and corresponds to a USER_ID in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table in which the constraint is defined, and corresponds to a TABLE_ID value in the SYS_TABLES_ meta table.

CONSTRAINT_ID

This is the identifier of the constraint, and corresponds to a CONSTRAINT_ID value in the SYS_CONSTRAINTS_ meta table.

CONSTRAINT_COL_ORDER

This is the position of the column within the constraint. For example, when the constraint UNIQUE (i1,i2,i3) is created, three records are inserted into the SYS_CONSTRAINT_COLUMNS_ meta table. The position of column i1 is 1, column i2 is 2, and column i3 is 3.

COLUMN_ID

This is the identifier of the column for which the constraint is defined, and corresponds to a COLUMN_ID value in the SYS_COLUMNS_ meta table.

3.1.9.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_CONSTRAINTS_

SYS_COLUMNS_

3.1.10 SYS_DATABASE_

This is the table that contains the database name and meta table version information.

Column name	Type	Description
DB_NAME	VARCHAR(40)	The database name
OWNER_DN	VARCHAR(2048)	Reserved for future use
META_MAJOR_VER	INTEGER	The database meta table version (Main)
META_MINOR_VER	INTEGER	The database meta table version (Sub)

Column name	Type	Description
META_PATCH_VER	INTEGER	The database meta table version (Patch)

3.1.10.1 Column Information

META_MAJOR_VER

This value increases when a meta table is modified, added or removed. If the database version and the corresponding binary version of ALTIBASE HDB do not match, the database must be migrated.

META_MINOR_VER

This value increases when the contents of one or more meta tables is modified. If the version of the database does not correspond to the current version of ALTIBASE HDB, the system internally compares this value and automatically upgrades the meta tables to the newer version.

META_PATCH_VER

This indicates the meta table patch version.

3.1.11 SYS_DATABASE_LINKS_

This meta table is for storing Database Link information.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
LINK_ID	INTEGER	The Database Link identifier
LINK_OID	BIGINT	The Database Link object identifier
LINK_NAME	VARCHAR(40)	The Database Link name
USER_MODE	INTEGER	The mode in which a remote server is accessed
REMOTE_USER_ID	VARCHAR(40)	The user account for a remote database
REMOTE_USER_PWD	BYTE(40)	The user password for a remote database
LINK_METHOD	INTEGER	The link method
LINK_INFO	VARCHAR(400)	The link information

3.1.11.1 Column Information

USER_ID

This is the identifier of the user who owns the Database Link object.

LINK_ID

This is the Database Link identifier.

LINK_OID

This is the Database Link object identifier.

LINK_NAME

This is the name of the Database Link object, which is specified by the user when the Database Link object is created.

USER_MODE

This indicates the mode in which a remote server is accessed.

- 0: DEDICATED USER MODE
- 1: CURRENT USER MODE (reserved for future use)

REMOTE_USER_ID

This indicates a user account on a remote server, to be used when accessing a remote database server.

REMOTE_USER_PWD

This is the password for the user account on the remote server, to be used when accessing a remote database server. The password is encrypted using an encryption algorithm before it is stored.

LINK_METHOD

This indicates the method of connecting to a remote server.

- 0: ODBC
- 1: (reserved for future use)

LINK_INFO

This is for storing information that is needed when connecting to a remote server.

3.1.12 SYS_DATA_PORTS_

This table contains information about export and import tasks that are either underway or have been completed.

For more information about data ports, please refer to *Section 10.2 DataPort* in the *Stored Procedures Manual*.

Column name	Type	Description
NAME	VARCHAR(40)	The name of the task
USER_NAME	VARCHAR(40)	The user who initiated the task
OPERATION	VARCHAR(16)	The current operation
STATE	VARCHAR(16)	The state of the task
OWNER_NAME	VARCHAR(40)	The name of the owner of the source or target table
TABLE_NAME	VARCHAR(40)	The name of the table
OBJECT_NAME	VARCHAR(256)	The file name
DIRECTORY_NAME	VARCHAR(1024)	The name of the directory
PROCESSED_ROW_CNT	BIGINT	The number of rows that have been processed
FIRST_ROW	BIGINT	The first imported row
LAST_ROW	BIGINT	The last imported row
SPLIT	BIGINT	The number of split rows

3.1.12.1 Column Information

For additional information about each column of the table, please refer to *Section 10.2 DataPort* in the *Stored Procedures Manual*.

NAME

This is the name of the task.

USER_NAME

This is the name of the user who started the task, and corresponds to a `USER_NAME` value in `SYS_USERS_`.

OPERATION

This indicates the operation that is underway. It can be either `EXPORT` or `IMPORT`.

STATE

This indicates the current state of the task. It can be either `START` or `FINISH`.

OWNER_NAME

This is the name of the user who owns the source table or target table.

TABLE_NAME

This is the name of the target table for an import or export operation. Its value corresponds to a `TABLE_NAME` value in `SYS_TABLES_`.

OBJECT_NAME

This is the name of the file that is the target of an export or import operation.

DIRECTORY_NAME

This is the name of the directory in which the files for an export or import operation are located.

PROCESSED_ROW_CNT

This is the number of rows that have already been processed.

FIRST_ROW

This is the first row to be imported, or the first row that was imported in the case of a completed task. Its value corresponds to the value which is specified in the `firstrow` parameter when executing the `IMPORT_FROM_FILE` procedure. For more information about the IMPORT_FROM_FILE procedure, please refer to the *Stored Procedures Manual*.

LAST_ROW

This is the last row to be imported, or the last row that was imported in the case of a completed task. Its value corresponds to the value which is specified in the `lastrow` parameter when executing the `IMPORT_FROM_FILE` procedure. For more information about the IMPORT_FROM_FILE procedure, please refer to the *Stored Procedures Manual*.

SPLIT

This is the number of rows to be split, or the number of rows that have been split in the case of a completed task. Its value corresponds to the value which is specified in the `split` parameter when executing the `EXPORT_TO_FILE` procedure. For more information about the EXPORT_TO_FILE procedure, please refer to the *Stored Procedures Manual*.

3.1.13 SYS_DIRECTORIES_

This table contains information about directories that are used when files are managed using stored procedures.

Column	Data Type	Description
DIRECTORY_ID	BIGINT	The directory identifier
USER_ID	INTEGER	The user identifier
DIRECTORY_NAME	VARCHAR(40)	The directory name

Column	Data Type	Description
DIRECTORY_PATH	VARCHAR(4000)	The absolute path of the directory on the system
CREATED	DATE	The time at which the directory was created
LAST_DDL_TIME	DATE	The most recent time at which a DDL task was used to change the directory object

3.1.13.1 Column Information

DIRECTORY_ID

This is a directory identifier. It is a unique value within the system.

USER_ID

This is the user identifier of the owner of the directory.

DIRECTORY_NAME

This is the name of the directory. It is a unique value within the system.

DIRECTORY_PATH

This is the absolute path where the directory is located. This value is explicitly set by the user when executing a CREATE DIRECTORY statement.

LAST_DDL_TIME

This is the most recent time at which a DDL task was used to change the directory object.

3.1.14 SYS_ENCRYPTED_COLUMNS_

This is the meta table for managing additional security information based on the security settings for individual columns.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the table to which the column belongs
TABLE_ID	INTEGER	The identifier of the table to which the column belongs
COLUMN_ID	INTEGER	The identifier of the encrypted column
ENCRYPT_PRECISION	INTEGER	The precision of the column encryption
POLICY_NAME	VARCHAR(16)	The name of the encryption policy

Column	Data Type	Description
POLICY_CODE	VARCHAR(128)	The verification code of the encryption policy

3.1.15 SYS_GRANT_OBJECT_

This contains information about object privileges granted to a user.

Column	Data Type	Description
GRANTOR_ID	INTEGER	The identifier of the user who granted the privileges
GRANTEE_ID	INTEGER	The identifier of the user to whom the privileges were granted
PRIV_ID	INTEGER	The privilege identifier
USER_ID	INTEGER	The identifier of the owner of the object
OBJ_ID	INTEGER	The identifier of the object
OBJ_TYPE	CHAR(1)	The type of object
WITH_GRANT_OPTION	INTEGER	Indicates whether the WITH_GRANT_OPTION is used when object access privileges are granted 0: Not used 1: Used

3.1.15.1 Column Information

GRANTOR_ID

This is the identifier of the user who granted the privilege, and corresponds to a USER_ID in the SYS_USERS_ meta table.

GRANTEE_ID

This is the identifier of the user to whom the privilege has been granted, and corresponds to a USER_ID in the SYS_USERS_ meta table.

PRIV_ID

This is the identifier of the privilege. It corresponds to a PRIV_ID in the SYS_PRIVILEGES_ meta table.

USER_ID

This is the user ID of the owner of the object for which the privilege has been granted. This value will correspond to a USER_ID in the SYS_USERS_ meta table.

OBJ_ID

This is the ID of the object for which the privilege has been granted. It corresponds with one, and only one, target object ID saved in the appropriate meta table.

If the target object is a table, view or sequence, it is mapped to a TABLE_ID in the SYS_TABLES_ meta table, whereas if it is a stored procedure or stored function, it is mapped to a PROC_OID in the SYS_PROCEDURES_ meta table.

OBJ_TYPE

This is the type of the object related to the privilege.

- T: Table
- S: Sequence
- P: Stored procedure or function
- V: View

WITH_GRANT_OPTION

The WITH_GRANT_OPTION indicates whether the user to whom the privilege was granted is permitted to grant the privilege to other users.

3.1.15.2 See Also

SYS_USERS_

SYS_PRIVILEGES_

SYS_TABLES_

SYS_PROCEDURES_

3.1.16 SYS_GRANT_SYSTEM_

This contains information about system privileges granted to users.

Column	Data Type	Description
GRANTOR_ID	INTEGER	The identifier of the user who granted the privilege
GRANTEE_ID	INTEGER	The identifier of the user to whom the privilege was granted
PRIV_ID	INTEGER	The identifier of the privilege

3.1.16.1 Column Information

GRANTOR_ID

This is the identifier of the user who granted the privilege, and corresponds to a USER_ID in the SYS_USERS_ meta table.

GRANTEE_ID

This is the identifier of the user to whom the privilege was granted, and corresponds to a USER_ID in the SYS_USERS_ meta table.

PRIV_ID

This is the identifier of the privilege, and corresponds to a PRIV_ID found in the SYS_PRIVILEGES_ meta table.

3.1.16.2 See Also

SYS_USERS_

SYS_PRIVILEGES_

3.1.17 SYS_INDEX_COLUMNS_

This is the meta table that contains information about all columns associated with indexes defined for all tables.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the user
INDEX_ID	INTEGER	The identifier of the index
COLUMN_ID	INTEGER	The column identifier
INDEX_COL_ORDER	INTEGER	The position of the column in the index
SORT_ORDER	CHAR(1)	The sort order
TABLE_ID	INTEGER	The table identifier

3.1.17.1 Column Information

USER_ID

This is the identifier of the owner of the index, and corresponds to a USER_ID in the SYS_USERS_ meta table.

INDEX_ID

This is the identifier of the index, and corresponds to an INDEX_ID in the SYS_INDICES_ meta table.

COLUMN_ID

This is the identifier of the column for which the index was created, and corresponds to a COLUMN_ID in the SYS_COLUMNS_ meta table.

INDEX_COL_ORDER

In the case of a composite index, because a single index spans multiple columns, this value indicates the position of the column in the index.

SORT_ORDER

This indicates whether the index is arranged in ascending or descending order.

- A: Ascending order
- D: Descending order

TABLE_ID

This is the identifier of the table in which the index was created, and corresponds to a TABLE_ID value in the SYS_TABLES_ meta table.

3.1.17.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_COLUMNS_

SYS_INDICES_

3.1.18 SYS_INDEX_PARTITIONS_

This is the meta table for managing index partitions.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
INDEX_ID	INTEGER	The index identifier
TABLE_PARTITION_ID	INTEGER	The table partition identifier
INDEX_PARTITION_ID	INTEGER	The index partition identifier

Column name	Type	Description
INDEX_PARTITION_NAME	VARCHAR(40)	The index partition name
PARTITION_MIN_VALUE	VARCHAR(4000)	Reserved for future use
PARTITION_MAX_VALUE	VARCHAR(4000)	Reserved for future use
TBS_ID	INTEGER	The tablespace identifier

3.1.18.1 Column Information

USER_ID

This is the user identifier of the owner of the index. It corresponds to a USER_ID in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table in which the index is created. It is the same as a TABLE_ID value in the SYS_TABLES_ meta table.

INDEX_ID

This is the index identifier, and corresponds to an INDEX_ID in the SYS_INDICES_ meta table.

TABLE_PARTITION_ID

This is the table partition identifier.

INDEX_PARTITION_ID

This is the index partition identifier.

INDEX_PARTITION_NAME

This is the name of the index partition. It is specified by the user.

TBS_ID

This is the identifier of the tablespace in which the index is stored.

3.1.18.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_INDICES_

SYS_TABLE_PARTITIONS_

3.1.19 SYS_INDICES_

This is the meta table that contains information about all indexes defined for all tables.

Column	Data Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
INDEX_ID	INTEGER	The index identifier
INDEX_NAME	VARCHAR(40)	The index name
INDEX_TYPE	INTEGER	The index type
IS_UNIQUE	CHAR(1)	Indicates whether the use of duplicate key values is allowed
COLUMN_CNT	INTEGER	The number of columns in the index
IS_RANGE	CHAR(1)	Indicates whether range scanning is possible using the index
IS_PERS	CHAR(1)	Indicates whether the index is stored perma-nently
TBS_ID	INTEGER	The tablespace identifier
IS_PARTITIONED	CHAR(1)	Indicates whether the index is partitioned
CREATED	DATE	Indicates when the index was created
LAST_DDL_TIME	DATE	The time at which the index was most recently changed using a DDL statement

3.1.19.1 Column Information

USER_ID

This is the identifier of the owner of the index, and corresponds to a USER_ID value in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table in which the index was created, and corresponds to a TABLE_ID of the SYS_TABLES_ meta table.

INDEX_ID

This is an index identifier. It is automatically assigned by the system sequence.

INDEX_NAME

This is the name of the index.

INDEX_TYPE

This indicates the index type. A value of 1 indicates a B-TREE index, while a value of 2 indicates an R-TREE index.

IS_UNIQUE

This indicates whether duplicate key values are allowed.

- T: Do not allow duplicate key values.
- F: Allow duplicate key values.

COLUMN_CNT

This is the number of columns with which the index is associated.

IS_RANGE

This is indicates whether range scanning is possible using the index.

- T: Range scanning is possible.
- F: Range scanning is not possible.

IS_PERS

When a server is powered up, in the case of memory tables, data are read from tables and all indexes are created. Alternatively, when a server is shut down, the indexes can be saved to disk, in which case the indexing information is read directly from the index files that were saved to disk when the server is restarted. This eliminates the expense of constructing indexes when the server is powered up.

Indexes that are saved to disk in index files are called persistent indexes. The user can specify that an index is a persistent index when creating the index.

- T: Permanent index
- F: Non-permanent index

TBS_ID

This is the identifier of the tablespace in which the index was created.

IS_PARTITIONED

This indicates whether the index is partitioned. If it is 'Y', the index is partitioned. If it is 'N', the index is not partitioned.

3.1.19.2 See Also

SYS_USERS_

SYS_TABLES_

3.1.20 SYS_LOBS_

This is the meta table containing information about LOB columns defined in tables.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
COLUMN_ID	INTEGER	The column identifier
TBS_ID	INTEGER	The tablespace identifier
LOGGING	CHAR(1)	This field is reserved for future use.
BUFFER	CHAR(1)	This field is reserved for future use.
IS_DEFAULT_TBS	CHAR(1)	Indicates whether a tablespace is designated for LOB column storage

3.1.20.1 Column Information

USER_ID

This is the identifier of the owner of the table to which the LOB column belongs, and corresponds to a USER_ID value in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table to which the LOB column belongs, and corresponds to a TABLE_ID value in the SYS_TABLES_ meta table.

COLUMN_ID

This is the LOB column identifier.

TBS_ID

This is the identifier of the tablespace to which the LOB column belongs.

IS_DEFAULT_TBS

This indicates whether a tablespace for storing a LOB column was specified by the user when the LOB column was created.

3.1.20.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_COLUMNS_

3.1.21 SYS_PART_INDICES_

This is the meta table for managing partitioned indexes. It contains information on partitioned indexes for which IS_PARTITIONED in SYS_INDICES_ is set to ‘Y’.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
INDEX_ID	INTEGER	The index identifier
PARTITION_TYPE	INTEGER	The partition type
IS_LOCAL_UNIQUE	CHAR(1)	Indicates whether an index is a local unique index

3.1.21.1 Column Information

USER_ID

This is the user identifier of the owner of the index, and corresponds to a USER_ID in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table for which the index was created, and corresponds to a TABLE_ID value in the SYS_TABLES_ meta table.

INDEX_ID

This is the index identifier. It corresponds to an INDEX_ID value in the SYS_INDICES_ meta table.

PARTITION_TYPE

This indicates whether the partition type is LOCAL or GLOBAL. However, because the GLOBAL partition type is not supported at present, it is always 0.

- 0: LOCAL
- 1: GLOBAL

IS_LOCAL_UNIQUE

This indicates whether an index is a local unique index, and can be ‘Y’ or ‘N’.

- Y: A local unique index.
- N: Not a local unique index.

3.1.21.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_INDICES_

3.1.22 SYS_PART_KEY_COLUMNS_

This meta table shows information about the partitioning key columns for the partitioned objects.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
PARTITION_OBJ_ID	INTEGER	The partitioned object identifier
COLUMN_ID	INTEGER	The column identifier
OBJECT_TYPE	INTEGER	The object type
PART_COL_ORDER	INTEGER	The position of the column in the partitioning key (starting with 0)

3.1.22.1 Column Information

USER_ID

This is the identifier of the owner of the partitioned table or index. It corresponds to a USER_ID value in the SYS_USERS_ meta table.

PARTITION_OBJ_ID

This is the identifier of a partitioned object, and corresponds to a TABLE_ID value in the SYS_PART_TABLES_ meta table or INDEX_ID value in the SYS_PART_INDICES_ meta table.

COLUMN_ID

This is the identifier of the column in the partitioning key, and corresponds to a COLUMN_ID value in the SYS_COLUMNS_ meta table.

OBJECT_TYPE

This identifies the type of the object.

- 0: TABLE

- 1: INDEX

PART_COL_ORDER

This is the position of the column in the partitioning key (starting with 0).

3.1.22.2 See Also

SYS_PART_INDICES_

SYS_ TABLES_PARTITIONS_

SYS_COLUMNS_

3.1.23 SYS_PART_LOBS_

This is a meta table for managing LOB columns for respective partitions.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
PARTITION_ID	INTEGER	The partition identifier
COLUMN_ID	INTEGER	The column identifier
TBS_ID	INTEGER	The tablespace identifier
LOGGING	CHAR(1)	This field is reserved for future use.
BUFFER	CHAR(1)	This field is reserved for future use.

3.1.23.1 Column Information

USER_ID

This is the identifier of the owner of the table to which the LOB column belongs, and corresponds to a USER_ID value in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table to which the LOB column belongs, and corresponds to a TABLE_ID value in the SYS_TABLES_ meta table.

PARTITION_ID

This is the identifier of the partition in which the LOB column is stored.

COLUMN_ID

This is the LOB column identifier.

TBS_ID

This is the identifier of the tablespace to which the LOB column belongs.

3.1.23.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_PART_TABLES_

SYS_COLUMNS_

3.1.24 SYS_PART_TABLES_

This is the meta table for the management of partitioned tables. The table information in SYS_PART_TABLES_ is information about partitioned tables for which IS_PARTITIONED in SYS_TABLES_ is set to 'Y'.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
PARTITION_METHOD	INTEGER	The partitioning method
PARTITION_KEY_COUNT	INTEGER	The number of partitioning key columns
ROW_MOVEMENT	CHAR(1)	Indicates whether updated records can be moved between partitions

3.1.24.1 Column Information

USER_ID

This is the identifier of the owner of the index, and corresponds to a USER_ID value in the SYS_USERS_ meta table.

TABLE_ID

This is the identifier of the table in which the index was created, and corresponds to a TABLE_ID value in the SYS_TABLES_ meta table.

PARTITION_METHOD

This indicates the partitioning method.

- 0: RANGE
- 1: HASH
- 2: LIST

ROW_MOVEMENT

This indicates whether it is permissible for records that have been updated to be moved to other partitions when the value of a partition key column is updated.

- T: movement of updated records between partitions is permitted
- F: movement of updated records between partitions is forbidden

3.1.24.2 See Also

SYS_USERS_

SYS_TABLES_

3.1.25 SYS_PRIVILEGES_

This meta table contains information about the kinds of privileges supported by ALTIBASE HDB. For more detailed information, please refer to the descriptions of database privileges and of the GRANT statement in the *SQL Reference*.

Column	Data Type	Description
PRIV_ID	INTEGER	The privilege identifier
PRIV_TYPE	INTEGER	The privilege type
PRIV_NAME	VARCHAR(40)	The privilege name

3.1.25.1 Column Information

PRIV_ID

This is the privilege identifier. It is defined internally by the system.

PRIV_TYPE

This indicates the type of privilege.

- 1: indicates an object privilege

- 2: indicates a system privilege

PRIV_NAME

This is the name of the privilege.

3.1.26 SYS_PROCEDURES_

This table is for storing information about stored procedures and stored functions, such as the stored procedure name, return type, number of parameters, whether it can be executed, etc.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the stored procedure
PROC_OID	BIGINT	The identifier of the stored procedure
PROC_NAME	VARCHAR(40)	The name of the stored procedure
OBJECT_TYPE	INTEGER	Indicates whether the object is a stored procedure, stored function, or type set
STATUS	INTEGER	Indicates the status of the object. The object cannot be executed if it is INVALID. 0: VALID 1: INVALID
PARA_NUM	INTEGER	The number of parameters for the stored procedure
RETURN_DATA_TYPE	INTEGER	The return data type for the stored function
RETURN_LANG_ID	INTEGER	The return type language identifier
RETURN_SIZE	INTEGER	The size of the stored function return data type
RETURN_PRECISION	INTEGER	The precision of the stored function return data type
RETURN_SCALE	INTEGER	The scale of the stored function return data type
PARSE_NO	INTEGER	The number of records containing statement fragments stored in SYS_PROC_PARSE_ for the procedure
PARSE_LEN	INTEGER	The total length of the procedure statement stored in SYS_PROC_PARSE_
CREATED	DATE	The date on which the object was created
LAST_DDL_TIME	DATE	The time when DDL was most recently used to make changes to a stored procedure

3.1.26.1 Column Information

USER_ID

This is the identifier of the owner of the stored procedure or stored function, and corresponds to a USER_ID value in the SYS_USERS_ meta table.

PROC_OID

This is the identifier of the stored procedure or stored function, and is automatically assigned by the system.

PROC_NAME

This is the name of the stored procedure or stored function.

OBJECT_TYPE

This value allows stored procedures to be distinguished from stored functions. Stored functions differ from stored procedures in that they return a value.

- 0: Stored procedure
- 1: Stored function
- 3: Type set

STATUS

This value indicates whether a stored procedure or function may be executed. A value of 0 (VALID) indicates that it can be executed.

If a DDL statement is executed on an object that is accessed by a stored procedure or stored function, the stored procedure or stored function will become invalid. For example, if a new column is added to a table that is accessed by a stored procedure, the stored procedure will need to be recompiled before it can be deemed VALID and executed. The status values are as follows:

- 0: VALID
- 1: INVALID

PARA_NUM

This indicates the number of parameters defined for a stored procedure or stored function.

RETURN_DATA_TYPE

This is the data type identifier for the return value of a stored function. Information on data type identifiers can be found in the DATA_TYPE column of the SYS_COLUMNS_ meta table.

For more information on data types, please refer to Chapter1: Data Types.

RETURN_LANG_ID

This column contains information about the language properties of the character data types (CHAR, VARCHAR).

RETURN_SIZE

This is the physical size of the return data type.

RETURN_PRECISION

This is the precision of the return data type, which is either defined by the user or set based on the system default. For character types, it is the length of the user-defined character type.

RETURN_SCALE

This is the scale of the return data type, which is either defined by the user or set as the system default. Depending on the type, this value may not be used.

For more information about data type precision and scale, please refer to Chapter1: Data Types.

PARSE_NO

Stored procedure and stored function statements are divided into multiple records containing text fragments and stored in the SYS_PROC_PARSE_ meta table. This value indicates the number of records used to store a stored procedure or function.

PARSE_LEN

Stored procedure and stored function statements are divided into multiple records containing text fragments and stored in the SYS_PROC_PARSE_ meta table. This value indicates the overall length of the statement.

LAST_DDL_TIME

This is the most recent time at which a DDL statement was used to make changes to a stored procedure.

3.1.26.2 See Also

SYS_USERS_

3.1.27 SYS_PROC_PARAS_

This meta table contains information about the parameters of stored procedures and stored functions.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the stored procedure
PROC_OID	BIGINT	The identifier of the stored procedure
PARA_NAME	VARCHAR(40)	The parameter name
PARA_ORDER	INTEGER	The parameter order. The first parameter is assigned the number 1.
INOUT_TYPE	INTEGER	Whether the parameter is an Input, Output, or Input/Output parameter
DATA_TYPE	INTEGER	The data type of the parameter
LANG_ID	INTEGER	The language identifier for the parameter type
SIZE	INTEGER	The size of the parameter type
PRECISION	INTEGER	The precision of the parameter type
SCALE	INTEGER	The scale of the parameter type
DEFAULT_VAL	VARCHAR(4000)	The default value for the parameter

3.1.27.1 Column Information

USER_ID

This is the identifier of the user who is the owner of the stored procedure or the stored function, and corresponds to a USER_ID in the SYS_USERS_ meta table.

PROC_OID

This is the identifier of the stored procedure or stored function, and corresponds to a PROC_ID in the SYS_PROCEDURES_ meta table.

PARA_NAME

This is the parameter name.

PARA_ORDER

When there are multiple parameters, this value indicates the position of the parameter in the defined parameter order.

INOUT_TYPE

This value indicates whether the parameter for the stored procedure or stored function is an input, output, or input/output parameter.

- 0: IN
- 1: OUT
- 2: IN/OUT

DATA_TYPE

This is the data type identifier for the parameter. The DATA_TYPE column in the SYS_COLUMNS_ meta table contains information on data type identifiers.

For more information about data types, please refer to Chapter1: Data Types.

LANG_ID

This column displays the language properties for character type parameters (CHAR and VARCHAR).

SIZE

This is the physical size of the data type.

PRECISION

This is the precision of the parameter, which is either determined by the user or set based on the system default. The precision (length) of character data types is defined by the user.

SCALE

This is the scale of the parameter, which is either determined by the user or set to the system default. Depending on the data type, this value may not be used.

For more information on the scale and precision of data types, please refer to Chapter1: Data Types.

DEFAULT_VAL

When a parameter is defined, this is the user-defined default parameter value.

3.1.27.2 See Also

SYS_USERS_

SYS_PROCEDURES_

3.1.28 SYS_PROC_PARSE_

This meta table contains the text constituting user-defined stored procedures and stored functions.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the stored procedure or stored function
PROC_OID	BIGINT	The object identifier of the stored procedure
SEQ_NO	INTEGER	The position of the record among multiple records for a statement that was split and then saved
PARSE	VARCHAR(100)	A fragment of the text of the stored procedure or stored function

3.1.28.1 Column Information

USER_ID

This is the identifier of the owner of the stored procedure or stored function, and corresponds to a USER_ID in the SYS_USERS_ meta table.

PROC_OID

This is the identifier of the stored procedure or the stored function, and corresponds to a PROC_ID in the SYS_PROCEDURES_ meta table.

SEQ_NO

When the information for a statement for one stored procedure is saved across multiple records in SYS_PROC_PARSE_, this is the sequential position of an individual record.

PARSE

This is a line of text belonging to the stored procedure or stored function. An entire statement of a stored procedure can be re-created by retrieving all records that correspond to a single PROC_OID value and combining the PARSE values in order according to the SEQ_NO values.

3.1.28.2 See Also

SYS_USERS_

SYS_PROCEDURES_

3.1.29 SYS_PROC_RELATED_

This table contains information about tables, sequences, stored procedures, stored functions, and views accessed by a stored procedure or stored function.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the stored procedure
PROC_OID	BIGINT	The identifier of the stored procedure
RELATED_USER_ID	INTEGER	The identifier of the owner of an object referenced within a stored procedure
RELATED_OBJECT_NAME	VARCHAR(40)	The name of an object referenced within a stored procedure
RELATED_OBJECT_TYPE	INTEGER	The type of an object referenced within a stored procedure

In the case where stored procedure PROC1 performs INSERT on table t1, the identifiers for the owner of the stored procedure PROC1 and for the stored procedure itself would be stored in USER_ID and PROC_OID respectively, the identifiers for the owner of table t1 and for the table itself would be stored in RELATED_USER_ID and RELATED_OBJECT_NAME respectively, and the number 2 (signifying a table) would be stored in RELATED_OBJECT_TYPE.

3.1.29.1 Column Information

USER_ID

This is the identifier of the owner of the stored procedure or the stored function, and corresponds to a USER_ID in the SYS_USERS_ meta table.

PROC_OID

This is the identifier of the stored procedure or the stored function, and corresponds to a PROC_ID in the SYS_PROCEDURES_ meta table.

RELATED_USER_ID

This is the identifier of the owner of the object accessed by the stored procedure, and corresponds to a USER_ID in the SYS_USERS_ meta table.

RELATED_OBJECT_NAME

This is the name of the object accessed by the stored procedure.

RELATED_OBJECT_TYPE

This is the type of the object accessed by the stored procedure. The possible values are as follows:

0: Stored procedure

1: Stored function

2: Table, Sequence, View

3: Type set

4: Database link

3.1.29.2 See Also

SYS_USERS_

SYS_PROCEDURES_

SYS_TABLES_

3.1.30 SYS_REPLICATIONS_

This meta table contains information related to replication.

Column	Data Type	Description
REPLICATION_NAME	VARCHAR(40)	The name of the replication object
LAST_USED_HOST_NO	INTEGER	The most recently used remote server
HOST_COUNT	INTEGER	The number of remote servers
IS_STARTED	INTEGER	Whether replication is active
XSN	BIGINT	The Restart SN (Sequence Number), i.e. the SN from which the Sender will resume transmission of XLogs
ITEM_COUNT	INTEGER	The number of replication target tables
CONFLICT_RESOLUTION	INTEGER	The replication conflict resolution method
REPL_MODE	INTEGER	The default replication mode
ROLE	INTEGER	The role of the sender thread
OPTIONS	INTEGER	A flag for additional replication features
INVALID_RECOVERY	INTEGER	Whether replication recovery is possible
REMOTE_FAULT_DETECT_TIME	DATE	The time at which a fault was detected on a remote server

3.1.30.1 Column Information

REPLICATION_NAME

This is the name of the replication object, and is set by the user when the replication object is created.

LAST_USED_HOST_NO

This is the number of the most recently used remote server, and corresponds to a HOST_NO in the SYS_REPL_HOSTS_ meta table.

HOST_COUNT

This is the number of remote servers involved in replication, and is equal to the number of IP addresses stored in SYS_REPL_HOSTS_.

IS_STARTED

Indicates whether replication is active.

- 0: suspended
- 1: active

XSN

This indicates the SN from which the Sender thread must begin sending logs when replication is started.

ITEM_COUNT

This is the number of replication target tables. This number corresponds to the number of records in the SYS_REPL_ITEMS_ meta table for this replication object, with one record corresponding to each of these tables.

CONFLICT_RESOLUTION

This describes the replication conflict resolution method.

- 0: Default
- 1: Act as the Master server
- 2: Act as the Slave server

Please refer to the *Replication Manual* for detailed information about replication conflict resolution methods.

REPL_MODE

This is the default replication mode, which is set when the replication object is created.

- 0: LAZY MODE (Default)
- 2: EAGER MODE

The default replication mode is used if the ALTER SESSION SET REPLICATION statement is not used to set the replication mode for a session.

For detailed information about the default replication mode, please refer to the *Replication Manual*, and for detailed information about the ALTER SESSION SET REPLICATION statement, please refer to the *SQL Reference*.

ROLE

This indicates the role of the Sender thread.

- 0: Replication
- 1: Log Analyzer

For more information, please refer to the *Log Analyzer User's Manual*.

OPTIONS

This flag indicates whether to use the recovery and offline options, which are extra replication features.

- 0: do not use the recovery or offline options
- 1: use the recovery option
- 2: use the offline option

INVALID_RECOVERY

This value indicates whether recovery using replication is possible.

- 0: replication-based recovery is possible.
- 1: replication-based recovery is not possible.

REMOTE_FAULT_DETECT_TIME

This is the time at which a fault was detected on a remote server while replication was running.

3.1.31 SYS_REPL_HOSTS_

This meta table contains information related to remote servers defined in replication objects.

Column	Data Type	Description
HOST_NO	INTEGER	The host identifier
REPLICATION_NAME	VARCHAR(40)	The replication name
HOST_IP	VARCHAR(64)	The IP address of the remote server
PORT_NO	INTEGER	The replication port number on the remote server

3.1.31.1 Column Information

HOST_NO

This is the serial number of the remote server, which is automatically assigned by the system sequence.

REPLICATION_NAME

This is the name of the replication object set by the user, and corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table.

HOST_IP

This is the IP address of the remote server.

PORT_NO

This is the replication port number on the remote server.

3.1.31.2 See Also

SYS_REPLICATIONS_

3.1.32 SYS_REPL_ITEMS_

This meta table contains information about replication target tables.

Column	Data Type	Description
REPLICATION_NAME	VARCHAR(40)	The replication name
TABLE_OID	BIGINT	The table object identifier
LOCAL_USER_NAME	VARCHAR(40)	The name of a user owning a target table on the local server
LOCAL_TABLE_NAME	VARCHAR(40)	The name of a target table on the local server
LOCAL_PARTITION_NAME	VARCHAR(40)	The name of a partition on the local server
REMOTE_USER_NAME	VARCHAR(40)	The name of a user owning a target table on the remote server
REMOTE_TABLE_NAME	VARCHAR(40)	The name of a target table on the remote server
REMOTE_PARTITION_NAME	VARCHAR(40)	The name of a partition on the remote server
IS_PARTITION	CHAR(1)	Whether or not a table is partitioned

Column	Data Type	Description
INVALID_MAX_SN	BIGINT	The highest log SN to skip
CONDITION	VARCHAR(1000)	A Replication Condition Clause

One replication object can pertain to more than one table, and SYS_REPL_ITEMS_ has a record for each of these tables. For example, if a replication pertains to 10 tables, this meta table will contain 10 records pertaining to this replication.

3.1.32.1 Column Information

REPLICATION_NAME

This is the name of the replication object, which is defined by the user, and corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table.

TABLE_OID

This is the identifier of the replication target table, and corresponds to a TABLE_OID in the SYS_TABLES_ meta table.

LOCAL_USER_NAME

This is the user name of the owner of the replication target table in the local system, and corresponds to a USER_NAME in the SYS_USERS_ meta table.

LOCAL_TABLE_NAME

This is the name of the replication target table in the local system, and corresponds to a TABLE_NAME in the SYS_TABLES_ meta table.

LOCAL_PARTITION_NAME

This is the name of the replication target partition on the local server.

REMOTE_USER_NAME

This is the user name of the owner of the replication target table in the remote system, and corresponds to a USER_NAME in the SYS_USERS_ meta table.

REMOTE_TABLE_NAME

This is the name of the replication target table in the remote system, and corresponds to a TABLE_NAME in the SYS_TABLES_ meta table.

REMOTE_PARTITION_NAME

This is the name of the replication target partition on the remote server.

IS_PARTITION

This is an identifier indicating whether a table is partitioned. If it is 'Y', the table is partitioned. If it is 'N', the table is not partitioned.

INVALID_MAX_SN

If DDL statements or Sync operations are executed on replication target tables, the most recently recorded SN is saved here. Table logs up to this SN are skipped when the table is replicated.

CONDITION

This is a condition clause, which is input by a user and takes effect when replication is executed.

3.1.32.2 See Also

SYS_REPLICATIONS_

SYS_USERS_

SYS_TABLES_

3.1.33 SYS_REPL_OFFLINE_DIR_

This meta table stores log directory information related to the offline replication option.

Column name	Type	Description
REPLICATION_NAME	VARCHAR(40)	The replication name
LFG_ID	INTEGER	The identifier of the log file group
PATH	VARCHAR(512)	The offline log path

3.1.33.1 Column Information

REPLICATION_NAME

This is the user-defined replication name. It corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table.

LFG_ID

One archive directory exists for each LFG (Log File Group). This is the identifier for this LFG.

PATH

This is the absolute path in the system where the log file is saved.

3.1.34 SYS_REPL_OLD_COLUMNS_

This meta table is for storing information on columns that are currently replicated by the replication Sender thread.

Column name	Type	Description
REPLICATION_NAME	VARCHAR(40)	The name of the replication object
TABLE_OID	BIGINT	The object identifier of the table
COLUMN_NAME	VARCHAR(40)	The column name
MT_DATATYPE_ID	INTEGER	The data type identifier
MT_LANGUAGE_ID	INTEGER	The language identifier
MT_FLAG	INTEGER	An internal flag
MT_PRECISION	INTEGER	The number of digits
MT_SCALE	INTEGER	The number of digits to the right of the decimal point
MT_ENCRYPT_PRECISION	INTEGER	The number of digits in an encrypted column
MT_POLICY_NAME	VARCHAR(16)	The name of the policy used for an encrypted column
SM_ID	INTEGER	The column identifier
SM_FLAG	INTEGER	An internal flag
SM_OFFSET	INTEGER	The internal offset
SM_SIZE	INTEGER	The internal size

3.1.34.1 Column Information

REPLICATION_NAME

This is the replication name, which is specified by the user. It corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table.

TABLE_OID

This is the identifier for a replication target table currently being used by the replication Sender thread. Its value may not correspond to any TABLE_OID value in SYS_TABLES_.

COLUMN_NAME

This is the name of a column currently being replicated by the replication Sender thread.

MT_DATATYPE_ID

This is the data type identifier, and is an internal value.

MT_LANGUAGE_ID

This is the language identifier, and is an internal value.

MT_FLAG

This is an internal flag used by ALTIBASE HDB.

MT_PRECISION

For a numeric type column, this is the number of digits in the column.

MT_SCALE

For a numeric type column, this is the number of digits to the right of the decimal point in the column.

MT_ENCRYPT_PRECISION

For an encrypted numeric type column, this is the number of digits in the column.

MT_POLICY_NAME

For an encrypted column, this is the name of the policy used for the column.

SM_ID

This is the column identifier. Column identifiers start with 0.

SM_FLAG

This is a flag internally used by ALTIBASE HDB.

SM_OFFSET

This is an offset value internally used by ALTIBASE HDB.

SM_SIZE

This is a size value internally used by ALTIBASE HDB.

3.1.34.2 See Also

SYS_REPL_OLD_ITEMS_

SYS_REPL_OLD_INDICES_

SYS_REPL_OLD_INDEX_COLUMNS_

3.1.35 SYS_REPL_OLD_INDEX_COLUMNS_

This meta table is for storing information on columns currently being replicated by the replication Sender thread.

Column name	Type	Description
REPLICATION_NAME	VARCHAR(40)	The replication name
TABLE_OID	BIGINT	The table object identifier
INDEX_ID	INTEGER	The index identifier
KEY_COLUMN_ID	INTEGER	The column identifier
KEY_COLUMN_FLAG	INTEGER	An internal flag
COMPOSITE_ORDER	INTEGER	The position of the column on which the index is based

3.1.35.1 Column Information

REPLICATION_NAME

This value corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table, and is the user-defined replication name.

TABLE_OID

This is the identifier of a table currently being replicated by the replication Sender thread. Its value may not correspond to any TABLE_OID value in SYS_TABLES_.

INDEX_ID

This is the identifier of an index currently being replicated by the replication Sender thread.

KEY_COLUMN_ID

This is the identifier of the column on which the index is based.

KEY_COLUMN_FLAG

This is an internal flag for the column on which the index is based.

COMPOSITE_ORDER

This is the position of the column on which the index is based.

3.1.35.2 See Also

SYS_REPL_OLD_ITEMS_

SYS_REPL_OLD_COLUMNS_

SYS_REPL_OLD_INDICES_

3.1.36 SYS_REPL_OLD_INDICES_

This meta table contains information about indexes currently being replicated by the replication Sender thread.

Column name	Type	Description
REPLICATION_NAME	VARCHAR(40)	The replication name
TABLE_OID	BIGINT	The object identifier of the table
INDEX_ID	INTEGER	The index identifier
INDEX_NAME	VARCHAR(40)	The index name
TYPE_ID	INTEGER	The index type identifier
IS_UNIQUE	CHAR(1)	Indicates whether or not the index is globally unique
IS_LOCAL_UNIQUE	CHAR(1)	Indicates whether or not the index is locally unique
IS_RANGE	CHAR(1)	Indicates whether or not range scanning is possible using the index

3.1.36.1 Column Information

REPLICATION_NAME

This is the user-defined replication name. Its value corresponds to a REPLICATION_NAME value in the SYS_REPLICATIONS_ meta table.

TABLE_OID

This is the identifier of a table currently being replicated by the replication Sender thread. Its value may be different from that of TABLE_OID in the SYS_TABLES_ meta table.

INDEX_ID

This is the identifier of an index currently being replicated by the replication Sender thread.

INDEX_NAME

This is the name of an index currently being replicated by the replication Sender thread.

TYPE_ID

This is an index type identifier, and is an internal value.

IS_UNIQUE

This indicates whether or not the index is globally unique. 'Y' signifies that the identifier is globally unique, and 'N' signifies that it is not globally unique.

IS_LOCAL_UNIQUE

This indicates whether or not the index is locally unique. 'Y' signifies that it is locally unique, and 'N' means that it is not locally unique.

IS_RANGE

This indicates whether or not range scanning is possible using the index. 'Y' means that range scanning is possible, and 'N' means that range scanning is impossible.

3.1.36.2 See Also

SYS_REPL_OLD_ITEMS_

SYS_REPL_OLD_COLUMNS_

SYS_REPL_OLD_INDEX_COLUMNS_

3.1.37 SYS_REPL_OLD_ITEMS_

This meta table contains information on tables currently being replicated by the replication Sender thread.

Column name	Type	Description
REPLICATION_NAME	VARCHAR(40)	The name of the replication
TABLE_OID	BIGINT	The table object identifier
USER_NAME	VARCHAR(40)	The user name
TABLE_NAME	VARCHAR(40)	The table name
PARTITION_NAME	VARCHAR(40)	The partition name
PRIMARY_KEY_INDEX_ID	INTEGER	The index identifier of the primary key

3.1.37.1 Column Information

REPLICATION_NAME

This value corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table, and is the user-defined replication name.

TABLE_OID

This is the identifier of a table currently being replicated by the replication Sender thread. Its value may be different from the value of TABLE_OID in the SYS_TABLES_ meta table.

USER_NAME

This is the user name of the owner of the table being replicated on the local server. Its value corresponds to a USER_NAME in the SYS_USERS_ meta table.

TABLE_NAME

This is the name of the table being replicated on the local server. Its value corresponds to a TABLE_NAME value in the SYS_TABLES_ meta table.

PARTITION_NAME

This is the name of the partition containing the table being replicated on the local server.

PRIMARY_KEY_INDEX_ID

This is the identifier of a primary key index.

3.1.37.2 See Also

SYS_REPL_OLD_COLUMNS_

SYS_REPL_OLD_INDICES_

SYS_REPL_OLD_INDEX_COLUMNS_

3.1.38 SYS_REPL_RECOVERY_INFOS_

This is the meta table in which log information is written for use in recovery of the remote server.

Column name	Type	Description
REPLICATION_NAME	VARCHAR(40)	The name of the replication
MASTER_BEGIN_SN	BIGINT	The starting log number of a master transaction

Column name	Type	Description
MASTER_COMMIT_SN	BIGINT	The final log number of the master transaction
REPLICATED_BEGIN_SN	BIGINT	The starting log number of a replication transaction
REPLICATED_COMMIT_SN	BIGINT	The final log number of the replication transaction

3.1.38.1 Column Information

REPLICATION_NAME

This is the replication object name defined by the user, and corresponds to a REPLICATION_NAME in the SYS_REPLICATIONS_ meta table.

MASTER_BEGIN_SN

The starting log number of a master transaction occurring on a remote server.

MASTER_COMMIT_SN

The final log number of a master transaction occurring on a remote server.

REPLICATED_BEGIN_SN

The starting log number of a replication transaction occurring on the local server.

REPLICATED_COMMIT_SN

The final log number of a replication transaction occurring on the local server.

3.1.38.2 See Also

SYS_REPLICATIONS_

3.1.39 SYS_SECURITY_

This table contains information about the state of the security module.

Column	Data Type	Description
MODULE_NAME	VARCHAR(24)	The name of the security module
MODULE_VERSION	VARCHAR(40)	The version of the security module
ECC_POLICY_NAME	VARCHAR(16)	The name of the ECC policy

Column	Data Type	Description
ECC_POLICY_CODE	VARCHAR(64)	The verification code of the ECC policy

This table shows whether a security module authored by a third party is being used.

In the case where a security module authored by a third party is in use, the SYS_SECURITY_ meta table contains information about the properties of the security module, whereas if no such security module is in use, the SYS_SECURITY_ meta table will contain no records.

3.1.40 SYS_SYNONYMS_

This is the table for storing information about synonyms, which provide alias functions for database objects.

Column	Data Type	Description
SYNONYM_OWNER_ID	INTEGER	The user identifier
SYNONYM_NAME	VARCHAR(40)	The synonym name
OBJECT_OWNER_NAME	VARCHAR(40)	The name of the object owner
OBJECT_NAME	VARCHAR(40)	The name of the synonym target object
CREATED	DATE	The time at which the synonym was created
LAST_DDL_TIME	DATE	The most recent time at which a DDL statement was used to make changes to a synonym

3.1.40.1 Column Information

SYNONYM_OWNER_ID

This is the identifier of the owner of the synonym, and corresponds to a USER_ID in the SYS_USERS_ meta table.

SYNONYM_NAME

This is the synonym name, which is defined by the user.

OBJECT_OWNER_NAME

This is the name of the owner of the schema containing the object that is the target of the user-defined synonym.

OBJECT_NAME

This is the name of the object targeted by the user-defined synonym.

CREATED

This is the time at which the synonym was created.

LAST_DDL_TIME

This is the most recent time at which a DDL statement was used to create or make changes to the synonym.

3.1.40.2 See Also

SYS_USERS_

3.1.41 SYS_TABLES_

This table contains information on meta tables, user-defined tables, sequences and views.

Column	Data Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
TABLE_OID	BIGINT	The table object identifier
COLUMN_COUNT	INTEGER	The number of columns in the table
TABLE_NAME	VARCHAR(40)	The name of the table
TABLE_TYPE	CHAR(1)	The object type
REPLICATION_COUNT	INTEGER	The number of replications related to the table
REPLICATION_RECOVERY_COUNT	INTEGER	The number of replications that use the recovery option and are related to the table
MAXROW	BIGINT	The maximum number of records that can be entered (0: no limit)
TBS_ID	INTEGER	The tablespace identifier
PCTFREE	INTEGER	See below
PCTUSED	INTEGER	See below
INIT_TRANS	INTEGER	The initial number of transactions that can be simultaneously used for update in a page
MAX_TRANS	INTEGER	The maximum number of transactions that can be simultaneously used for update in a page
INITEXTENTS	BIGINT	The initial number of extents when a table is created

Column	Data Type	Description
NEXTEXTENTS	BIGINT	The number of extents that are added when a table is expanded
MINEXTENTS	BIGINT	The minimum number of extents in a table
MAXEXTENTS	BIGINT	The maximum number of extents in a table
IS_PARTITIONED	CHAR(1)	Indicates whether a table is partitioned
CREATED	DATE	The time at which the table was created
LAST_DDL_TIME	DATE	The time at which the table was most recently changed using a DDL statement

3.1.41.1 Column Information

USER_ID

This is the identifier of the owner of the table, and corresponds to a USER_ID in the SYS_USERS_ meta table.

TABLE_ID

This is the table identifier, which is automatically assigned by the system sequence.

TABLE_OID

This is the table object identifier, which is automatically and internally assigned by the system. Unlike TABLE_ID, which is used when the user reads meta tables, this value is used only for internal operations.

COLUMN_COUNT

This is the number of columns defined in the table.

TABLE_NAME

This is the table name, which is defined by the user.

TABLE_TYPE

Information not only about tables, but also about sequences, views, etc. is saved in the SYS_TABLES_ meta table. This type identifier is used to distinguish them, and comprises the following types:

- T: Table
- S: Sequence
- V: View
- W: Sequence for Queue Use Only

- Q: Queue

REPLICATION_COUNT

This is the number of replication objects associated with the table.

REPLICATION_RECOVERY_COUNT

This is the number of replication objects that use the recovery option and are associated with the table.

MAXROW

This is the maximum number of records that can be inserted into the table.

TBS_ID

This is the identifier of the tablespace in which the table is saved.

PCTFREE

This is the minimum percentage of free space that must exist in order for it to be possible to update a page. Usually, an amount of space equal to the percentage specified in PCTFREE is kept free so that existing rows saved in a page can be updated. For example, if PCTFREE is set to 20, 20% of the space in the page is set aside for update operations, so data can be inserted only into 80% of the space in the page.

The user can set PCTFREE between 0 and 99 when executing the CREATE TABLE statement.

PCTUSED

This is a threshold below which the amount of used space in a page must decrease in order for the page to return to the state in which records can be inserted from the state in which only update operations are possible. If the amount of free space falls below the percentage specified in PCTFREE, it will become impossible to insert new records into the page, and it will only be possible to update and delete rows. If subsequent update or delete operations reduce the percentage of used space below the threshold specified by PCTUSED, it will become possible to insert new rows into the page again.

The user can set PCTUSED between 0 and 99 when the CREATE TABLE statement is executed.

* For more detailed explanations of PCTFREE and PCTUSED, please refer to the description of the CREATE TABLE statement in the *SQL Reference*.

INIT_TRANS

This is the initial number of update transactions that can be simultaneously executed, and is set when a page is created. The actual number of transactions can increase to the number specified in MAX_TRANS, as long as sufficient page space is available.

MAX_TRANS

This is the maximum number of update transactions that can be simultaneously executed for a single page.

INITEXTENTS

This denotes the number of extents that are available to be allocated when a table is created.

NEXTEXTENTS

This denotes the number of additional extents that are available to be allocated when the size of a table is increased.

MINEXTENTS

This denotes the minimum number of available extents for a table.

MAXEXTENTS

This denotes the maximum number of available extents for a table.

IS_PARTITIONED

This is an identifier that indicates whether a table is partitioned. If it is 'T', the table is partitioned. If it is 'F', the table is not partitioned.

3.1.41.2 See Also

SYS_USERS_

3.1.42 SYS_TABLE_PARTITIONS_

This is a meta table for the management of table partitions.

Column name	Type	Description
USER_ID	INTEGER	The user identifier
TABLE_ID	INTEGER	The table identifier
PARTITION_OID	BIGINT	The partition object identifier
PARTITION_ID	INTEGER	The partition identifier
PARTITION_NAME	VARCHAR(40)	The partition name
PARTITION_MIN_VALUE	VARCHAR(4000)	The minimum reference value for a partition (NULL in the case of a hash partition)
PARTITION_MAX_VALUE	VARCHAR(4000)	The maximum reference value for a partition (NULL in the case of a hash partition)
PARTITION_ORDER	INTEGER	The position of the partition (required for hash partitions)

Column name	Type	Description
TBS_ID	INTEGER	The identifier of a tablespace

3.1.42.1 Column Information

USER_ID

This is the identifier of the table owner, and corresponds to a USER_ID in the SYS_USERS_ meta table.

TABLE_ID

This is the table identifier. It is assigned automatically by the system sequence.

PARTITION_OID

This is the partition object identifier. It is assigned automatically by the system. Unlike PARTITION_ID, which is used when viewing meta tables, it is used only internally by the system.

PARTITION_ID

This is the partition identifier.

PARTITION_NAME

This is the user-defined partition name.

PARTITION_MIN_VALUE

This is a string that gives the minimum reference value for a partition. It is NULL for hash partitions.

PARTITION_MAX_VALUE

This is a string that gives the maximum reference value for a partition. It is NULL for hash partitions.

PARTITION_ORDER

This is the position of the partition among the partitions. It is required for hash partitions.

TBS_ID

This is the identifier of the tablespace in which the table is stored.

3.1.42.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_PART_TABLES_

3.1.43 SYS_TBS_USERS_

This meta table contains information about the relationship between users and user-defined tablespaces.

Column	Data Type	Description
TBS_ID	INTEGER	The tablespace identifier
USER_ID	INTEGER	The user identifier
IS_ACCESS	INTEGER	Whether the user is allowed to access the tablespace

3.1.43.1 Column Information

TBS_ID

This is the tablespace identifier.

USER_ID

This is the identifier of a particular user. It corresponds to a USER_ID in the SYS_USERS_ meta table.

IS_ACCESS

This indicates whether the user is permitted to access the tablespace.

- 0: access not permitted
- 1: access permitted

3.1.43.2 See Also

SYS_USERS_

3.1.44 SYS_TRIGGERS_

This meta table contains default information about triggers.

Column	Data Type	Description
USER_ID	INTEGER	The user identifier
USER_NAME	VARCHAR(40)	The user name
TRIGGER_OID	BIGINT	The trigger identifier
TRIGGER_NAME	VARCHAR(40)	The trigger name

Column	Data Type	Description
TABLE_ID	INTEGER	The table identifier
IS_ENABLE	INTEGER	Indicates whether the trigger is enabled
EVENT_TIME	INTEGER	Indicates when the trigger fires
EVENT_TYPE	INTEGER	The trigger event type
UPDATE_COLUMN_CNT	INTEGER	The number of columns that can cause a trigger to fire if updated
GRANULARITY	INTEGER	The units in which the trigger is executed
REF_ROW_CNT	INTEGER	The number of ALIASes for a REFERENCING statement
SUBSTRING_CNT	INTEGER	The number of records in which the trigger statement is saved
STRING_LENGTH	INTEGER	The total length of the trigger statement character string
CREATED	DATE	The time at which the trigger was created
LAST_DDL_TIME	DATE	The most recent time at which a DDL statement was used to make changes to the trigger

3.1.44.1 Column Information

USER_ID

This is the identifier of the user who owns the trigger, and corresponds to a USER_ID in the SYS_USERS_ meta table.

USER_NAME

This is the user name, and corresponds to a USER_NAME in the SYS_USERS_ meta table.

TRIGGER_OID

This is the trigger identifier. It is automatically assigned by the system.

TRIGGER_NAME

This is the user-defined trigger name.

TABLE_ID

This is the identifier of the table on which the trigger is defined, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

IS_ENABLE

This value indicates whether or not the trigger is enabled. It can be modified using the ALTER TRIGGER statement.

- 0: DISABLED
- 1: ENABLED

EVENT_TIME

This value classifies triggers based on whether they fire before or after the event that caused them.

- 1: BEFORE
- 2: AFTER

EVENT_TYPE

This is the type of the event that causes the trigger to fire.

- 1: INSERT
- 2: DELETE
- 4 UPDATE

UPDATE_COLUMN_CNT

This is the number of columns that cause a trigger to fire when updated. This value is equal to the number of records related to the trigger in the SYS_TRIGGER_UPDATE_COLUMNS_ meta table.

GRANULARITY

This value indicates how often the trigger fires:

- 1: FOR EACH ROW
- 2: FOR EACH STATEMENT

REF_ROW_CNT

This is the number of ALIASes defined in a REFERENCING statement.

SUBSTRING_CNT

One trigger statement is divided into several records and stored in the SYS_TRIGGER_STRINGS_ meta table. This value indicates the number of records used to store the statement.

STRING_LENGTH

This is the total length of the trigger statement character string.

3.1.44.2 See Also

SYS_USERS_

SYS_TABLES_

3.1.45 SYS_TRIGGER_DML_TABLES_

This meta table contains information about tables referenced by triggers.

Column	Data Type	Description
TABLE_ID	INTEGER	The table identifier
TRIGGER_OID	BIGINT	The trigger identifier
DML_TABLE_ID	INTEGER	The table identifier within the trigger
STMT_TYPE	INTEGER	The type of executable statement

3.1.45.1 Column Information

TABLE_ID

This is the identifier of the table on which the trigger is defined, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

TRIGGER_OID

This is the trigger identifier, and corresponds to a TRIGGER_OID in the SYS_TRIGGERS_ meta table.

DML_TABLE_ID

This is the identifier of the table that is accessed using the DML statements within the trigger, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

STMT_TYPE

This is the type of statement executed on a table.

- 8: DELETE
- 19: INSERT
- 33: UPDATE

3.1.45.2 See Also

SYS_TABLES_

SYS_TRIGGERS_

3.1.46 SYS_TRIGGER_STRINGS_

This is the meta table in which the trigger statements are saved.

Column	Data Type	Description
TABLE_ID	INTEGER	The table identifier
TRIGGER_OID	BIGINT	The trigger identifier
SEQNO	INTEGER	The position of this text fragment in the trigger statement
SUBSTRING	VARCHAR(100)	A fragment of trigger statement text

3.1.46.1 Column Information

TABLE_ID

This is the table identifier, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

TRIGGER_OID

This is the trigger identifier, and corresponds to a TRIGGER_OID in the SYS_TRIGGERS_ meta table.

SEQNO

When information about a single trigger statement is saved as several records in SYS_TRIGGER_STRINGS, this is the position of this record among the records.

SUBSTRING

This is a fragment of the trigger statement text. When records are searched for using a single TRIGGER_OID and their SUBSTRING values are concatenated in the order described in SEQNO, the complete trigger command can be reconstructed.

3.1.46.2 See Also

SYS_TABLES_

SYS_TRIGGERS_

3.1.47 SYS_TRIGGER_UPDATE_COLUMNS_

This meta table contains information about columns that cause triggers to fire when updated.

Column	Data Type	Description
TABLE_ID	INTEGER	The table identifier

Column	Data Type	Description
TRIGGER_OID	BIGINT	The trigger identifier
COLUMN_ID	INTEGER	The column identifier

3.1.47.1 Column Information

TABLE_ID

This is the table identifier, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

TRIGGER_OID

This is the trigger identifier, and corresponds to a TRIGGER_OID in the SYS_TRIGGERS_ meta table.

COLUMN_ID

This is the column ID, and corresponds to a COLUMN_ID in the SYS_COLUMNS_ meta table.

3.1.47.2 See Also

SYS_TABLES_

SYS_TRIGGERS_

3.1.48 SYS_USERS_

This meta table contains information about database users.

Column	Data Type	Description
USER_ID	INTEGER	The user identifier
USER_NAME	VARCHAR(40)	The user name
PASSWORD	VARCHAR(40)	The user password
DEFAULT_TBS_ID	INTEGER	The default tablespace identifier
TEMP_TBS_ID	INTEGER	The temporary tablespace identifier
CREATED	DATE	The time at which the database user was created
LAST_DDL_TIME	DATE	The most recent time at which a DDL statement was used to make changes to the user

3.1.48.1 Column Information

USER_ID

This is the user identifier. It is automatically assigned by the system sequence.

USER_NAME

This is the user-defined user name.

PASSWORD

This is the encrypted user password.

DEFAULT_TBS_ID

This is the identifier of the default tablespace, which is used when the user creates an object without explicitly specifying a tablespace.

TEMP_TBS_ID

This is the identifier for the user temporary tablespace.

3.1.49 SYS_VIEWS_

Basic information about views is stored in the SYS_TABLES_ meta table. This meta table contains additional information about views.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the view
VIEW_ID	INTEGER	The view identifier
STATUS	INTEGER	The view status

3.1.49.1 Column Information

USER_ID

This is the identifier of the view owner, and corresponds to a USER_ID in the SYS_USERS_ meta table.

VIEW_ID

This is the view identifier, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

STATUS

This value indicates the status of the view:

- 0: VALID
- 1: INVALID

3.1.49.2 See Also

SYS_USERS_

SYS_TABLES_

3.1.50 SYS_VIEW_PARSE_

This meta table contains the text of view creation statements.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the view
VIEW_ID	INTEGER	The identifier of the view
SEQ_NO	INTEGER	When a view creation statement text is split and the text is saved as multiple text fragments in SYS_VIEW_PARSE_, this is the position of the record among the records.
PARSE	VARCHAR(100)	A text fragment of the view creation statement

3.1.50.1 Column Information

USER_ID

This is the identifier of the view owner, and corresponds to a USER_ID in the SYS_USERS_ meta table.

VIEW_ID

This is the view identifier, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

SEQ_NO

When a single statement corresponding to one view is saved as multiple records in SYS_VIEW_PARSE_, this is the position of the record among the records.

PARSE

When records are searched for using a single VIEW_ID and their PARSE values are concatenated in the order described in SEQ_NO, the complete view statement can be reconstructed.

3.1.50.2 See Also

SYS_USERS_

SYS_TABLES_

3.1.51 SYS_VIEW_RELATED_

This meta table contains information about objects accessed by user-defined views.

Column	Data Type	Description
USER_ID	INTEGER	The identifier of the owner of the view
VIEW_ID	INTEGER	The view identifier
RELATED_USER_ID	INTEGER	The identifier of the owner of the object that the view accesses
RELATED_OBJECT_NAME	VARCHAR(40)	The name of the object accessed by the view
RELATED_OBJECT_TYPE	INTEGER	The type of the object accessed by the view

3.1.51.1 Column Information

USER_ID

This is the identifier of the view owner, and corresponds to a USER_ID in the SYS_USERS_ meta table.

VIEW_ID

This is the identifier of the view, and corresponds to a TABLE_ID in the SYS_TABLES_ meta table.

RELATED_USER_ID

This is the identifier of the owner of the object accessed by the view, and corresponds to a USER_ID in the SYS_USERS_ meta table.

RELATED_OBJECT_NAME

This is the name of the object accessed by the view.

RELATED_OBJECT_TYPE

This identifies the type of object accessed by the view. Views can access stored functions, tables, sequences, other views, Database Link objects, and synonyms. The identifiers are as follows:

1: Stored function

2: Table, Sequence, View

4: Database link

5: Synonym

3.1.51.2 See Also

SYS_USERS_

SYS_TABLES_

SYS_PROCEDURES_

3.1.52 SYS_XA_HEURISTIC_TRANS_

This is a meta table that contains identifiers and information about the status of the database's global transactions.

Column name	Type	Description
FORMAT_ID	BIGINT	The identifier of the format of the global transaction
GLOBAL_TX_ID	VARCHAR(128)	The identifier of the global transaction
BRANCH_QUALIFIER	VARCHAR(128)	The branch qualifier of the global transaction
STATUS	INTEGER	The status of the global transaction

3.1.52.1 Column Information

FORMAT_ID

This is the identifier of the format of the global transaction.

GLOBAL_TX_ID

This is the identifier of the global transaction.

BRANCH_QUALIFIER

This is the branch qualifier of the global transaction.

STATUS

This is the status of the global transaction.

3.2 Performance Views

Performance views are structures that exist in memory but have the form of regular tables, and allow users to monitor internal information about an ALTIBASE HDB system, such as system memory, process status, sessions, buffers, threads, etc.

Performance views allow ALTIBASE HDB users to easily obtain information about memory objects (e.g. session information, log information, thread information) using SQL statements while ALTIBASE HDB is running, in the same way that they would use SQL to search for data saved in regular tables.

This section describes the kinds of performance views provided with ALTIBASE HDB, their structure and function, how to access them, and the information that each view provides.

Note: Performance views provide data on memory objects which are in use by ALTIBASE HDB. Therefore, information about memory objects which have already been released cannot appear in performance views.

For example, when stopping a Replication Sender Thread, the thread object is freed and information about it cannot appear in the V$REPSENDER performance view.

3.2.1 Structures and Features

Inside ALTIBASE HDB there is not only information about user-created objects such as tables; there is also a variety of kinds of information required for the operation of the DBMS itself. Because ALTIBASE HDB has a hybrid structure, in which tables can be created and queried not only in memory space but also in disk space, monitoring ALTIBASE HDB is particularly critical.

Performance views provide information about most of the internal memory structures used by ALTIBASE HDB processes in the form of views. Because the data are dynamically created in real time when a view is queried, users can always obtain up-to-date information about internal processes.

Performance views are always read-only. If a user attempts to modify the data in a performance view, ALTIBASE HDB returns an error and rolls back the transaction.

3.2.2 How to Use Performance Views

Users can retrieve the entire list of performance views by executing the "SELECT * FROM V$TAB" query statement from iSQL as follows:

```
iSQL> SELECT * FROM V$TAB;
```

Performance view schemas can be checked from iSQL using the DESC command, just as with regular tables, and SELECT statements can also be used to query data in the same way that they would be used to query regular tables.

3.2.3 V$ Views

Performance views are identified by the prefix V$. The following table lists all performance views.

Name	Description
V$ALLCOLUMN	Information on the columns that make up a performance view
V$ARCHIVE	Archive and backup- related information
V$BUFFPAGEINFO	Statistics on the buffer frame of the buffer manager
V$BUFFPOOL_STAT	Buffer pool related statistics, including the buffer pool hit ratio
V$CATALOG	Information about the structure of tables
V$DATABASE	Internal information about memory database space
V$DATAFILES	Information on data files which are related to tablespaces
V$DATATYPE	Information about data types supported by ALTIBASE HDB
V$DBA_2PC_PENDING	A list of distributed transactions whose status is “in-doubt”
V$DBLINK_REMOTE_STATEMENT_INFO	Information about statements that are executed on the remote server when using Database Link
V$DBLINK_REMOTE_TRANSACTION_INFO	Information about transactions that occur on the remote server when using Database Link
V$DBLINK_TRANSACTION_INFO	Transaction information used by Database Link
V$DB_FREEPAGELISTS	Information about all usable page lists
V$DB_PROTOCOL	Information about database protocols input into the server
V$DISKTBL_INFO	Information on disk tables
V$DISK_BTREE_HEADER	Information about headers of disk BTREE indexes
V$DISK_RTREE_HEADER	Information about headers of disk RTREE indexes
V$DISK_UNDO_USAGE	Information about the amount of undo tablespace on disk that is currently being used
V$EVENT_NAME	Information about ALTIBASE HDB server wait events
V$FILESTAT	Statistical information about disk data file I/O
V$FLUSHER	Information about the flusher which flushes the buffers
V$FLUSHINFO	Buffer flush information
V$INDEX	Information about table indexes
V$INSTANCE	Information about the current startup phase
V$LATCH	Information about the Buffer Control Block (BCB) latch of the buffer pool and statistical information about read/write latch attempts made on data pages
V$LFG	Information about LFG and statistical information related to GROUP COMMIT

Name	Description
V$LINKER_STATUS	Information about the status of AltiLinker for Database Link
V$LOCK	Information about all table level lock nodes in the database at the current point in time
V$LOCK_STATEMENT	Information about locks and statements, shown together
V$LOCK_WAIT	Information about the status of transactions waiting to obtain locks
V$LOG	Information on log anchor files
V$MEMGC	Information about garbage collection (memory space recovery)
V$MEMSTAT	Statistical information about memory use by ALTIBASE HDB processes
V$MEMTBL_INFO	Information about memory tables
V$MEM_BTREE_HEADER	Information about headers of memory BTREE indexes
V$MEM_BTREE_NODEPOOL	Information about node pools for memory BTREE Indices
V$MEM_RTREE_HEADER	Information about headers of memory RTREE indexes
V$MEM_RTREE_NODEPOOL	Information about node pools for memory RTREE indexes
V$MEM_TABLESPACES	Information about tablespaces created in memory
V$MEM_TABLESPACE_CHECKPOINT_PATHS	Information about the location of DB files in which to record checkpointing details during checkpointing
V$MEM_TABLESPACE_STATUS_DESC	Internal information about the status of memory tablespaces
V$MUTEX	Statistical information about mutexes, used by ALTIBASE HDB for concurrency control
V$NLS_PARAMETERS	Information about parameters related to NLS
V$PLANTEXT	Information about SQL execution plan text
V$PROCTEXT	Information about stored procedure text
V$PROPERTY	Information about internally set ALTIBASE HDB properties
V$REPEXEC	Information about the replication manager
V$REPGAP	Information about the difference between the log record currently being processed by the replication Sender and the most recently created log record
V$REPGAP_PARALLEL	Information about the difference between the sequence number of the log record currently being processed by replication sender threads working in parallel and the sequence number of the most recently created log record
V$REPLOGBUFFER	Information about the log buffer used for replication

Name	Description
V$REPOFFLINE_STATUS	Information about the status of offline replication execution
V$REPRECEIVER	Information about the replication Receiver
V$REPRECEIVER_COLUMN	Information about target columns for the replication Receiver
V$REPRECEIVER_PARALLEL	Information about replication Receiver threads working in parallel
V$REPRECEIVER_TRANSTBL	Information about transaction tables for the replication Receiver
V$REPRECEIVER_TRANSTBL_PARALLEL	Information about transaction tables used by replication Receiver threads working in parallel
V$REPRECOVERY	Recovery information used in replication
V$REPSENDER	Information about the replication Sender
V$REPSENDER_PARALLEL	Information about replication Sender threads working in parallel
V$REPSENDER_TRANSTBL	Information about transaction tables used by the replication Sender
V$REPSENDER_TRANSTBL_PARALLEL	Information about transaction tables used by replication Sender threads working in parallel
V$REPSYNC	Information about tables that are synchronized using replication
V$SEGMENT	Information about segments, which constitute tables and indexes
V$SEQ	Sequence-related information
V$SERVICE_THREAD	Information about service threads related to multiplexing
V$SESSION	Information about client sessions created internally in ALTIBASE HDB
V$SESSION_EVENT	Statistical information on all wait events for all currently connected sessions
V$SESSION_WAIT	Information about wait events for all currently connected sessions
V$SESSION_WAIT_CLASS	Cumulative wait statistic information classified by session, wait event and wait class for all currently connected sessions.
V$SESSIONMGR	Statistical information about ALTIBASE HDB sessions
V$SESSTAT	Information about the status of currently connected sessions
V$SQLTEXT	Information about the text of all SQL statements executed in the system

Name	Description
V$SQL_PLAN_CACHE	Information about the current status and statistical information about the SQL Plan Cache
V$SQL_PLAN_CACHE_PCO	Information about Plan Cache objects registered in the SQL Plan Cache
V$SQL_PLAN_CACHE_SQLTEXT	Information about SQL statements registered in the SQL Plan Cache
V$STABLE_MEM_DATAFILES	Information about the paths of data file(s)
V$STATEMENT	Information about statements for all current ALTIBASE HDB sessions
V$STATNAME	Information about the name and status of the system and sessions
V$ST_ANGULAR_UNIT	Reserved for future use
V$ST_AREA_UNIT	Reserved for future use
V$ST_LINEAR_UNIT	Reserved for future use
V$SYSSTAT	Information about the status of the system
V$SYSTEM_CONFLICT_PAGE	Information about latch contention according to page type
V$SYSTEM_EVENT	Cumulative statistical information about waits from startup to the current time, classified according to wait event
V$SYSTEM_WAIT_CLASS	Cumulative statistical information about waits from startup to the current time, classified according to wait class
V$TABLE	Information about records and columns for all performance views
V$TABLESPACES	Information about tablespaces
V$TRACELOG	Information about trace logging
V$TRANSACTION	Information about transaction objects
V$TRANSACTION_MGR	Information about the transaction manager of ALTIBASE HDB
V$TSSEGS	Information about all TSS segments
V$TXSEGS	Information about bound transaction segments
V$UDSEGS	Information about all undo segments
V$UNDO_BUFF_STAT	Statistical Information about the undo tablespace buffer pool
V$VERSION	ALTIBASE HDB version information
V$WAIT_CLASS_NAME	Information for grouping wait events into classes
V$VOL_TABLESPACES	Information about volatile tablespaces

Name	Description
V$XID	List of XIDs, which are branches of distributed transactions, that currently exist in the DBMS

3.2.4 V$ALLCOLUMN

This view displays information about the columns in all performance views.

Column	Data Type	Description
TABLENAME	VARCHAR(39)	The name of the performance view
COLNAME	VARCHAR(39)	The name of the column in the performance view

3.2.4.1 Column Information

TABLENAME

This is the name of the performance view.

COLNAME

This is the name of the column in the performance view.

3.2.5 V$ARCHIVE

This view displays the information related to archiving and backups.

Column	Data Type	Description
LFG_ID	INTEGER	The log file group identifier
ARCHIVE_MODE	BIGINT	Archive log mode 0: no archive log mode 1: archive log mode
ARCHIVE_THR_RUNNING	BIGINT	Information about the execution of the archivelog thread
ARCHIVE_DEST	VARCHAR(1024)	The directory in which logs are to be archived
NEXTLOGFILE_TO_ARCH	INTEGER	The number of the next log file to be archived
OLDEST_ACTIVE_LOGFILE	INTEGER	The number of the oldest of the online log files

Column	Data Type	Description
CURRENT_LOGFILE	INTEGER	The number of the current online log file

3.2.5.1 Column Information

LFG_ID

There is one archive directory for each Log File Group (LFG). This is the identifier of the LFG.

ARCHIVE_MODE

This indicates the archive log mode of the database.

0: No archive log mode

1: Archive log mode

3.2.6 V$BUFFPAGEINFO

This view shows statistics about the main operations managed by the buffer manager for each type of page in the buffer frame.

Column	Data Type	Description
PAGE_TYPE	VARCHAR(20)	The type of page
READ_PAGE_COUNT	BIGINT	The number of times that disk I/O (READ) was initiated
GET_PAGE_COUNT	BIGINT	The number of times that buffer frames have been requested
FIX_PAGE_COUNT	BIGINT	The number of times that buffer frames have been fixed
CREATE_PAGE_COUNT	BIGINT	The number of times that new buffer frames have been requested
HIT_RATIO	DOUBLE	The buffer frame hit ratio

3.2.6.1 Column Information

PAGE_TYPE

PAGE_TYPE indicates the type of buffer page. The possible values are as follows:

PAGE_TYPE	Description
PAGE UNFORMAT	An unformatted page
PAGE FORMAT	A formatted page
PAGE INDEX META BTREE	A page in which meta information about a B-Tree index is written
PAGE INDEX META RTREE	A page in which meta information about an R-Tree index is written
PAGE INDEX BTREE	A page in which a B-Tree index node is written
PAGE INDEX RTREE	A page in which an R-Tree index node is written
PAGE TABLE	A page in which table records are written
PAGE TEMP TABLE META	A page in which meta information about a single temporary table is written
PAGE TEMP TABLE DATA	A page in which the records stored in a temporary table are written
PAGE TSS	A page in which information about the status of a transaction is written. Multiple transaction status slots (TSS) can be written to a single page.
PAGE UNDO	A page in which UNDO information is written. A single page can contain multiple UNDO records.
PAGE LOB DATA	A page in which LOB type data are written. A single page cannot contain more than one LOB column. Moreover, a single LOB column can span multiple pages.
PAGE LOB INODE	A page in which an index node, which pertains to LOB data that exceed a certain size, is written
PAGE FMS SEGHDR	A page in which a single FMS header is written
PAGE FMS EXTDIR	A page in which a single FMS extent directory is written
PAGE TMS SEGHDR	A page in which a single TMS header is written
PAGE TMS LFBMP	A page in which a single TMS leaf bitmap node is written
PAGE TMS ITBMP	A page in which a single TMS internal bitmap node is written
PAGE TMS RTBMP	A page in which a single TMS root bitmap node is written
PAGE TMS EXTDIR	A page in which a single TMS extent directory is written
PAGE CMS SEGHDR	A page in which a single CMS header is written
PAGE CMS EXTDIR	A page in which a single CMS extent directory is written
PAGE FEBT FSB	A page in which a single datafile header is written
PAGE FEBT EGH	A page in which an extent group header within a data file is written. One page can contain only one header.
PAGE LOB META	A page in which meta information about a LOB data column is written

PAGE_TYPE	Description
PAGE HV TEMP NODE	A page in which a node of a Hash Value-Based Temp Index is written

READ_PAGE_COUNT

This is the total number of disk I/O (read) requests that have been made for buffer frames related to this PAGE_TYPE since the server was started.

The value can be 0 or greater.

GET_PAGE_COUNT

Shows the total number of read or write requests that have been made to the buffer manager for buffer frames related to this PAGE_TYPE since the server was started.

The value can be 0 or greater.

FIX_PAGE_COUNT

This shows the total number of fixes for buffer frames related to PAGE_TYPE received by the buffer manager for reading or writing data since the server was started. The value can be 0 or greater.

CREATE_PAGE_COUNT

This shows the number of requests for new buffer frames for this PAGE_TYPE made to the buffer manager since the server was started.

The value can be 0 or greater.

HIT_RATIO

This shows the hit ratio for this buffer since the server was started. Its value can be calculated as follows: (GET_PAGE_COUNT + FIX_PAGE_COUNT - READ_PAGE_COUNT) / (GET_PAGE_COUNT + FIX_PAGE_COUNT)

3.2.6.2 Example

The following SQL shows how to retrieve v$buffpageinfo and cumulative figures of main operations for each page type managed in the buffer since the server was started.

```
iSQL> select * from v$buffpageinfo;
PAGE_TYPE             READ_PAGE_COUNT      GET_PAGE_COUNT
---------------------------------------------------------------------
FIX_PAGE_COUNT        CREATE_PAGE_COUNT    HIT_RATIO
---------------------------------------------------------------------
PAGE UNFORMAT           0                    0
0                     0                    0
PAGE FORMAT             0                    0
0                     0                    0
PAGE INDEX META BTREE   4                    0
4                     0                    0
PAGE INDEX META RTREE   0                    0
0                     0                    0
PAGE INDEX BTREE        12                   0
```

```
12                    0                    0
PAGE INDEX RTREE        0                    0
0                     0                    0
PAGE TABLE              0                    0
0                     0                    0
PAGE TEMP TABLE META    0                    0
0                     0                    0
PAGE TEMP TABLE DATA    0                    0
0                     0                    0
PAGE TSS                0                    0
0                     0                    0
PAGE UNDO               0                    0
0                     0                    0
PAGE LOB DATA           0                    0
0                     0                    0
PAGE LOB INODE          0                    0
0                     0                    0
PAGE FMS SEGHDR         0                    0
0                     0                    0
PAGE FMS EXTDIR         0                    0
0                     0                    0
PAGE TMS SEGHDR         5                    19
4                     0                    73.6842105263158
PAGE TMS LFBMP          0                    0
0                     0                    0
PAGE TMS ITBMP          0                    0
0                     0                    0
PAGE TMS RTBMP          0                    0
0                     0                    0
PAGE TMS EXTDIR         0                    0
0                     0                    0
PAGE CMS SEGHDR         0                    1536
0                     512                  100
PAGE CMS EXTDIR         0                    0
0                     0                    0
PAGE FEBT FSB           2                    1024
515                   2                    99.8046875
PAGE FEBT EGH           0                    512
0                     4                    100
PAGE LOB META           0                    0
0                     0                    0
PAGE HV TEMP NODE       0                    0
0                     0                    0
26 rows selected.
```

3.2.7 V$BUFFPOOL_STAT

This view displays statistics including the buffer pool hit ratio and the buffer control block (BCB) count of the buffer pool.

Column	Data Type	Description
ID	INTEGER	The identifier of the buffer pool
POOL_SIZE	INTEGER	The number of pages in the buffer pool
PAGE_SIZE	INTEGER	The size of a page (in bytes)
HASH_BUCKET_COUNT	INTEGER	The number of hash table buckets

Column	Data Type	Description
HASH_CHAIN_LATCH_COUNT	INTEGER	The number of chain latches used in the hash table of the buffer pool
LRU_LIST_COUNT	INTEGER	The number of LRU lists
PREPARE_LIST_COUNT	INTEGER	The number of prepare lists in the buffer pool
FLUSH_LIST_COUNT	INTEGER	The number of flush lists in the buffer pool
CHECKPOINT_LIST_COUNT	INTEGER	The number of checkpoint lists in the buffer pool
VICTIM_SEARCH_COUNT	INTEGER	The number of victim searches in an LRU List
HASH_PAGES	INTEGER	The number of pages inserted into the hash table at present
HOT_LIST_PAGES	INTEGER	The number of pages in LRU hot lists at present
COLD_LIST_PAGES	INTEGER	The number of pages in LRU cold lists at present
PREPARE_LIST_PAGES	INTEGER	The number of pages in all prepare lists at present
FLUSH_LIST_PAGES	INTEGER	The number of pages in all flush lists at present
CHECKPOINT_LIST_PAGES	INTEGER	The number of pages in all checkpoint lists at present
FIX_PAGES	BIGINT	The accumulated number of page fix requests without latches
GET_PAGES	BIGINT	The accumulated number of page requests for which latches were obtained
READ_PAGES	BIGINT	The accumulated number of page reads from disk
CREATE_PAGES	BIGINT	The accumulated number of new page creation tasks
HIT_RATIO	DOUBLE	The cumulative hit ratio from the buffer pool since the system was started
HOT_HITS	BIGINT	The accumulated number of accesses to an LRU hot list
COLD_HITS	BIGINT	The accumulated number of accesses to an LRU cold list
PREPARE_HITS	BIGINT	The accumulated number of accesses to a prepare list

Column	Data Type	Description
FLUSH_HITS	BIGINT	The accumulated number of accesses to a flush list
OTHER_HITS	BIGINT	The accumulated number of accesses to buf-fers not included on any list
PREPARE_VICTIMS	BIGINT	The accumulated number of searches for replacement targets on a prepare list
LRU_VICTIMS	BIGINT	The accumulated number of searches for replacement targets on an LRU list
VICTIM_FAILS	BIGINT	The number of failures to find a replacement target
PREPARE_AGAIN_VICTIMS	BIGINT	The cumulative number of searches for a replacement target buffer on a prepare list after failing to find a replacement target on an LRU list
VICTIM_SEARCH_WARP	BIGINT	The number of searches that continued to subsequent prepare lists after failing to find replacement targets on prepare lists or LRU lists
LRU_SEARCHS	BIGINT	The accumulated number of searched buf-fers on an LRU list
LRU_SEARCHS_AVG	INTEGER	The average number of buffers searched for a replacement target
LRU_TO_HOTS	BIGINT	The accumulated number of times that a Buf-fer Control Block (BCB) has moved into a hot area in an LRU list
LRU_TO_COLDS	BIGINT	The accumulated number of times that a BCB has moved into a cold area in an LRU list
LRU_TO_FLUSHS	BIGINT	The accumulated number of times that a BCB has moved from an LRU list to a flush list
HOT_INSERTIONS	BIGINT	The accumulated number of insertions into LRU hot lists
COLD_INSERTIONS	BIGINT	The accumulated number of insertions into LRU cold lists

3.2.7.1 Column Information

ID

This is a unique buffer pool number. Its value is 0 because multiple buffer pools are not currently supported.

POOL_SIZE

This is the number of pages in the buffer pool. POOL_SIZE * PAGE_SIZE is equal to the size specified by the BUFFER_AREA_SIZE property.

PAGE_SIZE

This is the size of the pages used in the buffer pool at present. Only the fixed value 8192 is possible, because multiple buffer pools are not currently supported.

HASH_BUCKET_COUNT

This is the number of hash table buckets. It is determined by the BUFFER_HASH_BUCKET_DENSITY property. This value cannot be changed while the server is running. The greater this value is, the less expensive it is to search the hash bucket list.

HASH_CHAIN_LATCH_COUNT

This is the number of chain latches used in the hash table. The greater this value is, the less competition there is for latches, which can occur when searching the hash table.

LRU_LIST_COUNT

This is the number of LRU lists in the buffer pool.

PREPARE_LIST_COUNT

This is the number of prepare lists in the buffer pool.

FLUSH_LIST_COUNT

This is the number of flush lists in the buffer pool.

CHECKPOINT_LIST_COUNT

This is the number of checkpoint lists in the buffer pool.

VICTIM_SEARCH_COUNT

This is the maximum number of BCBs that are searched when searching for replacement targets in LRU lists. If the search for replacement targets reaches the specified value and no replacement target is found, Buffer Manager waits until the flusher adds a clean buffer to the prepare list.

HASH_PAGES

This is the number of buffers that have been inserted into the hash table. Its value indicates the number of buffers currently in use.

HOT_LIST_PAGES

This is the number of buffers that exist on the LRU hot list.

COLD_LIST_PAGES

This is the number of buffers that exist on the LRU cold list.

PREPARE_LIST_PAGES

This is the number of buffers that exist on the prepare list. If the value is 0, the LRU list is searched in order to obtain replacement targets.

FLUSH_LIST_PAGES

This is the number of buffers that exist on the flush list. A high value means that there are many buffers to be flushed.

CHECKPOINT_LIST_PAGES

This is the number of buffers that exist on the checkpoint list. It also indicates the number of pages that have been renewed.

FIX_PAGES

This is the cumulative number of pages that have been requested without obtaining latches since the system was started.

GET_PAGES

This is the cumulative number of page latches that have been have been requested and obtained since the system was started.

READ_PAGES

This is the cumulative number of pages that have been read from disk when requesting a page. It also indicates the number of buffer misses.

CREATE_PAGES

This is the cumulative number of page assignments for the insertion of data into new pages. Page creation isn't actually accompanied by disk I/O.

HIT_RATIO

This is the cumulative hit ratio in the buffer pool. It can be calculated thus: (GET_PAGES + FIX_PAGES - READ_PAGES)/(GET_PAGES + FIX_PAGES). If this value is low, it means that many pages have been read from disk instead of from the cache. In other words, if the value is low, the system will not be able to process queries quickly.

HOT_HITS

This is the cumulative number of hits on the LRU hot list. If a requested page is already in the buffer, a hit doesn't cause a page to be read.

COLD_HITS

This is the cumulative number of hits on the LRU cold list.

PREPARE_HITS

This is the cumulative number of hits on the prepare list.

FLUSH_HITS

This is the cumulative number of hits on the flush list.

OTHER_HITS

This is the number of hits on a buffer that was not on any list at that moment. A hit buffer need not always be on a list.

PREPARE_VICTIMS

This is the cumulative number of searches for replacement buffers on a prepare list.

LRU_VICTIMS

This is the cumulative number of searches for replacement buffers on an LRU list.

VICTIM_FAILS

This is the cumulative number of failures to find a replacement target buffer. This value can be calculated thus: PREPARE_AGAIN_VICTIMS + VICTIM_SEARCH_WARP.

Summing PREPARE_VICTIMS + LRU_VICTIMS + VICTIM_FAILS gives the total number of replacements in the buffer pool.

PREPARE_AGAIN_VICTIMS

After failing to find replacement target buffers, it is necessary to wait for the insertion of buffers on a prepare list. While waiting, this is the number of clean buffers that have been received and selected as replacement targets.

VICTIM_SEARCH_WARP

This is the cumulative number of searches for replacement target buffers that failed after the specified period of time and thus passed to the next prepare list.

LRU_SEARCHS

This is the cumulative number of buffers for which searches for replacement target buffers have been made in the LRU list.

LRU_SEARCHS_AVG

This is the average number of buffers that are searched when searching for a replacement target.

LRU_TO_HOTS

This is the cumulative number of times that buffers have moved into hot areas in LRU lists.

LRU_TO_COLDS

This is the cumulative number of times that buffers have moved into cold areas in LRU lists.

LRU_TO_FLUSHS

This is the cumulative number of times that buffers have moved from LRU lists to flush lists.

HOT_INSERTIONS

This is the cumulative number of insertions into LRU hot lists.

COLD_INSERTIONS

This is the cumulative number of insertions into LRU cold lists.

3.2.8 V$CATALOG

This view displays information about the structure of the tables that exist in the database.

Column	Data Type	Description
TABLE_OID	BIGINT	The object identifier of the table
COLUMN_CNT	INTEGER	The number of columns in the table
COLUMN_VAR_SLOT_C NT	INTEGER	The number of variable slots, which are used to store information about columns
INDEX_CNT	INTEGER	The number of indexes in the table
INDEX_VAR_SLOT_CNT	INTEGER	The number of variable slots, which are used to store information about indexes

3.2.8.1 Column Information

TABLE_OID

This is the physical location of the header, which contains information about the table.

COLUMN_CNT

This is the number of columns in the table.

COLUMN_VAR_SLOT_CNT

This is the number of variable slots, which are used to store information about the columns in the table.

INDEX_CNT

This is the number of indexes in the table.

INDEX_VAR_SLOT_CNT

This is the number of variable slots, which are used to store information about the indexes in the table.

3.2.9 V$DATABASE

V$DATABASE displays internal information about the memory database.

Column	Data Type	Description
DB_NAME	VARCHAR(128)	The database name
PRODUCT_SIGNATURE	VARCHAR(512)	A string describing the product binary and build environment
DB_SIGNATURE	VARCHAR(512)	A unique database identification string
VERSION_ID	INTEGER	The version of the database
COMPILE_BIT	INTEGER	Whether the product was compiled for 32 bits or 64 bits
ENDIAN	BIGINT	Endian information
LOGFILE_SIZE	BIGINT	The log file size
TX_TBL_SIZE	INTEGER	The transaction table size
LAST_SYSTEM_SCN	VARCHAR(29)	For internal usage only
INIT_SYSTEM_SCN	VARCHAR(29)	For internal usage only
DURABLE_SYSTEM_SCN	VARCHAR(29)	The saved system SCN value
MEM_MAX_DB_SIZE	VARCHAR(256)	The maximum size of the memory database
MEM_ALLOC_PAGE_COUNT	BIGINT	The total number of allocated pages
MEM_FREE_PAGE_COUNT	BIGINT	The total number of available pages
MAX_ACCESS_FILE_SIZE	VARCHAR(12)	The maximum file size that can be created in the database

3.2.9.1 Column Information

DB_NAME

This is the name of the memory database.

PRODUCT_SIGNATURE

This is unique information of ALTIBASE HDB.

DB_SIGNATURE

A unique database identification string.

VERSION_ID

This is a unique version number managed by the storage manager of ALTIBASE HDB.

COMPILE_BIT

This indicates whether the database was compiled as a 32-bit or 64-bit application.

ENDIAN

This is the Endian of the database.

0: little Endian

1: big Endian

LOGFILE_SIZE

This is the size of the log files used by the database.

TX_TBL_SIZE

This is the size of the transaction table.

MEM_MAX_DB_SIZE

This is the maximum size to which the memory database can expand.

MEM_ALLOC_PAGE_COUNT

This is the total number of pages currently allocated to the memory database. This only indicates the current size of memory database space, not the maximum size to which it can expand. The current size of memory database space can be calculated by multiplying the sum of MEM_ALLOC_PAGE_COUNT and MEM_FREE_PAGE_COUNT by the page size (32kB).

MEM_FREE_PAGE_COUNT

This is the number of pages available to be allocated to memory database space, not including the number of pages that are currently allocated. This only pertains to the current size of memory database space, not the maximum size to which it can expand. The current size of memory database space can be calculated by multiplying the sum of MEM_ALLOC_PAGE_COUNT and MEM_FREE_PAGE_COUNT by the page size (32kB).

DURABLE_SYSTEM_SCN

This is the system SCN value saved in database.

3.2.10 V$DATAFILES

This view displays information about the data files used in tablespaces.

Column	Data Type	Description
ID	INTEGER	The data file identifier
NAME	VARCHAR(256)	Data file name
SPACEID	INTEGER	The tablespace identifier
OLDEST_LSN_LFGID	INTEGER	See below
OLDEST_LSN_FILENO	INTEGER	See below
OLDEST_LSN_OFFSET	INTEGER	See below
CREATE_LSN_LFGID	INTEGER	See below
CREATE_LSN_FILENO	INTEGER	See below
CREATE_LSN_OFFSET	INTEGER	See below
SM_VERSION	INTEGER	Version information
NEXTSIZE	BIGINT	The size at the next increase
MAXSIZE	BIGINT	The maximum size
INITSIZE	BIGINT	The initial size
CURRSIZE	BIGINT	The current size
AUTOEXTEND	INTEGER	An auto-extension flag
IOCOUNT	INTEGER	The number of I/O operations currently underway
OPENED	INTEGER	Indicates whether or not the file is currently in use
MODIFIED	INTEGER	Indicates whether or not the file is currently being modified
STATE	INTEGER	The status of the file

Column	Data Type	Description
MAX_OPEN_FD_COUNT	INTEGER	The maximum number of FDs that can be opened
CUR_OPEN_FD_COUNT	INTEGER	The number of open FDs

3.2.10.1 Column Information

ID

This is the identifier of the data file. In order to avoid duplicate identifiers, identifiers are assigned sequentially in the order in which data files are created.

NAME

This is the physical path and name of the data file.

SPACEID

This is the identifier of the tablespace containing the data file.

OLDEST_LSN_LFGID

This is the Log File Group (LFG) portion of the LSN value of the oldest of the pages that were loaded into the buffer and changed at the time of the last checkpoint, when pages in the data file were flushed to disk.

OLDEST_LSN_FILENO

This is the file number portion of the LSN value of the oldest of the pages that were loaded into the buffer and changed at the time of the last checkpoint, when pages in the data file were flushed to disk.

OLDEST_LSN_OFFSET

This is the offset value portion of the LSN value of the oldest of the pages that were loaded into the buffer and changed at the time of the last checkpoint, when pages in the data file were flushed to disk.

CREATE_LSN_LFGID

This is the identifier of the Log File Group (LFG) of the LSN that was current at the time at which the data file was created.

CREATE_LSN_FILENO

This is the file number portion of the LSN that was current at the time at which the data file was created.

CREATE_LSN_OFFSET

This is the offset value portion of the LSN that was current at the time at which the data file was created.

SM_VERSION

This is the version of the binary from which the data file was created.

NEXTSIZE

If the data file's `autoextend` property is set to "on", this is the size by which the data file will be increased when there is insufficient space.

MAXSIZE

If the data file's `autoextend` property is set to "on", this is the maximum size to which the data file can be increased when there is insufficient space.

INITSIZE

This is the initial size of the data file at the time of its creation.

CURRSIZE

This is the current size of the data file.

AUTOEXTEND

This indicates whether the size of the data file will be increased automatically when there is insufficient space.

0: No automatic increase

1: Automatic increase

IOCOUNT

This is the number of I/O operations currently underway on the data file. If no data I/O is in progress on the data file, the next data file can be opened.

OPENED

This indicates whether the data file is currently open.

0: closed

1: open

MODIFIED

This indicates whether the data file has been modified. If any pages have been flushed to the data file without subsequent synchronization, this value is 1. if synchronization has been executed on the data file since pages were last flushed to it, this value is 0.

STATE

This is the status of the data file.

1: Offline

2: Online

6: Backup is in progress

128: Dropped

MAX_OPEN_FD_COUNT

This is the maximum number of FDs (File Descriptors) that can be opened when performing I/O on the current disk data file.

CUR_OPEN_FD_COUNT

This is the number of open FDs (File Descriptors) for the current disk data file.

3.2.11 V$DATATYPE

This table shows information about the data types that are supported by ALTIBASE HDB.

Column name	Type	Description
TYPE_NAME	VARCHAR(40)	The name of a data type that is supported in the DBMS
DATA_TYPE	SMALLINT	An internally defined value indicating a data type that is supported in the DBMS
ODBC_DATA_TYPE	SMALLINT	The identifier of an ODBC SQL data type corresponding to the data type
COLUMN_SIZE	INTEGER	The maximum column size for the data type
LITERAL_PREFIX	VARCHAR(4)	Characters recognized as the prefix of the data type literal
LITERAL_SUFFIX	VARCHAR(4)	Characters recognized as the suffix of the data type literal
CREATE_PARAM	VARCHAR(20)	When using SQL to define a data type, a parameter keyword list enclosed in parentheses
NULLABLE	SMALLINT	Indicates whether NULL values are allowed for the data type

Column name	Type	Description
CASE_SENSITIVE	SMALLINT	Indicates whether the data type is case-sensitive
SEARCHABLE	SMALLINT	Indicates how the data type is used in a WHERE clause
UNSIGNED_ATTRIBUTE	SMALLINT	For a numeric data type, indicates whether the data type is a signed data type
FIXED_PREC_SCALE	SMALLINT	Indicates whether the data type is a fixed type
AUTO_UNIQUE_VALUE	SMALLINT	Reserved for future use
LOCAL_TYPE_NAME	VARCHAR(40)	The name of the data type in the local language
MINIMUM_SCALE	SMALLINT	The minimum allowable number of digits to the right of the decimal point
MAXIMUM_SCALE	SMALLINT	The maximum allowable number of digits to the right of the decimal point
SQL_DATA_TYPE	SMALLINT	(A defined value of an SQL data type that is provided by SQL_DESC_TYPE in ODBC)
SQL_DATETIME_SUB	SMALLINT	A type subcode for a `datetime` or `interval` data type
NUM_PREC_RADIX	INTEGER	The number of bits that are needed to perform operations on the maximum number of digits that a column can hold
INTERVAL_PRECISION	SMALLINT	When the DATA_TYPE is interval, the maximum number of digits needed to express the data

3.2.11.1 Column Information

ODBC_DATA_TYPE

This is the data type identifier for the ODBC SQL data type corresponding to the data type. For more information, please refer to the appendix pertaining to data types in the *ODBC Reference*.

COLUMN_SIZE

This is the maximum column size for the data type.

For numeric data types, this is the precision value, which was specified when the type was defined. For string data types, this is the length value, which was specified when the type was defined. For datetime data types, this is the total number of characters that are needed to display a value when it is converted to characters.

LITERAL_PREFIX

This is the characters that signify the prefix of a literal for the data type. For data types to which literal prefixes do not apply, it is NULL.

LITERAL_SUFFIX

This is the characters that signify the suffix of a literal for the data type. For data types to which literal suffixes do not apply, it is NULL.

CREATE_PARAM

When using SQL to define a data type, this is a comma-separated list of parameter keywords enclosed in parentheses. For example, to express a `NUMBER` as NUMBER(precision,scale), the content within the parentheses, that is, "precision, scale", is the list. "Precision" and "scale" are thus both keywords in the list. For data types that do not need parameters, this is set to NULL.

NULLABLE

This indicates whether NULL values are allowed for a data type.

1: NULL is allowed.

0: NULL is not allowed.

CASE_SENSITIVE

For character data types, indicates whether to distinguish between uppercase and lowercase letters when sorting data of the data type.

1: Case-sensitive.

0: Not case-sensitive.

SEARCHABLE

Indicates how a data type can be used in a WHERE clause.

0: It cannot be used in a WHERE clause (SQL_PRED_NONE).

1: It can be used in a WHERE clause, but must be used with LIKE (SQL_PRED_CHAR).

2: It can be used in a WHERE clause with any comparison operator except LIKE (SQL_PRED_BASIC).

3: It can be used in a WHERE clause with any comparison operator (SQL_SEARCHABLE).

UNSIGNED_ATTRIBUTE

Indicates whether a data type is signed.

1: The data type is an unsigned data type.

0: The data type is a signed data type.

NULL: The data type is not numeric, therefore this attribute is not applicable.

FIXED_PREC_SCALE

Indicates whether a data type is fixed. If a data type is a fixed numeric type and always has the same precision and scale, this value is 1 (SQL_TRUE). Otherwise, it is 0 (SQL_FALSE).

LOCAL_TYPE_NAME

Indicates a localized (region-specific) name for a data type. If there is no localized name, this value is NULL.

MINIMUM_SCALE

For numeric data types, this is the minimum allowable number of digits to the right of the decimal. This value exists for fixed scale types; it is set to NULL for types to which scale does not pertain.

MAXIMUM_SCALE

For numeric data types, this is the maximum allowable number of digits to the right of the decimal. It is specified when the data type is defined. It is set to NULL for types to which scale does not pertain.

SQL_DATA_TYPE

This is a SQL data type that is provided by SQL_DESC_TYPE in ODBC. For data types other than INTERVAL or DATETIME, this value is the same as that of ODBC_DATA_TYPE.

SQL_DATETIME_SUB

If the SQL_DATA_TYPE value is SQL_DATETIME or SQL_INTERVAL, this is the type sub code for the DATETIME or INTERVAL data type. If the data type is not DATETIME or INTERVAL, it is set to NULL.

NUM_PREC_RADIX

This is the number of bits or digits that are needed to perform mathematical operations on the highest number that a column can hold.

INTERVAL_PRECISION

This is the maximum number of digits that a DATA_TYPE of type INTERVAL can hold.

3.2.12 V$DBA_2PC_PENDING

This view shows a list of XIDs (transaction IDs) for distributed transactions that exist in the DBMS and whose status is in doubt. The status of a distributed transaction is said to be "in-doubt" when a branch thereof is ready to be committed, but has not yet been committed or rolled back.

Column name	Type	Description
LOCAL_TRAN_ID	BIGINT	An internal ALTIBASE HDB transaction identifier that is associated with the GLOBAL_TX_ID
GLOBAL_TX_ID	VARCHAR(256)	Globally unique transaction identifier

3.2.12.1 Column Information

LOCAL_TRAN_ID

This is an internal ALTIBASE HDB transaction identifier that is associated with a global transaction identifier.

GLOBAL_TX_ID

This is globally unique transaction identifier. The GLOBAL_TX_ID contains a format identifier, two length fields and a data field. The data field comprises at most two contiguous components: a global transaction identifier and a branch qualifier.

3.2.13 V$DBLINK_REMOTE_STATEMENT_INFO

This view shows information about a query statement that is parsed and executed on a remote server when Database Link is used.

Column name	Type	Description
TRANSACTION_ID	INTEGER	The identifier of the transaction that uses Database Link
REMOTE_TRANSACTION_ID	INTEGER	The identifier of a transaction that took place on a remote server
STATEMENT_ID	INTEGER	The identifier of a statement that is executed on a remote server
QUERY	VARCHAR(1024)	A query that is executed in a statement

3.2.13.1 Column Information

REMOTE_TRANSACTION_ID

This is the identifier of a transaction that takes place on a remote server. This identifier is not the actual identifier of the transaction on the remote server; it is an identifier that is assigned by AltiLinker when a transaction is created on a remote server. Since this identifier is created for administrative purposes, the value itself is not meaningful.

STATEMENT_ID

This is the identifier of a statement that is executed on a remote server. This identifier is not the actual identifier of the statement on the remote server; it is an identifier that is self-assigned by AltiLinker when a statement is created on a remote server. Since this identifier is created for administrative purposes, the value itself is not meaningful.

3.2.14 V$DBLINK_REMOTE_TRANSACTION_INFO

This view shows information about a transaction that takes place on a remote server when Database Link is used:.

Column name	Type	Description
TRANSACTION_ID	INTEGER	The identifier of a local transaction that uses Database Link
REMOTE_TRANSACTION_ID	INTEGER	The identifier of a transaction that occurs on a remote server
CONNECTION_METHOD	INTEGER	0: ODBC 1: Native (reserved for future use)
CONNECTION_STRING	VARCHAR(41)	A connection string
ACTIVE_STATEMENT_COUNT	INTEGER	The number of query statements that are currently being executed

3.2.14.1 Column Information

REMOTE_TRANSACTION_ID

This is the identifier of a transaction that takes place on a remote server. This identifier is not the actual identifier of the transaction on the remote server; it is an identifier that is self-assigned by AltiLinker when the transaction is created on the remote server. Since this identifier is created for administrative purposes, the value itself is not meaningful.

3.2.15 V$DBLINK_TRANSACTION_INFO

This view shows information of a transaction that uses the current Database Link:

Column name	Type	Description
TRANSACTION_ID	INTEGER	The identifier of a transaction that is currently using Database Link
STATUS	INTEGER	Reserved for future use
CONSISTENCY	INTEGER	Reserved for future use

3.2.16 V$DB_FREEPAGELISTS

This view displays information about lists of pages that can be used, that is, free pages, in a database.

Column	Data Type	Description
SPACE_ID	INTEGER	The identifier of the tablespace to which the free pages belong
RESOURCE_GROUP_ID	INTEGER	The identifier of the resource group
FIRST_FREE_PAGE_ID	INTEGER	The identifier of the first free page in the list
FREE_PAGE_COUNT	BIGINT	The total number of free pages in the list

3.2.16.1 Column Information

RESOURCE_GROUP_ID

This is a unique number that is used to identify the list.

FIRST_FREE_PAGE_ID

This is the identifier of the first free page in the list.

FREE_PAGE_COUNT

This is the number of free pages on the list.

3.2.17 V$DB_PROTOCOL

This view shows information on ALTIBASE HDB communication protocols of all incoming packets.

Column name	Type	Description
QP_NAME	VARCHAR(50)	The protocol name
QP_ID	INTEGER	The unique identifier of the protocol
COUNT	BIGINT	The cumulative number of incoming packets for this protocol

3.2.18 V$DIRECT_PATH_INSERT

This view displays historical statistics on direct-path uploads.

Column	Data Type	Description
COMMIT_TX_COUNT	BIGINT	The total number of transactions that were successfully committed using the direct-path option
ABORT_TX_COUNT	BIGINT	The total number of transactions that were rolled back while data were being uploaded using the direct-path option
INSERT_ROW_COUNT	BIGINT	The total number of rows that were inserted by iLoader using the direct-path option
ALLOC_BUFFER_PAGE_ TRY_COUNT	BIGINT	The total number of times that page allocation was requested
ALLOC_BUFFER_PAGE_ FAIL_COUNT	BIGINT	The total number of times that a page allocation request failed

3.2.18.1 Column Information

COMMIT_TX_COUNT

This is the total number of transactions which were committed by iLoader using the direct-path option, accumulated over past executions.

ABORT_TX_COUNT

This is the total number of transactions which were rolled back due to errors while data were being uploaded using the direct-path option, accumulated over past executions.

INSERT_ROW_COUNT

This is the total number of rows which were inserted by iLoader using the direct-path option, accumulated over past executions.

ALLOC_BUFFER_PAGE_TRY_COUNT

This is the total number of times that page allocation was requested for uploading data using the direct-path option, accumulated over past executions.

ALLOC_BUFFER_PAGE_FAIL_COUNT

This is the total number of times that a page allocation request for uploading data using the direct-path option failed due to insufficient memory, accumulated over past executions.

3.2.19 V$DISKTBL_INFO

This view displays information about disk tables.

Column	Data Type	Description
TABLESPACE_ID	SMALLINT	The tablespace identifier
TABLE_OID	BIGINT	The table object identifier
DISK_TOTAL_PAGE_CNT	BIGINT	The total number of pages in a table
DISK_PAGE_CNT	BIGINT	The number of pages containing data in a table
SEG_PID	INTEGER	The page identifier of a segment of a table
META_PAGE	INTEGER	This column has been deprecated
FST_EXTRID	BIGINT	The RID of the first extent in a table
LST_EXTRID	BIGINT	The RID of the last extent in a table
PCTFREE	SMALLINT	See SYS_TABLES_
PCTUSED	SMALLINT	See SYS_TABLES_
INITRANS	SMALLINT	The initial number of transactions that can be simultaneously processed in one page
MAXTRANS	SMALLINT	The maximum number of transactions that can be simultaneously processed in one page
INITEXTENTS	INTEGER	The initial number of extents when a table is created
NEXTEXTENTS	INTEGER	The number of extents that can be allocated when a table is expanded
MINEXTENTS	INTEGER	The minimum number of extents in a table
MAXEXTENTS	INTEGER	The maximum number of extents in a table
COMPRESSED_LOGGING	INTEGER	Whether to compress a log for a table

To display a view together with the name of the table on which it is based, use a query to join the performance view with a meta table as follows:

```
SELECT A.TABLE_NAME,
B.DISK_PAGE_CNT,
B.PCTFREE,
 B.PCTUSED
FROM SYSTEM_.SYS_TABLES_ A, V$DISKTBL_INFO B
WHERE A.TABLE_OID = B.TABLE_OID;
```

3.2.19.1 Column Information

PCTFREE

Please refer to the description of the corresponding column in the SYS_TABLES_ description.

PCTUSED

Please refer to the description of the corresponding column in the SYS_TABLES_ description.

INITRANS

This is the initial number of transactions that can be processed simultaneously in one table page.

MAXTRANS

This is the maximum number of transactions that can be processed simultaneously in one table page.

INITEXTENTS

This is the initial number of extents when a table segment is created.

NEXTEXTENTS

This is the number of additional extents that will be allocated when the size of a table segment is increased.

MINEXTENTS

This is the minimum number of extents in a table segment.

MAXEXTENTS

This is the maximum number of extents in a table segment.

3.2.20 V$DISK_BTREE_HEADER

This view displays information about the header of a disk BTREE index.

Column name	Type	Description
INDEX_NAME	CHAR(40)	The index name
INDEX_ID	INTEGER	The index identifier
INDEX_TBS_ID	INTEGER	The tablespace in which the index is saved
TABLE_TBS_ID	INTEGER	The tablespace in which the table is saved
IS_UNIQUE	CHAR(1)	Whether an index is a unique key index
COLLENINFO_LIST	CHAR(64)	A list of the sizes of the values in the index
IS_CONSISTENT	CHAR(1)	Whether an index is consistent
IS_CREATED_WITH_LOGGING	CHAR(1)	Whether the LOGGING option was specified at the time the index was created

Column name	Type	Description
IS_CREATED_WITH_FORCE	CHAR(1)	Whether the NOLOGGING FORCE or NOLOGGING NOFORCE option was specified at the time the index was created
COMPLETION_LSN_LFG_ID	INTEGER	The log group identifier when the index was created
COMPLETION_LSN_FILE_NO	INTEGER	The log file number when the index was created
COMPLETION_LSN_FILE_OFFSET	INTEGER	The log file offset when the index was created
INIT_TRANS	SMALLINT	The initial number of transactions that can be simultaneously processed in a single index node
MAX_TRANS	SMALLINT	The maximum number of transactions that can be simultaneously processed in a single index node
FREE_NODE_HEAD	INTEGER	The ID of the first page in a free node
FREE_NODE_CNT	BIGINT	The number of pages in a free node list
INITEXTENTS	INTEGER	The initial number of extents when the index was created.
NEXTEXTENTS	INTEGER	The number of extents to be allocated when the index is increased in size
MINEXTENTS	INTEGER	The minimum number of extents in the index segment
MAXEXTENTS	INTEGER	The maximum number of extents in the index segment

3.2.20.1 Column Information

INDEX_NAME

This is the name of the index.

INDEX_ID

This displays the identifier, unique in the system, of the index.

INDEX_TBS_ID

This is the identifier of the tablespace in which the index is saved.

TABLE_TBS_ID

This is the identifier of the tablespace containing the table that is connected to the corresponding index.

IS_UNIQUE

This indicates whether the index is a unique key index. It is set to 'T' for a unique key index, and to 'F' for a duplicate key index.

T: Unique key index

F: Duplicate key index

COLLENINFO_LIST

This is a list of the sizes of the values in the index. The list is expressed as a comma-delimited string. The size of a variable length column is expressed as '?'. The size of a key can be inferred based on this list.

```
Ex)
```

```
iSQL> CREATE TABLE D3(I1 SMALLINT, I2 INTEGER, I3 VARCHAR(10), I4 DATE)
TABLESPACE SYS_TBS_DISK_DATA;
Create success.
iSQL> CREATE INDEX D3X ON D3(I4,I3,I2,I1);
Create success.
iSQL> SELECT COLLENINFO_LIST FROM V$DISK_BTREE_HEADER WHERE INDEX_NAME='D3X';
COLLENINFO_LIST
---------------------------------------------------------------------
8,?,4,2
1 row selected.
```

IS_CONSISTENT

This indicates whether the index is consistent. It is usually set to 'T'. It may be set to 'F' when an index is created with NOLOGGING or NOFORCE.

T: Normal

F: Abnormal

IS_CREATED_WITH_LOGGING

This indicates whether the LOGGING option was specified at the time that the index was created.

IS_CREATED_WITH_FORCE

This value indicates whether the NOLOGGING FORCE or NOLOGGING NOFORCE option was specified at the time that the index was created.

COMPLETION_LSN_LFG_ID

This is the identifier of the log group that was current at the time that the index was created. This column does not have just a single meaning; rather, COMPLETION_LSN_FILE_NO and COMPLETION_LSN_FILE_OFFSET together constitute the LSN. The LSN indicates the time at which index construction was completed.

COMPLETION_LSN_FILE_NO

This is the log file number that was current at the time that the index was created.

COMPLETION_LSN_FILE_OFFSET

This is the log file offset that was current at the time that the index was created.

INIT_TRANS

This is the initial number of transactions that can simultaneously access a single index node (page) for an INSERT, UPDATE or DELETE operation.

MAX_TRANS

This is the maximum number of transactions that can simultaneously access a single index node (page) for an INSERT, UPDATE or DELETE operation.

FREE_NODE_HEAD

A FREE_NODE_HEAD shows the first page of a free node list within an index, a FREE NODE being a node in which a delete mark has been set for all keys therein.

FREE_NODE_CNT

This is the total number of FREE NODEs in an index.

INITEXTENTS

This is the initial number of extents, which is specified at the time that an index segment is created.

NEXTEXTENTS

This is the number of extents to be allocated when the size of an index segment is increased.

MINEXTENTS

This is the minimum number of extents in an index segment.

MAXEXTENTS

This is the maximum number of extents in an index segment.

3.2.21 V$DISK_RTREE_HEADER

This view displays information about the header of a disk RTREE index.

Column name	Type	Description
INDEX_NAME	CHAR(40)	see V$DISK_BTREE_HEADER
INDEX_ID	INTEGER	see V$DISK_BTREE_HEADER
INDEX_TBS_ID	INTEGER	see V$DISK_BTREE_HEADER
TABLE_TBS_ID	INTEGER	see V$DISK_BTREE_HEADER
IS_CONSISTENT	CHAR(1)	see V$DISK_BTREE_HEADER
IS_CREATED_WITH_LOGGING	CHAR(1)	see V$DISK_BTREE_HEADER
IS_CREATED_WITH_FORCE	CHAR(1)	see V$DISK_BTREE_HEADER
COMPLETION_LSN_LFG_ID	INTEGER	see V$DISK_BTREE_HEADER
COMPLETION_LSN_FILE_NO	INTEGER	see V$DISK_BTREE_HEADER
COMPLETION_LSN_FILE_OFFSET	INTEGER	see V$DISK_BTREE_HEADER
INIT_TRANS	SMALLINT	see V$DISK_BTREE_HEADER
MAX_TRANS	SMALLINT	see V$DISK_BTREE_HEADER
FREE_NODE_HEAD	INTEGER	see V$DISK_BTREE_HEADER
FREE_NODE_CNT	BIGINT	see V$DISK_BTREE_HEADER
FREE_NODE_SCN	CHAR(16)	The view SCN that was current when the first page was added to the free node list
INITEXTENTS	INTEGER	see V$DISK_BTREE_HEADER
NEXTEXTENTS	INTEGER	see V$DISK_BTREE_HEADER
MINEXTENTS	INTEGER	see V$DISK_BTREE_HEADER
MAXEXTENTS	INTEGER	see V$DISK_BTREE_HEADER

3.2.21.1 Column Information

For more information about each column, please refer to the V$DISK_BTREE_HEADER performance view.

FREE_NODE_SCN

This is the view SCN that was current when the first page was added to the free node list.

3.2.22 V$DISK_UNDO_USAGE

This view displays the amount of undo tablespace on disk that is currently being used.

Column name	Type	Description
TX_EXT_CNT	BIGINT	The number of extents in all transaction segments
USED_EXT_CNT	BIGINT	The number of extents currently being used in undo segments
UNSTEALABLE_EXT_CNT	BIGINT	The number of extents that cannot be stolen by other undo segments (when a segment does not have enough extents, it can take extents from other undo segments)
REUSABLE_EXT_CNT	BIGINT	The number of extents that can be reused
TOTAL_EXT_CNT	BIGINT	The total number of extents in undo tablespace

3.2.22.1 Column Information

TX_EXT_CNT

This is the number of extents in all transaction segments. These extents cannot be used in undo segments.

USED_EXT_CNT

This is the number of extents currently used in undo segments. Because these extents are currently being used, they cannot be reused by subseqent tasks.

UNSTEALABLE_EXT_CNT

Multiple undo segments exist in the database. Moreover, the number of extents that can be used within each undo segment differs for different undo segments. Therefore, for efficient undo segment management, the "steal" operation is provided so that extents that can be used by other undo segments can be taken by them. However, depending on the circumstances, each undo segment has a certain number of extents that cannot be stolen by other undo segments. These are called "unstealable" extents.

REUSABLE_EXT_CNT

This is the number of extents that can be reused because they contain undo records that are no longer necessary.

TOTAL_EXT_CNT

This is the total number of extents in undo tablespace.

3.2.23 V$EVENT_NAME

This displays information about various wait events for which an ALTIBASE HDB server is waiting.

Column name	Type	Description
EVENT_ID	INTEGER	The identifier of a wait event
NAME	VARCHAR(128)	The name of the wait event
WAIT_CLASS_ID	INTEGER	The identifier of a wait class
WAIT_CLASS	VARCHAR(128)	The name of the wait class

3.2.23.1 Column Information

EVENT_ID

This is the identifier of the wait event.

NAME

This is the name of the wait event. The identifiers, names and corresponding descriptions are given in the following table.

EVENT_ID	NAME	Description
0	latch: buffer busy waits	A wait to access a block being changed by another session
1	latch: drdb B-Tree index SMO	A wait caused by a session that is executing a Structure Modification Operation (SMO) of a B-tree index
2	latch: drdb B-Tree index SMO by other session	A wait until the completion of an SMO of a B-tree index by another session
3	latch: drdb R-Tree index SMO	A wait caused by a session that is executing an SMO of an R-tree index
4	db file multi page read	A wait caused by a session that is waiting for the completion of a request to read multiple pages
5	db file single page read	A wait caused by a session that is waiting for the completion of a request to read a single page
6	db file single page write	A wait until a free BCB is obtained before an LRU flush can be executed
7	enq: TX – row lock contention, data row	A wait to place a lock on a row so that it can be updated
8	enq: TX – allocate TXSEG entry	A wait to assign a transaction segment entry

EVENT_ID	NAME	Description
9	latch free: drdb file i/o	A wait to obtain a file latch in order to perform read/ write I/O on a disk file
10	latch free: drdb tbs list	A wait to obtain a hash latch on a tablespace being used by another thread
11	latch free: drdb tbs cre-ation	A wait caused by a session that is attempting to cre-ate a file when a tablespace is created
12	latch free: drdb page list entry	A wait to obtain a latch on a disk page list being used by another thread
13	latch free: drdb transac-tion segment freelist	A wait for a transaction segment free list
14	latch free: drdb LRU list	A wait for an LRU list in the buffer pool
15	latch free: drdb prepare list	A wait for a prepare list in the buffer pool
16	latch free: drdb prepare list wait	A wait until a BCB has been added to a prepare list in the buffer pool
17	latch free: drdb flush list	A wait for a flush list in the buffer pool
18	latch free: drdb check-point list	A wait for a checkpoint list in the buffer pool
19	latch free: drdb buffer flusher min recovery LSN	A wait for a latch for concurrency control of a Recov-ery LSN of the buffer pool flusher
20	latch free: drdb buffer flush manager req job	A wait for a latch for concurrency control of a flush job of the buffer pool
21	latch free: drdb buffer bcb mutex	A wait for a latch for concurrency control of a BCB of the buffer pool
22	latch free: drdb buffer bcb read io mutex	A wait for a latch on a BCB of the buffer pool for page loading
23	latch free: drdb buffer buffer manager expand mutex	A wait for expansion of the buffer pool
24	latch free: drdb buffer hash mutex	A wait for a buffer pool hash
25	latch free: plan cache LRU List mutex	A wait to obtain a latch on an LRU list in a plan cache when adding, moving, or removing a plan from the list.
26	latch free: statement list mutex	A wait to obtain a latch on a statement list when adding, moving, or removing a statement from the list.

EVENT_ID	NAME	Description
27	latch free: others	A wait to obtain a latch on anything being used by another thread that was not mentioned above
28	no wait event	No wait event exists

WAIT_CLASS_ID

This is the identifier of the class of a wait event. For more detailed information on wait class identifiers, please refer to V$WAIT_CLASS_NAME.

WAIT_CLASS

Wait events are conceptually grouped into broadly defined wait classes. For more detailed information on these wait classes, please refer to V$WAIT_CLASS_NAME.

3.2.24 V$FILESTAT

This view displays cumulative statistical information about I/O on individual disk files since the system was started. These statistics can be used to determine which data files are hot spots.

Column name	Type	Description
SPACEID	INTEGER	The tablespace identifier
FILEID	INTEGER	The data file identifier
PHYRDS	BIGINT	The number of physical read I/O operations that have been conducted
PHYWRTS	BIGINT	The number of physical write I/O operations that have occurred
PHYBLKRD	BIGINT	The number of pages that have been physically opened for reading
PHYBLKWRT	BIGINT	The number of pages that have been physically written to disk
SINGLEBLKRDS	BIGINT	The number of read operations that have taken place on single pages
READTIM	DOUBLE	The total time (in milliseconds) spent on read I/O operations
WRITETIM	DOUBLE	The total time (in milliseconds) spent on write operations
SINGLEBLKRDTIM	DOUBLE	The total time taken to read a single page (in milliseconds)
AVGIOTIM	DOUBLE	The average time (in milliseconds) per I/O operation

Column name	Type	Description
LSTIOTIM	DOUBLE	The time (in milliseconds) spent performing the most recent I/O operation
MINIOTIM	DOUBLE	The shortest time (in milliseconds) spent on a single I/O operation
MAXIORTM	DOUBLE	The longest time (in milliseconds) spent performing a single read operation
MAXIOWTM	DOUBLE	The longest time (in milliseconds) spent performing a single write operation

3.2.24.1 Column Information

SPACEID

This is the identifier of the tablespace.

FILEID

This is the identifier of the data file.

PHYRDS

This is the total number of physical read I/O operations that have been performed.

PHYWRTS

This is the total number of physical write operations that have been performed.

PHYBLKRD

This is the total number of pages that have been opened for physical reading.

PHYBLKWRT

This is the total number of pages that have been physically written to disk.

SINGLEBLKRDS

This is the total number of read I/O operations that have been performed on single pages.

READTIM

This is the total time (in milliseconds) spent performing read I/O operations.

WRITETIM

This is the total time (in milliseconds) spent performing write I/O operations.

SINGLEBLKRDTIM

This is the total amount of time (in milliseconds) spent performing read I/O operations on single pages.

AVGIOTIM

This is the average time (in milliseconds) spent performing a single I/O operation.

LSTIOTIM

This is the time (in milliseconds) spent performing the most recent I/O operation.

MINIOTIM

This is the minimum time (in milliseconds) spent performing a single I/O operation.

MAXIORTM

This is the maximum time (in milliseconds) spent performing a single read I/O operation.

MAXIOWTM

This is the maximum time (in milliseconds) spent performing a single write I/O operation.

3.2.25 V$FLUSHER

This view displays information about flushing tasks.

Column name	Type	Description
ID	INTEGER	This is the identifier of the flusher
ALIVE	INTEGER	This indicates whether the flusher is currently active.
CURRENT_JOB	INTEGER	Current job 1: replacement flushing is underway 2: checkpoint flushing is underway 3: an object is being flushed
DOING_IO	INTEGER	This indicates whether the flusher is performing disk I/O.
INIOB_COUNT	INTEGER	This is the number of times that an internal buffer has been directly accessed in order to save contents to be flushed therein.
REPLACE_FLUSH_JOBS	BIGINT	This is the cumulative number of replacement flushing tasks that have been completed.

Column name	Type	Description
REPLACE_FLUSH_PAGES	BIGINT	This is the cumulative number of pages that have been written to disk by replacement flushing.
REPLACE_SKIP_PAGES	BIGINT	This is the cumulative number of pages for which flushing was canceled during replacement flushing.
CHECKPOINT_FLUSH_JOBS	BIGINT	This is the cumulative number of checkpoint flushing tasks that have been completed.
CHECKPOINT_FLUSH_PAGES	BIGINT	This is the cumulative number of pages that have been written to disk by checkpoint flushing.
CHECKPOINT_SKIP_PAGES	BIGINT	This is the cumulative number of pages for which flushing was canceled during checkpoint flushing.
OBJECT_FLUSH_JOBS	BIGINT	This is the cumulative number of times that object flushing has been performed.
OBJECT_FLUSH_PAGES	BIGINT	This is the cumulative number of pages that have been written to disk by object flushing.
OBJECT_SKIP_PAGES	BIGINT	This is the cumulative number of pages for which flushing was canceled during object flushing.
LAST_SLEEP_SEC	INTEGER	This is the length of time that the flusher has slept after having completed all of its tasks.
TIMEOUT	BIGINT	This is the number of times that a sleeping flusher has woken up in order to check whether it has any tasks.
SIGNALED	BIGINT	This is the number of times that the flusher has been woken up by a signal from ALTIBASE HDB.
TOTAL_SLEEP_SEC	BIGINT	This is the total length of time that the flusher has slept.

3.2.25.1 Column Information

ID

This is the identifier of the flusher. A newly created identifier cannot be a duplicate of an existing identifier.

ALIVE

This indicates whether the flusher is currently active. Individual flushers can be started or stopped using DCL statements.

CURRENT_JOB

This indicates the type of job that the flusher is currently performing. A value of 1 indicates that the flusher is performing replacement flushing. The purpose of replacement flushing is to flush buffers that have not been accessed for a long time so that they can be replaced.

A value of 2 indicates that the flusher is performing checkpoint flushing. The purpose of checkpoint flushing is to flush the buffer that has not been flushed for the longest time in order to reduce the amount of time required to perform checkpointing.

A value of 3 indicates that the flusher is performing object flushing on a particular object, such as an index, table, segment, etc.

DOING_IO

This indicates whether the flusher is currently performing disk I/O in order to fulfill its current task.

INIOB_COUNT

In order to save pages to disk, their contents are saved in an internal buffer (IOB). This value indicates the number of times that this internal buffer has been directly accessed in order to save contents to be flushed therein.

REPLACE_FLUSH_PAGES

This is the cumulative number of pages that have been written to disk in the course of performing replacement flushing tasks.

REPLACE_SKIP_PAGES

This is the cumulative number of pages for which a flushing task was canceled during replacement flushing. Such cancellation can occur either according to some policy or in the interests of efficiency.

CHECKPOINT_FLUSH_PAGES

This is the cumulative number of pages that have been written to disk in the course of performing checkpoint flushing tasks.

CHECKPOINT_SKIP_PAGES

This is the cumulative number of pages for which a flushing task was canceled during checkpoint flushing. Such cancellation can occur either according to some policy or in the interests of efficiency.

OBJECT_FLUSH_PAGES

This is the cumulative number of pages that have been written to disk in the course of performing object flushing tasks.

OBJECT_SKIP_PAGES

This is the cumulative number of pages for which a flushing task was canceled during object flushing. Such cancellation can occur either according to some policy or in the interests of efficiency.

TIMEOUT

Flushers that have no tasks and thus go to sleep are required to wake up at regular intervals to check whether they have work to do. This is the number of times that this has occurred.

SIGNALED

In order to improve the performance with which some task is performed, ALTIBASE HDB can signal a sleeping flusher and wake it up. This value is the number of times that the flusher has been woken up by such a signal.

TOTAL_SLEEP_SEC

This is the total length of time that the flusher has slept because the flusher did not have any work to do.

3.2.26 V$FLUSHINFO

This view displays buffer flush information.

Column	Data Type	Description
LOW_FLUSH_LENGTH	INTEGER	The minimum length of the flush list above which replacement flushing can occur
HIGH_FLUSH_LENGTH	INTEGER	The flush list length at which the flusher ignores REPLACE_FLUSH_COUNT and flushes all the buffers in the flush list.
LOW_PREPARE_LENGTH	INTEGER	The threshold length of the prepare list that can cause replacement flushing. Replacement flushing occurs when the prepare list is shorter than this length.
CHECKPOINT_FLUSH_COUNT	INTEGER	The number of buffers to be flushed when checkpoint flushing occurs.
FAST_START_IO_TARGET	BIGINT	The number of dirty pages that will not be flushed when checkpoint flushing occurs
FAST_START_LOGFILE_TARGET	INTEGER	The number of log files that will not be flushed when checkpoint flushing occurs
REQ_JOB_COUNT	INTEGER	The number of tasks currently registered for the flush manager

3.2.26.1 Column Information

LOW_FLUSH_LENGTH

This is the minimum length of the flush list above which replacement flushing can occur.

HIGH_FLUSH_LENGTH

This is the flush list length at which the flusher ignores REPLACE_FLUSH_COUNT and flushes all the buffers in the flush list.

LOW_PREPARE_LENGTH

This is the threshold length of the prepare list. Replacement flushing occurs if the length of a prepare list drops below this length.

CHECKPOINT_FLUSH_COUNT

This is the number of buffers that will be flushed when checkpoint flushing is performed.

FAST_START_IO_TARGET

This is the number of dirty pages that are not flushed when checkpoint flushing occurs.

FAST_START_LOGFILE_TARGET

This is the number of log files that are not flushed when checkpoint flushing occurs. These are the most recently created log files.

REQ_JOB_COUNT

This is the number of jobs registered in the flush manager.

3.2.27 V$INDEX

This view shows information about the indexes that currently exist in the database:

Column Name	Type	Description
TABLE_OID	BIGINT	The object identifier of the table header
INDEX_SEG_PID	INTEGER	The page identifier of a segment header in the case of a disk index
INDEX_ID	INTEGER	The identifier of the index
INDEX_TYPE	VARCHAR(7)	An indicator that identifies whether the index is a primary key or a standard index

3.2.27.1 Column Information

TABLE_OID

This is the object identifier of the table for which the index was created, and stores the physical location of the header, which contains the table information.

INDEX_SEG_PID

For a disk index, this is the page identifier of a segment header.

INDEX_ID

This is the identifier of the index in the system.

INDEXTYPE

This indicates whether the index is used as a primary key or as a normal index.

PRIMARY: The index is used as primary key.

NORMAL: The index is used as normal one.

3.2.28 V$INSTANCE

This view displays information about an Altibase database, the amount of time it took to start up, and the amount of time that has elapsed since startup.

Column	Data Type	Description
STARTUP_PHASE	VARCHAR(13)	The current startup phase
STARTUP_TIME_SEC	BIGINT	The system time at which the system was started (in seconds).
WORKING_TIME_SEC	BIGINT	The amount of time that has elapsed from startup to the present

3.2.29 V$LATCH

This view displays statistical information about the BCB latch of the buffer pool, including the number of attempts to obtain a latch on pages on which it is desired to perform read or write I/O, the number of latches that were successfully obtained immediately, and the number of failures to obtain a latch. These statistics are calculated separately for read and write latches.

Column	Data Type	Description
SPACE_ID	INTEGER	The tablespace identifier
PAGE_ID	INTEGER	The page identifier
TRY_READ_LATCH	BIGINT	The number of attempts to obtain read latches
READ_SUCCESS_IMME	BIGINT	The number of immediate successes to obtain read latches
READ_MISS	BIGINT	The number of failures to obtain read latches

Column	Data Type	Description
TRY_WRITE_LATCH	BIGINT	The number of attempts to obtain write latches
WRITE_SUCCESS_IMME	BIGINT	The number of immediate successes to obtain write latches
WRITE_MISS	BIGINT	The number of failures to obtain write latches
SLEEPS_CNT	BIGINT	The number of sleeps related to latch attempts

3.2.30 V$LFG

This view provides statistical information to help database administrators monitor group commit activity. For more detailed information on each column, please refer to the section in *Administrator's Manual* pertaining to Group Commit.

Column	Data Type	Description
LFG_ID	INTEGER	The log file group identifier
CUR_WRITE_LF_NO	INTEGER	The log file number of the log file currently being written to
CUR_WRITE_LF_OFFSET	INTEGER	The offset of the log file currently being written to
LF_OPEN_COUNT	INTEGER	The number of open log files
LF_PREPARE_COUNT	INTEGER	The number of log files that have been created in advance
LF_PREPARE_WAIT_COUNT	INTEGER	The number of waits to switch to new log files
LST_PREPARE_LF_NO	INTEGER	The identifier of the most recently prepared log file
END_LSN_LFGID	INTEGER	The Log File Group portion of the LSN (Log Sequence Number) at which a REDO operation will start when ALTIBASE HDB is restarted
END_LSN_FILE_NO	INTEGER	The file number portion of the LSN (Log Sequence Number) at which a REDO operation will start when ALTIBASE HDB is restarted
END_LSN_OFFSET	INTEGER	The offset within a LSN (Log Sequence Number) at which a REDO operation will start when ALTIBASE HDB is restarted

Column	Data Type	Description
FIRST_DELETED_LOGFILE	INTEGER	The first log file that was deleted (inclusive)
LAST_DELETED_LOGFILE	INTEGER	The log file immediately preceding this log file is the last log file that was deleted
RESET_LSN_LFGID	INTEGER	The Log File Group identifier portion of the LSN (Log Sequence Number) used after database recovery
RESET_LSN_FILE_NO	INTEGER	The file number portion of the LSN (Log Sequence Number) used after database recovery
RESET_LSN_OFFSET	INTEGER	The offset of the LSN (Log Sequence Number) used after database recovery
UPDATE_TX_COUNT	INTEGER	The number of transactions in the LFG that are currently making changes to the database (only available for group commit)
GC_WAIT_COUNT	INTEGER	The number of waits for disk I/O (only available for group commit)
GC_ALREADY_SYNC_COUNT	INTEGER	The number of completed disk I/O operations (only available for group commit)
GC_REAL_SYNC_COUNT	INTEGER	The number of actual disk I/O operations that occurred during group commit

3.2.30.1 Column Information

LFG_ID

This is a unique log file group number, starting from 0 and incremented by 1.

For example, if there are four log file groups in a system, querying LFG_ID will result in four rows with the values 0, 1, 2, and 3.

CUR_WRITE_LF_NO

This is the number of the log file currently being used to store logs.

CUR_WRITE_LF_OFFSET

This is the log file offset currently being used to store logs.

LF_OPEN_COUNT

This is the number of log files on disk that are open for use by ALTIBASE HDB.

LF_PREPARE_COUNT

This is the number of log files that have been created in advance (prepared) by the log file creation thread up to the present moment.

LF_PREPARE_WAIT_COUNT

When all of the prepared log files have been used, it is necessary to create new log files. This is the total number of waits for log files to be created in order to switch to a new log file.

If this value is large, setting the `PREPARE_LOG_FILE_COUNT` property to a higher value will help ensure that a sufficient number of log files is prepared in advance. For more information about `PREPARE_LOG_FILE_COUNT`, please refer to the *ALTIBASE HDB General Reference*.

LST_PREPARE_LF_NO

This is the number of the log file that was most recently prepared (created in advance) by the log file creation thread.

END_LSN_LFGID

When the system is restarted, this is a unique LFG number, which is part of the LSN (Log Sequence Number) at which REDO restarts. It is the same as the value in the LFG_ID column.

When the system is restarted, REDO may not start at precisely this position within the LFG. However, it can be guaranteed that REDO will definitely begin with a log having a greater LSN value than the one shown here.

END_LSN_FILE_NO

This shows the number of the log file, which is part of the LSN (Log Sequence Number), at which REDO commences when the system is restarted.

END_LSN_OFFSET

This shows the offset within the log file, which is part of the LSN (Log Sequence Number), at which REDO commences when the system is restarted.

FIRST_DELETED_LOGFILE

This shows the number of the first of the log files that were classified as unnecessary and deleted during checkpointing. This means that the log file having this number was deleted during check-pointing.

LAST_DELETED_LOGFILE

This shows a number which is 1 greater than the number of the last of the log files that were classified as unnecessary and deleted during checkpointing. This means that the log file having this number was not deleted during checkpointing.

RESET_LSN_LFGID

RESET_LSN is the LSN for recording logs pertaining to new tasks that arise after the time point at which database recovery occurs due to the system suffering from a fault or for some other reason. This column contains the unique LFG number, which is part of the RESET_LSN. It has the same value as that in the LFG_ID column.

RESET_LSN_FILE_NO

RESET_LSN is the first LSN after the time point at which recovery was performed.
RESET_LSN_FILE_NO is the log file number portion of RESET_LSN.

RESET_LSN_OFFSET

This shows the offset within the log file, and is a portion of RESET_LSN.

UPDATE_TX_COUNT

This returns, in real time, the number of transactions in the LFG that are currently making changes to the database.

GC_WAIT_COUNT

This shows the total number of times transactions in this LFG had to wait for disk I/O for group commit.

GC_ALREADY_SYNC_COUNT

During group commit, it is sometimes not necessary to perform disk I/O for some transactions, because the logs containing them have already been written to disk. This is the cumulative number of times this has occurred.

GC_REAL_SYNC_COUNT

This shows the number of actual disk I/O operations related to transactions in this LFG during group commit.

3.2.31 V$LINKER_STATUS

This view shows the status of AltiLinker for Database Link.

Column name	Type	Description
LINKER_STATUS	INTEGER	Indicates the linker status. If it is 1, the linker is in a normal state. If it is 0, the linker is in an abnormal state, or is not available.
SESSION_COUNT	INTEGER	Indicates the number of Database Link sessions between ALTIBASE HDB and the linker.

3.2.31.1 Column Information

LINKER_STATUS

This is the status of the linker. A value of 1 indicates that the linker is in a normal state, while a value of 0 indicates that the linker is in an abnormal state, or is not available.

3.2.32 V$LOCK

This view displays information about lock nodes for all tables in the database at the current point in time.

Column	Data Type	Description
LOCK_ITEM_TYPE	VARCHAR(7)	The type of object that is locked
TBS_ID	INTEGER	The tablespace identifier
TABLE_OID	BIGINT	The table object identifier
DBF_ID	BIGINT	The database file identifier
TRANS_ID	BIGINT	The transaction identifier
LOCK_DESC	VARCHAR(32)	A character string indicating the lock mode e.g.) IX, IS, X
LOCK_CNT	INTEGER	The number of locks for this lock node
IS_GRANT	BIGINT	Indicates whether the table is locked or is waiting to be locked

3.2.32.1 Column Information

LOCK_ITEM_TYPE

This indicates the type of object that is locked, and can have the following values:

Value	Description
NONE	Cannot have this value
TBS	Tablespace
TBL	Table
DBF	Database file
UNKNOWN	Unknown object type

3.2.33 V$LOCK_STATEMENT

This view displays information about statements that are holding or waiting to acquire locks.

Column	Data Type	Description
SESSION_ID	INTEGER	The session identifier
ID	INTEGER	The statement identifier
TX_ID	BIGINT	The transaction identifier
QUERY	VARCHAR(16384)	The query statement
STATE	INTEGER	The state of the statement
BEGIN_FLAG	INTEGER	A flag indicating the beginning of the statement
LOCK_ITEM_TYPE	VARCHAR(7)	The type of object that is locked
TBS_ID	INTEGER	The tablespace identifier
TABLE_OID	BIGINT	The table object identifier
DBF_ID	BIGINT	The database file identifier
LOCK_DESC	VARCHAR(32)	A character string indicating the lock mode e.g.) IX, IS, X
LOCK_CNT	INTEGER	The number of locks for the lock node
IS_GRANT	BIGINT	Indicates whether the table is locked or is waiting to be locked

3.2.34 V$LOG

This view displays information about log anchors.

Column	Data Type	Description
BEGIN_CHKPT_LFGID	INTEGER	The LFGID of the checkpoint start log of the most recently executed checkpoint
BEGIN_CHKPT_FILE_NO	INTEGER	The log file number of the checkpoint start log of the most recently executed checkpoint
BEGIN_CHKPT_FILE_OFFSET	INTEGER	The log offset of the checkpoint start log of the most recently executed checkpoint
END_CHKPT_LFGID	INTEGER	The LFGID of the checkpoint end log of the most recently executed checkpoint

Column	Data Type	Description
END_CHKPT_FILE_NO	INTEGER	The log file number of the checkpoint end log of the most recently executed check-point
END_CHKPT_FILE_OFFSET	INTEGER	The log offset of the checkpoint end log of the most recently executed checkpoint
SERVER_STATUS	VARCHAR(15)	A character string indicating the status of the server
ARCHIVELOG_MODE	VARCHAR(12)	A character string indicating the status of database archive mode
TRANSACTION_SEGMENT_COUNT	INTEGER	The number of transaction segments to be created in undo tablespace
OLDEST_LFGID	INTEGER	When restart recovery is performed, the LFGID of the LSN from which disk-related redo will begin
OLDEST_LOGFILE_NO	INTEGER	When restart recovery is performed, the log file number from which disk-related redo will begin
OLDEST_LOGFILE_OFFSET	INTEGER	When restart recovery is performed, the log file offset from which disk-related redo will begin

3.2.34.1 Column Information

BEGIN_CHKPT_LFGID

This is the Log File Group ID of the log file containing the log at which the most recent checkpoint began.

BEGIN_CHKPT_FILE_NO

This is the file number of the log file containing the log at which the most recent checkpoint began.

BEGIN_CHKPT_FILE_OFFSET

This is the log offset of the log file containing the log at which the most recent checkpoint began.

END_CHKPT_LFGID

This is the Log File Group ID of the log file containing the log at which the most recent checkpoint ended.

END_CHKPT_FILE_NO

This is the file number of the log file containing the log at which the most recent checkpoint ended.

END_CHKPT_FILE_OFFSET

This is the log offset of the log file containing the log at which the most recent checkpoint ended.

SERVER_STATUS

This is the status of the server.

SERVER SHUTDOWN: The server has been shut down.

SERVER STARTED: The server is running.

ARCHIVELOG_MODE

This indicates whether Archivelog mode is enabled for the database.

ARCHIVE: In this mode, unnecessary log files are stored in an extra directory for use in performing media recovery.

NOARCHIVE: In this mode, unnecessary log files are deleted.

TRANSACTION_SEGMENT_COUNT

This is the number of transaction segments to be created in undo tablespace.

OLDEST_LFGID

This is the Log File Group ID of the log file containing the LSN from which REDO will start when the database is restarted in recovery mode. Every log is identified by a unique log sequence number (LSN). This ensures that recovery performs REDO on all log records required to bring pages up to date.

OLDEST_LOGFILE_NO

This is the number of the log file containing the LSN from which REDO will start when the database is restarted in recovery mode.

OLDEST_LOGFILE_OFFSET

This is the offset of the log file containing the LSN from which REDO will start when the database is restarted in recovery mode.

3.2.35 V$LOCK_WAIT

This view shows wait information between transactions that are executed on the system.

Column name	Type	Description
TRANS_ID	BIGINT	The identifier of the waiting transaction

Column name	Type	Description
WAIT_FOR_TRANS_ID	BIGINT	The identifier of the transaction being waited for

3.2.35.1 Column Information

TRANS_ID

This is the identifier of the transaction that is currently waiting.

WAIT_FOR_TRANS_ID

This is the identifier of the transaction for which the transaction identified by TRANS_ID is waiting.

```
SQL> select * from v$lock_wait;
V$LOCK_WAIT.TRANS_ID V$LOCK_WAIT.WAIT_FOR_TRANS_ID
---------------------------------------------
1216 2208
5344 2208
2 rows selected.
```

In the above example, transactions 1216 and 5344 are waiting for transaction 2208.

3.2.36 V$MEMGC

This view displays memory space recovery (that is, memory garbage collection) information.

Column	Data Type	Description
GC_NAME	VARCHAR(128)	MEM_LOGICAL_AGER: Previous version index key slot release thread MEM_DELTHR: A thread that releases deleted records and supports pending operations such as DROP TABLE etc.
CURRSYSTEMVIEWSCN	VARCHAR(29)	The current system view SCN
MINMEMSCNINTXS	VARCHAR(29)	The lowest of the view SCNs for memory-related transactions
OLDESTTX	INTEGER	The identifier of the oldest transaction (the identifier of the transaction to which MIN-MEMSCNINTXS belongs)
SCNOFTAIL	VARCHAR(29)	The commit SCN of the tail in garbage collection OID list

Column	Data Type	Description
IS_EMPTY_OIDLIST	BIGINT	Whether the garbage collection OID list is empty 0: empty 1: not empty
ADD_OID_CNT	BIGINT	The number of transactions that caused OIDs to be added for garbage collection management
GC_OID_CNT	BIGINT	The number of times OIDs are deleted for garbage collection
AGING_REQUEST_OID_CNT	BIGINT	The number of outdated versions of records for which deletion has been requested
AGING_PROCESSED_OID_CNT	BIGINT	The number of outdated versions of records that have been deleted
THREAD_COUNT	INTEGER	The number of garbage collection threads

3.2.36.1 Column Information

Because ALTIBASE HDB supports MVCC, multiple versions of a single record can exist. In other words, one record consists of a most recent version and a number of previous versions. For more details on MVCC, please refer to the sections pertaining to Multi-Version Concurrency Control (MVCC) in both the *Administrator's Manual* and the *Getting Started Guide*.

AGING_REQUEST_OID_CNT

If 10 records are deleted in one transaction, which is then committed, there are now 10 outdated records that can be cleared to recover space. However, because ADD_OID_CNT is determined on the basis of transactions, it is incremented by 1. To remedy this, AGING_REQUEST_OID_CNT, which is determined on the basis of OIDs, is incremented by 10.

AGING_PROCESSED_OID_CNT

If the garbage collector (or ager) deletes 10 outdated versions of records from the same OID list, GC_OID_CNT is only incremented by 1 because it determined on the basis of lists. To remedy this, AGING_PROCESSED_OID_CNT, which is determined on the basis of OIDs, is incremented by 10.

THREAD_COUNT

This shows the number of garbage collection threads.

3.2.37 V$MEMSTAT

This view displays statistics about the memory being used by ALTIBASE HDB processes.

Column	Data Type	Description
NAME	VARCHAR(40)	The name of the memory module
ALLOC_SIZE	BIGINT	The amount of memory being used by the module (in bytes)
ALLOC_COUNT	BIGINT	The number of units of memory that make up ALLOC_SIZE
MAX_TOTAL_SIZE	BIGINT	The maximum memory size of the module (in bytes)

3.2.37.1 Column Information

NAME

This is the name of the module being used by ALTIBASE HDB. This column contains the following memory modules.

Name	Description
Async_IO_Manager	The memory that is used when asynchronous I/O occurs
CM_Buffer	The buffer memory used for communcation (TCP, Unix domain Socket, IPC)
CM_DataType	The memory that is used for sending and receiving large pack-ets
CM_Multiplexing	The memory that is used for saving session information for communication
CM_NetworkInterface	The memory that is used for saving information about individ-ual communication nodes
Clock_Manager	The memory for the clock manager. The clock manager uses the CPU clock when it checks the system time.
Cond_Manager	The memory that is used for managing condition variables used for multiple thread control
DatabaseLink	The memory that is used by Database Link
Dynamic Module Loader	The memory that is used when the shared library is loaded
GIS_DataType	The memory that is used for handling GIS data
GIS_Disk_Index	The memory that is used for managing the Disk Spatial Index for GIS data
GIS_Function	The memory that is used for space-related calculations
GIS_TEMP_MEMORY	The memory that is used for creating R-tree indexes

Name	Description
Index_Memory	The memory that is used for managing index information
Linker	The memory that is used by the Linker process of the Database Link module
Main_Module_Channel	ALTIBASE HDB Main Module Process
Main_Module_Distributed	The memory that is used for XA management
Main_Module_Queue	The memory that is used for queues
Main_Module_Thread	The memory that is used for managing threads
Main_Module_Utility	Not used at present
Mathematics	The memory that is used for various kinds of mathematical operations
Mutex_Manager	The memory that is used for managing mutexes
OS_Independent	Not used at present
Profile_Manager	The memory that is used by the Profile Manager
Query_Binding	The memory that is used for binding host variables
Query_Common	Memory that is used for other purposes
Query_Conversion	The memory that is used to perform conversion when binding host variables
Query_DML	The memory that is used for executing DML statements
Query_Execute	The memory that is used when queries are executed
Query_Meta	The memory that is used to manage cached meta information, which is checked while the server is active
Query_PSM_Execute	The memory that is used for executing PSM (Persistent Stored Module)
Query_PSM_Node	The memory that is used for managing PSM array variables
Query_Prepare	The memory that is used for preparing queries for execution
Query_Sequence	The memory that is used for managing sequences
Query_Transaction	The memory that is used for executing triggers
Replication_Common	Not used at present
Replication_Control	The memory that is used by the Replication Manager
Replication_Data	The memory that is used for processing XLOGs
Replication_Executor	Not used at present
Replication_Met	The memory that is used by the meta cache
Replication_Network	The memory that is used for communication for replication

Name	Description
Replication_Receiver	The memory that is used by the replication Receiver
Replication_Recovery	The memory that is used to perform recovery using replication
Replication_Sender	The memory that is used by the replication Sender
Replication_Storage	The memory that is used to apply XLOGs
Replication_Sync	The memory that is used for synchronization in replication
SQL Plan Cache Control	The memory that is used for the SQL Plan Cache
Socket_Manager	Not used at present
Storage_DataPort	Memory that is used for executing DataPort
Storage_Disk_Buffer	The memory that is used by the Disk Buffer Manager
Storage_Disk_Collection	The memory that is used for performing Direct-Path Insert and LOB calculations for disk tables
Storage_Disk_Datafile	The memory that is used for data file management tasks, such as creating I/O buffers and data file nodes
Storage_Disk_Index	The memory that is used for managing disk indexes
Storage_Disk_Page	The memory that is used for assigning disk LOB segment descriptors and disk table page list mutexes
Storage_Disk_Recovery	Memory that is used to ensure the consistency of a disk data-base
Storage_Memory_Ager	Memory that is used for the garbage collector and the data-base recovery ("refining") thread
Storage_Memory_Collection	Memory that is used for managing records in memory tables
Storage_Memory_Index	Memory that is used for managing memory indexes
Storage_Memory_Interface	Memory that is used at the storage module interface level
Storage_Memory_Locking	Memory that is used for locking tables and tablespaces
Storage_Memory_Manager	The memory in which memory data are actually stored
Storage_Memory_Page	The memory that is used for managing memory pages
Storage_Memory_Recovery	The memory that is used to perform recovery
Storage_Memory_Transaction	Memory that is used for managing transaction information
Storage_Memory_Utility	Memory that is used when the Storage Manager Tool is used
Storage_Tablespace	Memory that is used for managing and allocating tablespace nodes
Tablespace Free Extent Pool	The memory that is used for managing free extent pools of tablespaces
Temp_Memory	The memory that is used when allocating temporary space

Name	Description
Timer_Manager	Memory for the timer manager, which uses the timer thread when checking the system time
Transaction_DiskPage_Touched_List	The memory that is used for managing disk data pages that are affected by a transaction
Transaction_OID_List	The memory that is used for making the OID (object identifier) list of a memory database
Transaction_Segment_Table	Memory that is used for managing Undo segments and Transaction Status segments
Transaction_Table	Memory that is used for assigning transaction objects
Transaction_Table_Info	Memory that is used for managing information about the tables changed by a transaction
Volatile_Log_Buffer	Volatile Log Buffer memory
Volatile_Memory_Manager	The memory in which volatile memory data are stored
Volatile_Memory_Page	The memory that is used for managing volatile memory pages

ALLOC_SIZE

This is the amount of memory being used by the module.

ALLOC_COUNT

This is the number of units of memory that make up ALLOC_SIZE in the module.

MAX_TOTAL_SIZE

This is the maximum memory size that the module has occupied.

3.2.38 V$MEMTBL_INFO

This view displays information about the status of memory tables.

Column	Data Type	Description
TABLESPACE_ID	SMALLINT	The tablespace identifier
TABLE_OID	BIGINT	The table object identifier
MEM_PAGE_CNT	BIGINT	The number of pages containing fixed-length columns in the table
MEM_VAR_PAGE_CNT	BIGINT	The number of pages containing variable-length columns in the table

Column	Data Type	Description
MEM_SLOT_PERPAGE	INTEGER	The number of slots that can be stored in a page containing fixed-length columns
MEM_SLOT_SIZE	BIGINT	The size of the fixed area in the table record
FIXED_ALLOC_MEM	DOUBLE	The amount of fixed memory area (in bytes) allocated to a table
FIXED_USED_MEM	BIGINT	The amount of fixed memory area (in bytes) actually being used by a table
VAR_ALLOC_MEM	DOUBLE	The amount of variable memory area (in bytes) allocated to a table
VAR_USED_MEM	BIGINT	The amount of variable memory area (in bytes) actually being used by a table
MEM_FIRST_PAGEID	BIGINT	The number of the first of the fixed-length pages in the table
STATEMENT_REBUILD_COUNT	BIGINT	The number of times a statement has been rebuilt
UNIQUE_VIOLATION_COUNT	BIGINT	The number of times a unique key violation has occurred
UPDATE_RETRY_COUNT	BIGINT	The number of times an update operation has been retried
DELETE_RETRY_COUNT	BIGINT	The number of times a delete operation has been retried
COMPRESSED_LOGGING	INTEGER	Indicates whether log compression is enabled or not

To view this information together with the table name, join this view with the SYS_TABLES_ meta table and execute a query as follows:

```
SELECT A.TABLE_NAME,
B.MEM_PAGE_CNT,
B.MEM_SLOT_SIZE
B.MEM_FIRST_PAGEID
FROM SYSTEM_.SYS_TABLES_ A, V$MEMTBL_INFO B
WHERE A.TABLE_OID = B.TABLE_OID;
```

3.2.38.1 Column Information

TABLESPACE_ID

This is the identifier of the tablespace in which the current table is stored. The following tablespaces are created by default. Identifiers of new user-created tablespaces will have values greater than 4.

0: SYS_TBS_MEM_DIC

1: SYS_TBS_MEM_DATA

2: SYS_TBS_DISK_DATA

3: SYS_TBS_DISK_UNDO

4: SYS_TBS_DISK_TEMP

TABLE_OID

This is the default table object identifier, and indicates the physical location of the header that contains information about the table. This is only used internally by the system.

MEM_PAGE_CNT

This is the number of fixed-length pages allocated to the table.

MEM_VAR_PAGE_CNT

This is the number of variable-length pages allocated to the table.

MEM_SLOT_PERPAGE

This is the number of slots that can be stored in a single fixed-length page.

MEM_SLOT_SIZE

This is the fixed area that is occupied by one record in a memory table.

FIXED_ALLOC_MEM

This is the amount of fixed area (in bytes) allocated to a table.

FIXED_USED_MEM

This is the amount of fixed area (in bytes) that is actually used by a table.

VAR_ALLOC_MEM

This is the amount of variable area (in bytes) allocated to a table.

VAR_USED_MEM

This is the amount of variable area (in bytes) actually used by a table.

MEM_FIRST_PAGEID

This is the identifier of the first of the fixed-length pages allocated to a table.

STATEMENT_REBUILD_COUNT

When the Prepare-Execute process is performed, a prepared statement is executed without being parsed, validated, or optimized. However, after the statement is prepared, if a DDL statement is executed on a query target object (a tablespace, table or index), the corresponding statement is automatically rebuilt when the statement is executed, and this value is incremented.

UNIQUE_VIOLATION_COUNT

This value is incremented when a unique key restriction is violated.

UPDATE_RETRY_COUNT

This value is incremented when an attempt to perform an update operation is repeated.

DELETE_RETRY_COUNT

This value is incremented when an attempt to perform a delete operation is repeated.

3.2.39 V$MEM_BTREE_HEADER

This view shows information about a memory BTREE header.

Column name	Type	Description
INDEX_NAME	CHAR(40)	The name of the index
INDEX_ID	INTEGER	The index identifier
INDEX_TBS_ID	INTEGER	The tablespace in which the index is stored
TABLE_TBS_ID	INTEGER	The tablespace in which the associated table is stored
IS_UNIQUE	CHAR(1)	Whether an index is a unique key index
IS_NOT_NULL	CHAR(1)	Whether NULL values are allowed
USED_NODE_COUNT	INTEGER	The number of nodes that are being used by an index
PREPARE_NODE_COUNT	INTEGER	The number of nodes that are prepared in advance to meet the demand for nodes
BUILT_TYPE	CHAR(1)	The key type that was used when the index was created

3.2.39.1 Column Information

INDEX_NAME

This is the name of the index.

INDEX_ID

This is a unique identifier for the index in the system.

INDEX_TBS_ID

This is the identifier of the tablespace in which the index is stored.

TABLE_TBS_ID

This is the identifier of the tablespace containing the table that is related to the index.

IS_UNIQUE

This indicates whether the index is a unique key index. It is set to 'T' to indicate a unique key index, and to 'F' to indicate a duplicate key index.

IS_NOT_NULL

This indicates whether NULL values are allowed. It is set to 'T' for a primary key index, and to 'F' for other kinds of indexes.

USED_NODE_COUNT

This indicates the total number of nodes for the current index. This number increases when a node is split, and decreases when a node is deleted.

PREPARE_NODE_COUNT

This is the number of nodes that are allocated in advance in consideration of system load, based on the number of nodes that have been assigned.

BUILT_TYPE

This indicates whether a key value or a record pointer was used when the index was built. It is set to 'V' to indicate that a key value was used, and to 'P' to indicate that a record pointer was used.

3.2.40 V$MEM_BTREE_NODEPOOL

This view shows information about the node pool for memory BTREE indexes. The node pool manages node allocation and return for all memory BTREE indexes.

Column name	Type	Description
TOTAL_PAGE_COUNT	INTEGER	The total number of pages in the node pool
TOTAL_NODE_COUNT	INTEGER	The total number of nodes in the node pool
FREE_NODE_COUNT	INTEGER	The number of unallocated nodes in the node pool
USED_NODE_COUNT	INTEGER	The number of nodes allocated to indexes
NODE_SIZE	INTEGER	The size of a node (in bytes)

Column name	Type	Description
TOTAL_ALLOC_REQ	BIGINT	The cumulative number of node allocation requests made to the node pool
TOTAL_FREE_REQ	BIGINT	The cumulative number of node deletion requests made to the node pool
FREE_REQ_COUNT	INTEGER	The number of nodes in the node pool waiting to be deleted

3.2.40.1 Column Information

TOTAL_PAGE_COUNT

This shows the number of pages allocated to the node pool for BTREE indexes.

TOTAL_NODE_COUNT

This indicates the number of nodes allocated to the node pool for BTREE indexes. It is determined by TOTAL_PAGE_COUNT and NODE_SIZE.

FREE_NODE_COUNT

This is the number of nodes that have not been allocated to BTREE indexes, and thus remain in the node pool.

USED_NODE_COUNT

This shows the total number of nodes that are currently allocated to BTREE indexes.

NODE_SIZE

This is the size of a BTREE index node.

TOTAL_ALLOC_REQ

This is the number of node allocation requests that have been made to the node pool. This is the cumulative number since the system was started.

TOTAL_FREE_REQ

This is the number of return requests that have been made to the node pool for nodes that were used for BTREE indexes and then deleted. This is the cumulative number since the system was started.

FREE_REQ_COUNT

This is the number of nodes that were being used by BTREE indexes and are waiting to be deleted.

3.2.41 V$MEM_RTREE_HEADER

This view shows information about the header of a memory RTREE index.

Column name	Type	Description
INDEX_NAME	CHAR(40)	The name of the index
INDEX_ID	INTEGER	The index identifier
TABLE_TBS_ID	INTEGER	The identifier of the tablespace in which the table is stored
TREE_MBR_MIN_X	DOUBLE	The minimum X value of the RTREE index
TREE_MBR_MIN_Y	DOUBLE	The minimum Y value of the RTREE index
TREE_MBR_MAX_X	DOUBLE	The maximum X value of the RTREE index
TREE_MBR_MAX_Y	DOUBLE	The maximum Y value of the RTREE index
USED_NODE_COUNT	INTEGER	The number of nodes that are being used by the index
PREPARE_NODE_COUNT	INTEGER	The number of nodes that have been pre-allocated to meet node demand

3.2.41.1 Column Information

INDEX_NAME

This is the name of the index.

INDEX_ID

This is the identifier of the index. This identifier is unique within the system.

TABLE_TBS_ID

This is the identifier of the tablespace containing the table that is related to the index.

TREE_MBR_MIN_X

This is the minimum X value of the minimum bounding box of the RTREE index.

TREE_MBR_MIN_Y

This is the minimum Y value of the minimum bounding box of the RTREE index.

TREE_MBR_MAX_X

This is the maximum X value of the minimum bounding box of the RTREE index.

TREE_MBR_MAX_Y

This is the maximum Y value of the minimum bounding box of the RTREE index.

USED_NODE_COUNT

This is the total number of nodes being used by the current index. This number increases when a node is split and decreases when a node is deleted.

PREPARE_NODE_COUNT

This is the number of nodes that are allocated in advance in consideration of system load, based on the number of nodes that have been assigned.

3.2.42 V$MEM_RTREE_NODEPOOL

This view shows information about the node pool for memory RTREE indexes. This node pool manages node allocation and return for all memory RTREE indexes.

Column name	Type	Description
TOTAL_PAGE_COUNT	INTEGER	The total number of pages in the node pool
TOTAL_NODE_COUNT	INTEGER	The total number of nodes in the node pool
FREE_NODE_COUNT	INTEGER	The number of unallocated nodes in the node pool
USED_NODE_COUNT	INTEGER	The number of nodes allocated to indexes
NODE_SIZE	INTEGER	The size of a node (in bytes)
TOTAL_ALLOC_REQ	BIGINT	The cumulative number of node allocation requests made to the node pool
TOTAL_FREE_REQ	BIGINT	The cumulative number of node deletion requests made to the node pool
FREE_REQ_COUNT	INTEGER	The number of nodes in the node pool that are waiting to be deleted

3.2.42.1 Column Information

TOTAL_PAGE_COUNT

This is the number of pages allocated to the node pool for RTREE indexes.

TOTAL_NODE_COUNT

This is the total number of nodes allocated to the node pool for RTREE indexes. It is determined by TOTAL_PAGE_COUNT and NODE_SIZE.

FREE_NODE_COUNT

This is the number of nodes that have not been allocated to RTREE indexes and thus remain in the node pool.

USED_NODE_COUNT

This is the total number of nodes that are currently allocated to RTREE indexes.

NODE_SIZE

This is the size of an RTREE index node.

TOTAL_ALLOC_REQ

This is the number of node allocation requests that have been made to the node pool. This is the cumulative number since the system was started.

TOTAL_FREE_REQ

This is the number of return requests that have been made to the node pool for nodes that were being used by RTREE indexes and were then deleted. This is the cumulative number since the system was started.

FREE_REQ_COUNT

This is the number of nodes that were being used by RTREE indexes and are waiting to be deleted.

3.2.43 V$MEM_TABLESPACES

This view shows information about tablespaces that exist in memory.

Column name	Type	Description
SPACE_ID	INTEGER	The tablespace identifier
SPACE_NAME	VARCHAR(512)	The name of the tablespace
SPACE_STATUS	INTEGER	The tablespace status
SPACE_SHM_KEY	INTEGER	The shared memory key of the tablespace
AUTOEXTEND_MODE	INTEGER	The auto extension mode of the tablespace
AUTOEXTEND_NEXT_SIZE	BIGINT	The size (in bytes) by which the tablespace is automatically extended
MAXSIZE	BIGINT	The maximum size of the tablespace (in bytes)
CURRENT_SIZE	BIGINT	The current size of the tablespace (in bytes)
DBFILE_SIZE	DOUBLE	The size of the database image files (in bytes)

Column name	Type	Description
DBFILE_COUNT_0	INTEGER	The number of database image files in file group #0
DBFILE_COUNT_1	INTEGER	The number of database image files in file group #1
TIMESTAMP	VARCHAR(64)	The time point at which the tablespace was created
ALLOC_PAGE_COUNT	BIGINT	The total number of pages in the tablespace
FREE_PAGE_COUNT	BIGINT	The number of free pages in the tablespace
RESTORE_TYPE	BIGINT	How to load the tablespace into memory
CURRENT_DB	INTEGER	A set of files that are the target for ping pong checkpointing
HIGH_LIMIT_PAGE	BIGINT	The maximum number of pages that the tablespace can have
PAGE_COUNT_PER_FILE	BIGINT	The number of pages per database image file
PAGE_COUNT_IN_DISK	INTEGER	The number of pages that exist on disk

3.2.43.1 Column Information

SPACE_STATUS

This is a value that indicates the tablespace status. Please refer to V$MEM_TABLESPACE_STATUS_DESC for details.

SPACE_SHM_KEY

This is a shared memory key, which is used when a tablespace is loaded into shared memory.

AUTOEXTEND_MODE

This indicates whether Autoextend mode is enabled. If it is set to 1, Autoextend mode is enabled, whereas if it is set to some other value, Autoextend mode is not enabled.

AUTOEXTEND_NEXTSIZE

When the tablespace is automatically extended, this indicates the size (in bytes) by which the tablespace is automatically extended.

MAXSIZE

This is the maximum size of the tablespace (in bytes).

CURRENT_SIZE

This is the current size of the tablespace (in bytes).

DBFILE_SIZE

This is the size of the database image files for the tablespace (in bytes).

DBFILE_COUNT_0

Because ALTIBASE HDB uses ping pong checkpointing, it maintains two sets of databases image files. This value indicates the number of files in file group #0, which is one of these sets.

DBFILE_COUNT_1

Because ALTIBASE HDB uses ping pong checkpointing, it maintains two sets of databases image files. This value indicates the number of files in file group #1, which is one of these sets.

TIMESTAMP

This timestamp value indicates the time point at which the tablespace was created.

ALLOC_PAGE_COUNT

This is the number of pages in the tablespace.

FREE_PAGE_COUNT

This is the number of free pages in the tablespace.

RESTORE_TYPE

This indicates how the tablespace is loaded into memory. It can have the following values:

Loading Method	Value	Description
RESTORE_TYPE_DYNAMIC	0	The tablespace is loaded into dynamic memory.
RESTORE_TYPE_SHM_CREATE	1	Shared memory is created, and then the tablespace is loaded into it.
RESTORE_TYPE_SHM_ATTACH	2	The tablespace is attached to shared memory. When the database has already been loaded into shared memory, the shared memory is attached to the process.

CURRENT_DB

This is the database image file group into which dirty pages (changed pages) are downloaded during checkpointing. It can be 0 or 1.

HIGH_LIMIT_PAGE

This is the maximum number of pages that the tablespace can have.

PAGE_COUNT_PER_FILE

This is the number of pages per database image file.

PAGE_COUNT_IN_DISK

This is the total number of pages in all database image files that exist on disk. ALTIBASE HDB increases the size of a database during checkpointing, rather than directly increasing the size of files on disk. Therefore, the number of database pages that exist in memory can be different from the number of pages on disk.

3.2.44 V$MEM_TABLESPACE_CHECKPOINT_PATHS

This view shows the directory path of the database image files in which changed pages (dirty pages) are recorded during checkpointing for a tablespace.

Column name	Type	Description
SPACE_ID	INTEGER	The tablespace identifier
CHECKPOINT_PATH	VARCHAR(512)	The directory in which the database image files are located

3.2.45 V$MEM_TABLESPACE_STATUS_DESC

This view provides descriptions of values that indicate the status of memory tablespace. These are the values that the SPACE_STATUS column in the V$MEM_TABLESPACES view can have.

Column name	Type	Description
STATUS	INTEGER	The status value of memory tablespace
STATUS_DESC	VARCHAR(64)	The description of the status value

3.2.45.1 Column Information

STATUS

This is the status value of memory tablespace.

STATUS_DESC

This is a description of the status value of memory tablespace.

The status values and corresponding descriptions are as follows:

STATUS_DESC	Description
OFFLINE	The tablespace is offline.
ONLINE	The tablespace is online.
DISCARDED	The tablespace has been discarded.
DROPPED	The tablespace has been deleted.
BACKUP	The tablespace is being backed up.
CREATING	The tablespace is being created.
DROPPING	A request has been made to delete the tablespace.
DROP_PENDING	The tablespace is being deleted.
SWITCHING_TO_OFFLINE	The tablespace is switching to offline status.
SWITCHING_TO_ONLINE	The tablespace is switching to online status.
BLOCK_BACKUP	The tablespace cannot be backed up. Because another operation is in progress, it is necessary to wait until the other operation is complete before backup can be performed.

3.2.46 V$MUTEX

This view displays statistical information about mutexes, which are related to concurrency control performed by ALTIBASE HDB processes.

Column	Data Type	Description
NAME	VARCHAR(64)	The name of the mutex
TRY_COUNT	INTEGER	The number of lock attempts
LOCK_COUNT	INTEGER	The number of successful lock attempts
MISS_COUNT	INTEGER	The number of waits resulting from missed lock attempts
SPIN_VALUE	INTEGER	This field is reserved for future use.
TOTAL_LOCK_TIME_US	BIGINT	The total amount of time this mutex has been locked (in microseconds)
MAX_LOCK_TIME_US	BIGINT	The maximum time elapsed while locking this mutex (in microseconds)

3.2.47 V$NLS_PARAMETERS

This view shows NLS (National Language Support)-related information for both the server and client for each session.

Column name	Type	Description
SESSION_ID	INTEGER	The session identifier
NLS_USE	VARCHAR(40)	The client character set
NLS_CHARACTERSET	VARCHAR(40)	The database character set
NLS_NCHAR_CHARACTERSET	VARCHAR(40)	The national character set
NLS_COMP	VARCHAR(7)	How characters are compared
NLS_NCHAR_CONV_EXCP	VARCHAR(7)	How to handle errors that arise when converting character sets
NLS_NCHAR_LITERAL_REPLACE	VARCHAR(7)	Whether to check for the presence of NCHAR literals within SQL statements

3.2.47.1 Column Information

SESSION_ID

This is the identifier of the session.

NLS_USE

This is the client character set. The default character set should be set when processing character data on the client. The character sets and related NLS_USE settings currently supported by ALTIBASE HDB are as follows:

Table 3-1 Character Sets Supported by ALTIBASE HDB

Language	Character Set	NLS_USE
English (default)	US7ASCII	US7ASCII, ASCII, ENGLISH
Korean	KSC-5601 Complete	KSC5601, KO16KSC5601, KOREAN
	MS Extended Complete	MS949, CP949, WINDOWS949
Japanese	EUC-JP (UNIX)	EUCJP
	Shift-JIS (Windows)	SHIFTJIS
Chinese	China	GB231280, ZHS16CGB231280, CHINESE
	Taiwan	BIG5, ZHT16BIG5, TAIWAN
Universal	Unicode (UTF-8)	UTF8, UNICODE

When storing data of a different character set than the database character set, it is important to consider convertibility and compatibility between the individual character sets. Please refer to the *Getting Started Guide* for more detailed information about multilingual support.

NLS_CHARACTERSET

This is the database character set used on the server.

NLS_NCHAR_CHARACTERSET

This is the national character set.

NLS_COMP

This indicates the order in which characters are compared according to how they appear in a dictionary of the language corresponding to the character set that was specified when the database was created. At present, this is useful only when Korean (KSC-5601 Completion or MS Extended Completion) is specified as the database character set.

BINARY: Characters are compared on the basis of the binary value.

ANSI: Characters are compared on the basis of the order in which they appear in a dictionary of that language

NLS_NCHAR_CONV_EXCP

This shows how errors are handled when the character set is converted.

NLS_NCHAR_LITERAL_REPLACE

This shows whether the client will check whether NCHAR literals exist within an SQL statement. If this is TRUE, the client always checks whether NCHAR literals exist, and convert the remainder of the SQL statement, other than the NCHAR literals, to the database character set. If this is FALSE, the client doesn't check this, and convert the entire SQL statement to to the database character set.

3.2.48 V$PLANTEXT

This view displays information about execution plans for SQL statements that are executed by the server.

Column	Data Type	Description
SID	INTEGER	The session identifier
STMT_ID	INTEGER	The statement identifier
PIECE	INTEGER	The serial number for the fragment of execution plan text
TEXT	VARCHAR(64)	A fragment of execution plan text

3.2.48.1 Column Information

SID

This is the identifier of the session.

STMT_ID

This is the identifier of the statement.

PIECE

A complete execution plan for one statement is divided into text fragments 64 bytes long and then saved. PIECE shows the serial numbers for these 64-byte fragments, starting from 0.

TEXT

This shows the contents of the 64-byte text fragment that is part of the execution plan statement.

3.2.49 V$PROCTEXT

This view displays information about stored procedures being used by the system.

Column	Data Type	Description
PROC_OID	BIGINT	The object identifier of a stored procedure
PIECE	INTEGER	The serial number for the stored procedure fragment
TEXT	VARCHAR(64)	A fragment of the stored procedure text

3.2.49.1 Column Information

PROC_OID

This is an OID, which is a unique object identifier for a stored procedure.

PIECE

The complete text for a stored procedure is divided into text fragments 64 bytes long and then saved. PIECE shows the serial numbers for these 64-byte fragments, starting from 0.

TEXT

This shows the contents of the 64-byte text fragment that is part of the stored procedure text.

3.2.50 V$PROPERTY

This view displays information about all internally set ALTIBASE HDB properties.

Column	Data Type	Description
NAME	VARCHAR(256)	The property name
STOREDCOUNT	INTEGER	The number of values set for the property
ATTR	BIGINT	The property attribute
MIN	VARCHAR(256)	The minimum value
MAX	VARCHAR(256)	The maximum value
VALUE1	VARCHAR(256)	The first value
VALUE2	VARCHAR(256)	The second value
VALUE3	VARCHAR(256)	The third value
VALUE4	VARCHAR(256)	The fourth value
VALUE5	VARCHAR(256)	The fifth value
VALUE6	VARCHAR(256)	The sixth value
VALUE7	VARCHAR(256)	The seventh value
VALUE8	VARCHAR(256)	The eighth value

3.2.50.1 Column Information

NAME

This is the name of the property.

STOREDCOUNT

STOREDCOUNT displays the number of values set in the property. A property can have up to eight duplicate values.

ATTR

This is the attribute of the property.

MIN

This is the minimum value that the property can have.

MAX

This is the maximum value that the property can have.

VALUE1 ~ 8

The actual values set for the property.

3.2.51 V$REPEXEC

This view displays information related to the replication manager.

Column	Data Type	Description
PORT	INTEGER	The port number currently being used
MAX_SENDER_COUNT	INTEGER	The maximum number of Sender threads
MAX_RECEIVER_COUNT	INTEGER	The maximum number of Receiver threads

3.2.51.1 Column Information

PORT

The number of the port through which the replication manager on the local server receives replication requests from the remote server.

MAX_SENDER_COUNT

This is the maximum number of replication Sender threads that can be created on the local server.

MAX_RECEIVER_COUNT

This is the maximum number of replication Receiver threads that can be created on the local server.

3.2.52 V$REPGAP

This shows the difference between the most recently created log record and the log record currently being processed by the replication Sender. Please note that this information is only available while the replication Sender thread is active.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
START_FLAG	BIGINT	Startup options
REP_LAST_SN	BIGINT	The sequence number of the last log record
REP_SN	BIGINT	The sequence number of the log record currently being sent

Column	Data Type	Description
REP_GAP	BIGINT	The difference between REP_LAST_SN and REP_SN
READ_LFG_ID	INTEGER	The log file group currently being read
READ_FILE_NO	INTEGER	The log file number currently being read
READ_OFFSET	INTEGER	The location currently being read

3.2.52.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

START_FLAG

This is a replication startup option for use when replication is started on the local server. The following values are possible:

- NORMAL: 0
- QUICK: 1
- SYNC: 2
- SYNC_ONLY: 3
- SYNC RUN: 4
- SYNC END: 5
- RECOVERY from Replication: 6
- OFFLINE: 7
- PARALLEL: 8

REP_LAST_SN

This is the sequence number of the log record that was most recently written in response to a transaction on the local server.

REP_SN

This is the sequence number of the log record that is currently being sent by the replication Sender on the local server.

REP_GAP

This shows the interval between the log sequence numbers of REP_LAST_SN and REP_SN. In other words, this is the interval between the log record that was most recently written due to a transaction on the local server and the log record that is currently being sent by the replication Sender thread.

READ_LFG_ID

This indicates the log file group that is currently being read for transmission.

READ_FILE_NO

This indicates the number of the log file that is currently being read.

READ_OFFSET

This indicates the location in the log file that is currently being read.

3.2.53 V$REPGAP_PARALLEL

This view shows the difference between the most recently created log record and the log record currently being processed by replication Sender threads working in parallel. Please note that this information is only available when multiple replication Sender threads are working in parallel.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication
CURRENT_TYPE	VARCHAR(9)	The type of the replication Sender thread
REP_LAST_SN	BIGINT	The last log file number
REP_SN	BIGINT	The sequence number of the log record currently being sent
REP_GAP	BIGINT	The gap between REP_LAST_SN and REP_SN
READ_LFG_ID	INTEGER	The identifier of the log file group currently being read
READ_FILE_NO	INTEGER	The log file number currently being read
READ_OFFSET	INTEGER	The current reading offset
PARALLEL_ID	INTEGER	The identifier of one of several threads working in parallel for one Sender

3.2.53.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

CURRENT_TYPE

This can have one of the following values, which denote the current status of the replication Sender thread.

- NORMAL: This means that the Sender thread analyzes transaction logs and converts them to XLOGs on the active server. The Sender thread then transfers the XLOGs to a standby server.
- QUICK: This value can be returned when replication was started with the QUICKSTART option, and indicates the state in which the starting location is being changed so that the Sender thread will ignore old logs and start sending from the most recent log. After the starting location is changed, NORMAL will be returned, rather than this value.
- SYNC: This value can be returned when replication was started with the SYNC option. After synchronization is complete, NORMAL (LAZY mode) or PARALLEL (EAGER mode) will be returned, rather than this value.
- SYNC_ONLY: This value can be returned when replication was started with the SYNC ONLY option. After synchronization is complete, the Sender thread will be terminated.
- RECOVERY: This value indicates that the Sender thread is running in order to restore data that were lost on another server.
- OFFLINE: This value indicates that the Sender thread is running in order to read logs on the active server when the active server is offline and apply them to the standby server.
- PARALLEL: This value indicates that several Sender threads are sending XLOGs pertaining to the table(s) that is (or are) being replicated in parallel. This value can be returned when replication was started in EAGER mode with the PARALLEL option. It is different from the PARALLEL option which can be specified when starting replication with the SYNC or SYNC_ONLY option.

REP_LAST_SN

This is the most recent log record sequence number on the local server.

REP_SN

This is the sequence number of the log record that is currently being sent by the replication Sender on the local server.

REP_GAP

This is the difference between the log serial number returned by REP_LAST_SN and that returned by REP_SN. In other words, this is the gap between the log record that was most recently written by a transaction on the local server and the log record that is currently being sent by the replication Sender thread.

READ_LFG_ID

This indicates the group of log files that is currently being read for transmission.

READ_FILE_NO

This indicates the number of the log file that is currently being read.

READ_OFFSET

This indicates the current location in the log file that is currently being read.

PARALLEL_ID

This is the identifier of one of several threads working in parallel for one Sender.

3.2.54 V$REPLOGBUFFER

This view displays information about the state of the log buffer used by the replication Sender while the replication Sender thread is working.

Column name	Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
BUFFER_MIN_SN	BIGINT	The lowest log sequence number in the buffer that is being used by the replication Sender
READ_SN	BIGINT	The sequence number of the log record to be read next by the replication Sender thread
BUFFER_MAX_SN	BIGINT	The highest log sequence number in the buffer that is being used by the replication Sender

3.2.54.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

BUFFER_MIN_SN

This is the lowest of the sequence numbers of log records saved in the log buffer that is used for replication.

READ_SN

This is the sequence number of the log record that is to be read next by the replication Sender thread in the log buffer that is being used for replication.

BUFFRT_MAX_SN

This is the highest of the sequence numbers of log records saved in the log buffer that is being used for replication.

3.2.55 V$REPOFFLINE_STATUS

This view shows the status of offline replication.

Column name	Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
STATUS	BIGINT	The status of offline replication execution
SUCCESS_TIME	INTEGER	The time taken for offline replication to execute successfully

3.2.55.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

STATUS

This is the status of offline replication.

0: offline replication has not been started

1: offline replication has been started

2: offline replication has completed

3: offline replication failed

SUCCESS_TIME

This is the time point at which the most recent successful execution of offline replication occurred. It is based on the system time. In the case where replication was successfully started and completed, it is the time taken for replication to complete, and is 0 otherwise.

3.2.56 V$REPRECEIVER

This view displays information about the replication Receiver.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
MY_IP	VARCHAR(64)	The IP address of the local sever
MY_PORT	INTEGER	The port number on the local server
PEER_IP	VARCHAR(64)	The IP address on the remote server

Column	Data Type	Description
PEER_PORT	INTEGER	The port number on the remote server
APPLY_XSN	BIGINT	The XSN currently being processed
INSERT_SUCCESS_COUNT	BIGINT	The number of INSERT log records successfully applied to the local database by the replication Receiver thread
INSERT_FAILURE_COUNT	BIGINT	The number of INSERT log records that could not be applied to the local database by the replication Receiver thread
UPDATE _SUCCESS_COUNT	BIGINT	The number of UPDATE log records successfully applied to the local database by the replication Receiver thread
UPDATE_FAILURE_COUNT	BIGINT	The number of UPDATE log records that could not be applied to the local database by the replication Receiver thread
DELETE_SUCCESS_COUNT	BIGINT	The number of DELETE log records successfully applied to the local database by the replication Receiver thread
DELETE_FAILURE_COUNT	BIGINT	The number of DELETE log records that could not be applied to the local database by the replication Receiver thread

3.2.56.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

MY_IP

This is the IP address of the local server.

MY_PORT

This is the port number being used by the Receiver thread on the local server.

PEER_IP

This is the IP address of the remote server.

PEER_PORT

This is the port number being used by the Sender thread on the remote server.

APPLY_XSN

This shows the XLog sequence number (XSN) of the XLog that was sent by the Sender thread on the remote server and is being used by the Receiver thread on the local server.

INSERT_SUCCESS_COUNT

This is the number of INSERT log records that were successfully applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

INSERT_FAILURE_COUNT

This is the number of INSERT log records (including conflicts) that could not be applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

UPDATE_SUCCESS_COUNT

This is the number of UPDATE log records that were successfully applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

UPDATE_FAILURE_COUNT

This is the number of UPDATE log records (including conflicts) that could not be applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

DELETE_SUCCESS_COUNT

This is the number of DELETE log records that were successfully applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

DELETE_FAILURE_COUNT

This is the number of DELETE log records (including conflicts) that could not be applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

3.2.57 V$REPRECEIVER_COLUMN

This view shows information about columns that are replication targets used by the replication Receiver.

Column name	Type	Description
REP_NAME	VARCHAR(40)	The name of the replication
USER_NAME	VARCHAR(40)	The user name
TABLE_NAME	VARCHAR(40)	The table name
PARTITION_NAME	VARCHAR(40)	The name of the partition
COLUMN_NAME	VARCHAR(40)	The column name

3.2.57.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

USER_NAME

This is the user name of the owner of the table that is the target of replication on the local server. Its value corresponds to a USER_NAME in the SYS_USERS_ meta table.

TABLE_NAME

This is the name of a table that is the target of replication on the local server. It corresponds to a TABLE_NAME in the SYS_TABLES_ meta table.

PARTITION_NAME

This is the name of the partition that is the target for replication on the local server.

COLUMN_NAME

This is the name of the column that is the target of replication on the local server.

3.2.58 V$REPRECEIVER_PARALLEL

This view displays information about replication Receiver threads working in parallel.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
MY_IP	VARCHAR(64)	The IP address of the local server

Column	Data Type	Description
MY_PORT	INTEGER	The port number on the local server
PEER_IP	VARCHAR(64)	The IP address of the remote server
PEER_PORT	INTEGER	The port number on the remote server
APPLY_XSN	BIGINT	The XSN currently being processed
INSERT_SUCCESS_COUNT	BIGINT	The number of INSERT transactions successfully applied to the local database by the replication Receiver thread.
INSERT_FAILURE_COUNT	BIGINT	The number of INSERT transactions that could not be applied to the local database by the replication Receiver thread.
UPDATE _SUCCESS_COUNT	BIGINT	The number of UPDATE transactions successfully applied to the local database by the replication Receiver thread.
UPDATE_FAILURE_COUNT	BIGINT	The number of UPDATE transactions that could not be applied to the local database by the replication Receiver thread.
DELETE_SUCCESS_COUNT	BIGINT	The number of DELETE transactions successfully applied to the local database by the replication Receiver thread.
DELETE_FAILURE_COUNT	BIGINT	The number of DELETE transactions that could not be applied to the local database by the replication Receiver thread.
PARALLEL_ID	INTEGER	The identifier of one of several replication Receiver threads working in parallel

3.2.58.1 Column Information

REP_NAME

This is the name of the replication object.

MY_IP

This is the IP address of the local server.

MY_PORT

This is the port number used by the Receiver on the local server.

PEER_IP

This is the IP address of the remote server.

PEER_PORT

This is the port number used by the Sender on the remote server.

APPLY_XSN

This shows the XLog sequence number of the XLog that was sent by a Sender thread on the remote server and is being applied by the Receiver thread on the local server.

INSERT_SUCCESS_COUNT

This is the number of INSERT transactions that were successfully applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

INSERT_FAILURE_COUNT

This is the number of INSERT transactions (including conflicts) that could not be applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

UPDATE_SUCCESS_COUNT

This is the number of UPDATE transactions that were successfully applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

UPDATE_FAILURE_COUNT

This is the number of INSERT transactions (including conflicts) that could not be applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

DELETE_SUCCESS_COUNT

This is the number of DELETE transactions that were successfully applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

DELETE_FAILURE_COUNT

This is the number of INSERT transactions (including conflicts) that could not be applied to the local database by the replication Receiver thread.

This number is not dependent on whether statements are committed or rolled back. In other words, if a statement is rolled back, this number is not decreased.

PARALLEL_ID

This is the identifier of one of several replication Receivers having the same name.

3.2.59 V$REPRECEIVER_TRANSTBL

This view displays information about the replication Receiver's transaction table.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
LOCAL_TID	INTEGER	The local transaction identifier
REMOTE_TID	INTEGER	The remote transaction identifier
BEGIN_FLAG	INTEGER	Not currently used
BEGIN_SN	BIGINT	The first log record sequence number of the transaction

3.2.59.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

LOCAL_TID

This is the identifier of the transaction that is being executed on the local server.

REMOTE_TID

This is the identifier of the transaction that is executed on the remote server. It may or may not have already finished being executed.

3.2.60 V$REPRECEIVER_TRANSTBL_PARALLEL

This view displays information about transaction tables used by multiple replication Receiver threads working in parallel.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
LOCAL_TID	INTEGER	The local transaction identifier

Column	Data Type	Description
REMOTE_TID	INTEGER	The remote transaction identifier
BEGIN_FLAG	INTEGER	Not currently used
BEGIN_SN	BIGINT	The first log record sequence number of the transaction
PARALLEL_ID	INTEGER	The identifier of one of several Receivers having the same name

3.2.60.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

LOCAL_TID

This is the identifier of a transaction that is being executed on the local server.

REMOTE_TID

This is the identifier of a transaction that is executed on the remote server. It may or may not have already finished being executed.

PARALLEL_ID

This is the identifier of one of several replication Receivers working in parallel.

3.2.61 V$REPRECOVERY

This view shows information pertaining to recovery using replication.

Column name	Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
STATUS	INTEGER	The present status of recovery 1: Generating recovery information 2: Recovery request pending 3: Recovery in progress
START_XSN	BIGINT	The first SN sent for recovery
XSN	BIGINT	The SN currently being sent for recovery
END_XSN	BIGINT	The last SN sent for recovery
RECOVERY_SENDER_IP	VARCHAR(64)	The IP address of the Sender for recovery of the local server

Column name	Type	Description
PEER_IP	VARCHAR(64)	The IP address of the Receiver for recovery of the remote server
RECOVERY_SENDER_PORT	INTEGER	The port number used by the Sender for recovery of the local server
PEER_PORT	INTEGER	The port number used by the Receiver for recovery of the remote server

3.2.61.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

STATUS

This is the present status of replication Sender threads on the local server.

1: Recovery information is being generated

2: A recovery request is waiting

3: Recovery is underway

START_XSN

This shows the sequence number of the first log record to be sent by the Sender thread for recovery of the local server.

XSN

This shows the sequence number of the log record currently being sent by the Sender thread for recovery of the local server.

END_XSN

This shows the sequence number of the last log record to be sent by the Sender thread for recovery of the local server.

RECOVERY_SENDER_IP

This is the IP address of the Sender for recovery of the local server.

PEER_IP

This is the IP address of the remote server for recovery of the remote server.

RECOVERY_SENDER_PORT

This is the port number being used by the Sender thread for recovery of the local server.

PEER_PORT

This is the port number being used by the Receiver thread for recovery of the remote server.

3.2.62 V$REPSENDER

This view displays information about the replication Sender.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
START_FLAG	BIGINT	A flag indicating startup options
NET_ERROR_FLAG	BIGINT	A flag indicating a network error
XSN	BIGINT	The sequence number of the log record being sent
COMMIT_XSN	BIGINT	The sequence number of the committed log record that was most recently read by the Sender
STATUS	BIGINT	The current status of the replication Sender
SENDER_IP	VARCHAR(64)	The IP address of the Sender
PEER_IP	VARCHAR(64)	The IP address of the remote server
SENDER_PORT	INTEGER	The port number used by the Sender
PEER_PORT	INTEGER	The port number used by the Receiver on the remote server
READ_LOG_COUNT	BIGINT	The number of logs that have been read
SEND_LOG_COUNT	BIGINT	The number of logs that have been read and sent
REPL_MODE	VARCHAR(7)	The replication mode specified by the user
ACT_REPL_MODE	VARCHAR(7)	The actual replication mode

3.2.62.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

START_FLAG

This is a flag indicating the replication startup options on the local server. It can have the following values:

- NORMAL: 0
- QUICK: 1
- SYNC: 2
- SYNC_ONLY: 3
- SYNC RUN: 4
- SYNC END: 5
- RECOVERY USING REPLICATION: 6
- OFFLINE: 7
- PARALLEL: 8

NET_ERROR_FLAG

This indicates whether a network error has occurred. The default value is 0; 1 indicates that an error has occurred.

XSN

This is the sequence number of the log record that is currently being sent by the replication Sender thread on the local server.

COMMIT_XSN

This is the sequence number of the committed log record that was most recently read by the replication Sender.

STATUS

This is the current status of the replication Sender on the local server. It can have the following values:

0: STOP

1: RUN

2: RETRY

3: FAILBACK NORMAL

4: FAILBACK MASTER

5: FAILBACK SLAVE

6: SYNC

SENDER_IP

This is the IP address of the local server.

PEER_IP

This is the IP address of the remote server.

SENDER_PORT

This is the port number used by the replication Sender thread on the local server.

PEER_PORT

This is the port number used by the replication Receiver thread on the remote server.

READ_LOG_COUNT

This is the number of log records that have been read by the Sender thread on the local server.

SEND_LOG_COUNT

This is the number of log records that have been read and sent by the Sender thread on the local server.

REPL_MODE

This is the replication mode set by the user, which can be set to LAZY or EAGER.

For more detailed information about replication modes, please refer to the *Replication Manual*.

ACT_REPL_MODE

This is the actual replication mode, which may differ from REPL_MODE.

When the replication mode has been set to EAGER, if the value of the REPLICATION_SERVICE_WAIT_MAX_LIMIT property is exceeded due to some error, replication continues in LAZY mode.

Other than this case, the value is the same as that of REPL_MODE.

Please refer to the *General Reference* for a more detailed explanation of the REPLICATION_SERVICE_WAIT_MAX_LIMIT property.

3.2.63 V$REPSENDER_PARALLEL

This view displays information about replication Sender threads working in parallel.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object

Column	Data Type	Description
CURRENT_TYPE	VARCHAR(9)	The type of the replication Sender thread
NET_ERROR_FLAG	BIGINT	A flag indicating a network error
XSN	BIGINT	The sequence number of the log record currently being sent
COMMIT_XSN	BIGINT	The sequence number of the most recently committed log record
STATUS	BIGINT	The current status of the replication Sender
SENDER_IP	VARCHAR(64)	The IP address of the Sender
PEER_IP	VARCHAR(64)	The IP address of the remote server
SENDER_PORT	INTEGER	The port number used by the Sender
PEER_PORT	INTEGER	The port number used by the Receiver on the remote server
READ_LOG_COUNT	BIGINT	The number of logs that have been read
SEND_LOG_COUNT	BIGINT	The number of logs that have been read and transmitted
REPL_MODE	VARCHAR(7)	The current replication mode
PARALLEL_ID	INTEGER	The identifier of one of several replication Senders having the same name

3.2.63.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

CURRENT_TYPE

Please refer to the description of the CURRENT_TYPE column in the V$REPGAP_PARALLEL performance view.

NET_ERROR_FLAG

This indicates whether a network error has occurred. The default value is 0; 1 indicates that an error has occurred.

XSN

This is the sequence number of the log record that is currently being sent by the corresponding replication Sender thread on the local server.

COMMIT_XSN

This is the sequence number of the committed log record that was most recently read by this Sender thread.

STATUS

This is the current status of the replication Sender on the local server. It can have the following values:

0: STOP

1: RUN

2: RETRY

3: FAILBACK NORMAL

4: FAILBACK MASTER

5: FAILBACK SLAVE

6: SYNC

SENDER_IP

This is the IP address of the local server.

PEER_IP

This is the IP address of the remote server.

SENDER_PORT

This is the port number used by this replication Sender thread on the local server.

PEER_PORT

This is the port number used by the corresponding replication Receiver thread on the remote server.

READ_LOG_COUNT

This is the number of log records read by this Sender thread on the local server.

SEND_LOG_COUNT

This is the number of log records read and transmitted by this Sender thread on the local server.

REPL_MODE

This is the replication mode. It can be set to LAZY or EAGER.

For more detailed information about replication modes, please refer to the *Replication Manual*.

PARALLEL_ID

This is the identifier of one of several replication Senders working in parallel.

3.2.64 V$REPSENDER_TRANSTBL

This view displays information about the replication Sender's transaction table.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
START_FLAG	BIGINT	A flag indicating startup options
LOCAL_TID	INTEGER	The local transaction identifier
REMOTE_TID	INTEGER	The remote transaction identifier
BEGIN_FLAG	INTEGER	Indicates whether 'BEGIN' acknowledgement has been sent
BEGIN_SN	BIGINT	The first log record sequence number of the transaction

3.2.64.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

START_FLAG

Please refer to the description of the START_FLAG column in the V$REPSENDER performance view.

LOCAL_TID

This is the identifier of the transaction being executed on the local server.

REMOTE_TID

This is the identifier of the transaction being executed on the remote server.

3.2.65 V$REPSENDER_TRANSTBL_PARALLEL

This view displays information about transaction tables used by multiple replication Sender threads working in parallel.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object
CURRENT_TYPE	VARCHAR(9)	The type of the replication Sender thread
LOCAL_TID	INTEGER	The local transaction identifier
REMOTE_TID	INTEGER	The remote transaction identifier
BEGIN_FLAG	INTEGER	Indicates whether 'BEGIN' acknowledgement has been sent
BEGIN_SN	BIGINT	The first log record sequence number of the transaction
PARALLEL_ID	INTEGER	The identifier of one of several replication Senders working in parallel

3.2.65.1 Column Information

REP_NAME

This is the name of the replication object.

CURRENT_TYPE

Please refer to the description of the CURRENT_TYPE column in the V$REPGAP_PARALLEL performance view.

LOCAL_TID

This is the identifier of the transaction being executed on the local server.

REMOTE_TID

This is the identifier of the transaction being executed on the remote server.

PARALLEL_ID

This is the identifier of one of several replication Sender threads working in parallel.

3.2.66 V$REPSYNC

This view displays information about tables that are synchronized using replication.

Column	Data Type	Description
REP_NAME	VARCHAR(40)	The name of the replication object

Column	Data Type	Description
SYNC_TABLE	VARCHAR(40)	The name of the table to be synchronized
SYNC_PARTITION	VARCHAR(40)	The name of the partition to be synchronized
SYNC_RECORD_COUNT	BIGINT	The number of records that have been synchronized on the remote server
SYNC_SN	BIGINT	Not currently used

3.2.66.1 Column Information

REP_NAME

This is the name of the replication object on the local server.

SYNC_TABLE

This is the name of the table that is the target for synchronization.

SYNC_PARTITION

This is the name of the partition that is the target for synchronization.

SYNC_RECORD_COUNT

When data in replication target tables on the local server are synchronized with those on the remote server, the data are synchronized in batches of records, the size of which is specified in the ALTIBASE HDB REPLICATION_SYNC_TUPLE_COUNT property.

While synchronization is underway, this is the number of records that have been synchronized.

A value of -1 indicates that synchronization is complete.

3.2.67 V$SEGMENT

This view shows information about segments that make up disk tables and disk indexes, including their status, kind, and the index to which they are allocated.

Column	Data Type	Description
SPACE_ID	INTEGER	The tablespace identifier
TABLE_OID	BIGINT	The object identifier of the table header
SEGMENT_PID	INTEGER	The identifier of the page in which the segment is stored
SEGMENT_TYPE	VARCHAR(7)	The type of segment
SEGMENT_STATE	VARCHAR(7)	The status of the segment

Column	Data Type	Description
EXTENT_TOTAL_COUNT	BIGINT	The total number of extents assigned to the segment

3.2.67.1 Column Information

SPACE_ID

This is identifier of the tablespace in which the segment exists.

TABLE_OID

This is the object identifier of the table that uses the segment.

SEGMENT_PID

This is the identifier of the page in which the segment header is stored.

SEGMENT_TYPE

INDEX: This indicates that the segment is an index segment.

TABLE: This indicates that the segment is an table segment.

TSSEG: This indicates that the segment is a TSS segment.

UDSEG: This indicates that the segment is an undo segment.

SEGMENT_STATE

USED: This indicates that the segment is being used.

FREE: This indicates that the segment is empty.

EXTENT_TOTAL_COUNT

This is the total number of extents allocated to the segment.

3.2.68 V$SEQ

This view displays sequence-related information.

Column	Data Type	Description
SEQ_OID	BIGINT	The object identifier of the sequence
CURRENT_SEQ	BIGINT	The current value of the sequence
START_SEQ	BIGINT	The starting value of the sequence

Column	Data Type	Description
INCREMENT_SEQ	BIGINT	The increment value of the sequence
CACHE_SIZE	BIGINT	The size of the cache
MAX_SEQ	BIGINT	The maximum sequence value
MIN_SEQ	BIGINT	The minimum sequence value
IS_CYCLE	CHAR(7)	Indicates whether the sequence is cyclical

3.2.68.1 Column Information

SEQ_OID

This is a unique sequence identifier, which is assigned internally by the system when the sequence is created. It has the same value as a TABLE_OID in the SYS_TABLES_ meta table, for which the value in the TABLE_TYPE column will be 'S' (Sequence).

CURRENT_SEQ

This is the current sequence value.

START_SEQ

This is the sequence value that was specified when the sequence was first created.

INCREMENT_SEQ

This is value by which the sequence is incremented.

MAX_SEQ

This is the maximum value that can be generated using the sequence.

MIN_SEQ

This is the minimum value that can be generated using the sequence.

IS_CYCLE

When the sequence reaches its maximum possible value, this indicates whether the sequence will cycle and generate values starting from the minimum value again.

YES: The sequence cycles

NO: The sequence does not cycle. If the sequence has reached the maximum possible value and an attempt is made to generate another sequence value, an error occurs.

3.2.69 V$SERVICE_THREAD

This view displays information about service threads related to multiplexing.

Column	Data Type	Description
ID	INTEGER	The service thread identifier
TYPE	VARCHAR(7)	The service thread access method
STATE	VARCHAR(10)	The current status of the service thread
SESSION_ID	INTEGER	The identifier of the session in which the service thread is executed
RUN_MODE	VARCHAR(9)	The mode of execution of the service thread
STATEMENT_ID	INTEGER	The identifier of the statement being executed by the service thread
START_TIME	INTEGER	The time at which the service thread was created
EXECUTE_TIME	BIGINT	The time taken for the service thread to execute a query
TASK_COUNT	INTEGER	The number of sessions being handled by the service thread
READY_TASK_COUNT	INTEGER	The number of sessions waiting for service threads to process their requests

A thread in a server that receives and fulfills a request (query) from a client is called a service thread. When multiple clients connect to a server and execute queries, if a service thread is created for each client session, it may result in deterioration of performance.

Therefore, it is better to create only the number of service threads that is optimized for the server and can meet client requests. This is called service thread multiplexing. ALTIBASE HDB is designed to maintain the optimal number of service threads all of the time by dynamically adding or deleting service threads. However, the minimum number of service threads specified in the MULTIPLEXING_THREAD_COUNT property is always maintained.

Column Information

ID

This is the identifier of the service thread. This is an identifier that is managed within ALTIBASE HDB, rather than a system thread identifier (that is, a Light Weight Process ID).

TYPE

This shows the service thread connection method. It can have the following values:

SOCKET: Connection via TCP or Unix Domain (UDP)

IPC: Connection via IPC

STATE

This indicates the current status of the service thread. It can have the following values:

NONE: The service thread is being initialized.

POLL: The service thread is waiting for an event.

QUEUE-WAIT: The service thread is waiting in the queue.

EXECUTE: The service thread is executing a statement.

UNKNOWN: The status of the service thread cannot be determined.

RUN_MODE

This shows the mode of execution of the service thread. It can be either SHARED or DEDICATED.

SHARED: When multiple database tasks (connections) are allocated to a single service thread, this service thread is said to be multiplexed, and processes all of the database tasks.

DEDICATED: One database task (connection) is allocated to one service thread, and uses the thread exclusively.

Switching the operating mode of a service thread is currently possible only for queue-related tasks. The mode can only be switched from SHARED mode to DEDICATED mode.

STATEMENT_ID

This is the identifier of the SQL statement that is currently being executed by the service thread.

START_TIME

This is the time point at which the service thread was created. It is based on system time.

Unit: seconds

EXECUTE_TIME

This is the amount of time that the service thread has taken to execute the current query.

Unit: microseconds

TASK_COUNT

This is the total number of sessions that are assigned to the service thread.

READY_TASK_COUNT

This is the number of sessions that are waiting for their requests to be processed by service threads.

3.2.70 V$SESSION

This view displays internally generated information about client sessions.

Column name	Type	Description
ID	INTEGER	The session identifier
TRANS_ID	BIGINT	The identifier of the transaction currently being executed in the session
TASK_STATE	VARCHAR(11)	The task status
COMM_NAME	VARCHAR(64)	Connection information
XA_SESSION_FLAG	INTEGER	The XA session flag
XA_ASSOCIATE_FLAG	INTEGER	The XA associate flag
QUERY_TIME_LIMIT	INTEGER	See below
FETCH_TIME_LIMIT	INTEGER	See below
UTRANS_TIME_LIMIT	INTEGER	See below
IDLE_TIME_LIMIT	INTEGER	See below
IDLE_START_TIME	INTEGER	See below
ACTIVE_FLAG	INTEGER	The active transaction flag
OPENED_STMT_COUNT	INTEGER	The number of opened statements
CLIENT_PACKAGE_VERSION	VARCHAR(40)	The client package version
CLIENT_PROTOCOL_VERSION	VARCHAR(40)	The client communication protocol version
CLIENT_PID	BIGINT	The client process ID
CLIENT_TYPE	VARCHAR(40)	The type of the client
CLIENT_APP_INFO	VARCHAR(128)	The type of the client application
CLIENT_NLS	VARCHAR(40)	The client character set
DB_USERNAME	VARCHAR(40)	The database user name
DB_USERID	INTEGER	The database user identifier
DEFAULT_TBSID	BIGINT	The user's default tablespace identifier
DEFAULT_TEMP_TBSID	BIGINT	The user's default temporary tablespace identifier
SYSDBA_FLAG	INTEGER	Indicates whether the connection was made as sysdba
AUTOCOMMIT_FLAG	INTEGER	The autocommit flag

Column name	Type	Description
SESSION_STATE	VARCHAR(13)	The status of the session
ISOLATION_LEVEL	INTEGER	The isolation level
REPLICATION_MODE	INTEGER	The replication mode
TRANSACTION_MODE	INTEGER	The transaction mode
COMMIT_WRITE_WAIT_MODE	INTEGER	See below
OPTIMIZER_MODE	INTEGER	The optimization mode
HEADER_DISPLAY_MODE	INTEGER	Indicates whether only the column names are output, or whether the table names are output along with the column names when the results of a SELECT query are output. 0: The table names are displayed along with the column names. 1: Only the column names are output.
CURRENT_STMT_ID	INTEGER	The identifier of the current statement
STACK_SIZE	INTEGER	The size of the stack
DEFAULT_DATE_FORMAT	VARCHAR(64)	The default date format e.g.) DD-MON-RRRR
TRX_UPDATE_MAX_LOGSIZE	BIGINT	The maximum size of the DML Log
PARALLEL_DML_MODE	INTEGER	Deprecated
LOGIN_TIME	INTEGER	The amount of time the client has been logged in

3.2.70.1 Column Information

ID

This is the unique identifier of a currently connected session.

TRANS_ID

This is the identifier of the transaction currently being executed in the session. If no transaction is currently underway, the value of -1 will be shown.

TASK_STATE

This indicates the status of the current task. It can have the following values:

STATE	Description
WAITING	The state in which the task is waiting for a request from a client
READY	The state in which the task has been received from a client and is waiting for a thread to be assigned to it
EXECUTING	The state in which a thread has been assigned to the task and is processing it
QUEUE WAIT	The state in which the task is waiting to be queued. After the task is queued, it is eventually dequeued.
QUEUE READY	The state in which the task has been queued. It will be dequeued once a thread has been assigned to it.
UNKNOWN	The state of the task cannot be determined.

COMM_NAME

This is the client connection information, the format of which varies depending on which communication protocol (TCP/IP, UNIX domain sockets, or IPC) is used. In the case of TCP/IP, this information also includes the client IP address and port number.

XA_SESSION_FLAG

Indicates whether the current session is an XA session.

0: NON-XA (not an XA session)

XA_ASSOCIATE_FLAG

This shows the state of association between the session and the global transaction.

QUERY_TIME_LIMIT

This is the query timeout value for the current session.

FETCH_TIME_LIMIT

This is the fetch timeout value for the current session.

UTRANS_TIME_LIMIT

This is the update transaction timeout value for the current session.

IDLE_TIME_LIMIT

This is the idle timeout value for the current session.

IDLE_START_TIME

This shows the time at which the session entered an Idle state.

ACTIVE_FLAG

If the session is executing a statement, the value of 1 is shown. However, if the session has merely connected, or has finished committing or rolling back a transaction, a value of 0 will be shown.

OPENED_STMT_COUNT

This shows the number of statements that are currently being executed by the session.

CLIENT_PACKAGE_VERSION

This is the package version of the connected client.

CLIENT_PROTOCOL _VERSION

This is the communication protocol version being used by the connected client.

CLIENT_PID

This is the process ID of the connected client. This value is not available for Java applications.

CLIENT_TYPE

This is a string that indicates the type of the connected client.

It consists of the following:

```
CLIENT_TYPE ::= app-type hypen word-size endian
    app-type ::= CLI | WIN_ODBC | UNIX_ODBC
    hypen ::= -
    word-size ::= 32 | 64
    endian ::= BE | LE
BE : Big Endian, LE : Little Endian

Ex.) CLI-32LE
 UNIX_ODBC-32BE
```

CLIENT_APP_INFO

This is information about the connected client application. This information is set by the client application.

CLIENT_NLS

This indicates the character set in use on the connected client.

DB_USERNAME

This is the name of the database user being used on the connected client.

DB_USERID

This is a numerically expressed user identifier, used by ALTIBASE HDB to distinguish between users.

DEFAULT_TBSID

This is the identifier of the default tablespace for the user.

DEFAULT_TEMP_TBSID

This is the identifier of the default temporary tablespace for the user.

SYSDBA_FLAG

This indicates whether or not the session is connected in sysdba mode.

1: sysdba mode

AUTOCOMMIT_FLAG

This indicates whether AUTOCOMMIT is active for the connected session.

0: the connected session is not in AUTOCOMMIT mode

1: the connected session is in AUTOCOMMIT mode

SESSION_STATE

STATE	Description
INIT	The state in which the session is waiting for a request from a client
AUTH	The state in which user authorization is complete
SERVICE READY	The state in which service is ready (The state “A transaction cannot be created” is returned only for XA sessions.)
SERVICE	The service state
END	The state of normal completion (COMMIT in the case where there is a transaction)
ROLLBACK	The state of abnormal termination (ROLLBACK in the case where there is a transaction) This occurs when a client is disconnected or a server forcibly disconnects a session.
UNKNOWN	The state cannot be determined

ISOLATION_LEVEL

This indicates the isolation level for the session.

REPLICATION_MODE

This indicates the replication mode for the session.

0: DEFAULT

16: EAGER

48: LAZY

64: NONE

TRANSACTION_MODE

This indicates the transaction mode for the session.

0: READ/WRITE

4: READ ONLY

COMMIT_WRITE_WAIT_MODE

0: When a transaction is committed, do not wait until the logs are written to disk.

1: When a transaction is committed, wait until the logs are written to disk.

OPTIMIZER_MODE

This indicates the optimization mode that has been set for the session.

1: the optimization mode is rule-based

0: the optimization mode is cost-based

CURRENT_STMT_ID

This is the identifier of the statement that is currently being executed.

STACK_SIZE

This is the size of the stack for the query processor that has been set for the current session.

DEFAULT_DATE_FORMAT

This is the default date format that has been set for the session. (Please refer to the description of the Datetime data type in Chapter1: Data Types.)

e.g.: DD-MON-RRRR

TRX_UPDATE_MAX_LOGSIZE

This is the maximum size of logs that can be generated by a single DML statement.

LOGIN_TIME

This indicates the amount of time that the client has been logged in.

3.2.71 V$SESSION_EVENT

This view shows cumulative statistical wait information about all wait events for each session that is currently connected to an Altibase database.

Column name	Type	Description
SID	INTEGER	The identifier of the session
EVENT	VARCHAR(128)	The name of the wait event
TOTAL_WAITS	BIGINT	The total number of waits related to the wait event
TOTAL_TIMEOUTS	BIGINT	The total number of times that a resource could not be accessed after the specified time
TIME_WAITED	BIGINT	The total amount of time spent waiting for the wait event (in milliseconds)
AVERAGE_WAIT	BIGINT	The average amount of time spent waiting for the wait event (in milliseconds)
MAX_WAIT	BIGINT	The maximum time spent waiting for the wait event (in milliseconds)
TIME_WAITED_MICRO	BIGINT	The total amount of time spent waiting for the wait event (in microseconds)
EVENT_ID	INTEGER	The identifier of the wait event
WAIT_CLASS_ID	INTEGER	The identifier of the class of the wait event
WAIT_CLASS	VARCHAR(128)	The name of the class of the wait event

3.2.71.1 Column Information

SID

This is the identifier of a waiting session.

EVENT

This is the name of the wait event.

TOTAL_WAITS

This is the total number of waits related to the wait event.

TOTAL_TIMEOUTS

This is the number of failures to gain access to the requested resource even after the specified time has elapsed.

TIME_WAITED

This is the total time spent waiting for this wait event (in milliseconds).

AVERAGE_WAIT

This is the average amount of time spent waiting for the wait event (in milliseconds).

MAX_WAIT

This is the maximum time spent waiting for the wait event (in milliseconds).

TIME_WAITED_MICRO

This is the total amount of time spent waiting for this wait event (in microseconds).

EVENT_ID

This is the identifier of the wait event.

WAIT_CLASS_ID

This is the identifier of the wait class in which the wait event is classified.

WAIT_CLASS

This is the name of the class in which the wait event is classified.

3.2.72 V$SESSION_WAIT

This view displays information about wait events for all currently connected sessions. This view does not provide information about wait events related to sessions that are no longer connected.

Column name	Type	Description
SID	INTEGER	The identifier of the session
SEQNUM	INTEGER	The identifier of the wait event
EVENT	VARCHAR(128)	The name of the wait event
P1	BIGINT	Parameter 1 of the wait event
P2	BIGINT	Parameter 2 of the wait event
P3	BIGINT	Parameter 3 of the wait event

Column name	Type	Description
WAIT_CLASS_ID	INTEGER	The identifier of the wait class
WAIT_CLASS	VARCHAR(128)	The name of the wait class
WAIT_TIME	BIGINT	The time spent waiting (in milliseconds)
SECOND_IN_WAIT	BIGINT	The time spent waiting (in seconds)

3.2.72.1 Column Information

SID

This is the identifier of a currently connected session.

SEQNUM

This is the identifier of the wait event associated with the session.

EVENT

This is the name of the wait event.

WAIT_CLASS_ID

This is the identifier of the class of the wait event.

WAIT_CLASS

This is the name of the wait class.

WAIT_TIME

This is the amount of time spent waiting for the wait event (in milliseconds).

SECOND_IN_WAIT

This is the amount of time spent waiting for the wait event (in seconds).

3.2.73 V$SESSION_WAIT_CLASS

This view shows cumulative statistical information about waits, classified according to session and wait event, for all currently connected sessions. This view does not provide information about wait events related to sessions that are no longer connected.

Column name	Type	Description
SID	INTEGER	The session identifier

Column name	Type	Description
SERIAL	INTEGER	The identifier of the wait event
WAIT_CLASS_ID	INTEGER	The identifier of the wait class
WAIT_CLASS	VARCHAR(128)	The name of the wait class
TOTAL_WAITS	BIGINT	The total number of waits for this wait event in this session
TIME_WAITED	VACHAR(128)	The total amount of time waited for this wait event in this session (in milliseconds)

3.2.73.1 Column Information

WAIT_CLASS_ID

This is the identifier of the wait class.

WAIT_CLASS

This is the name of the wait class.

TOTAL_WAITS

This is the total number of waits for this wait event in this session.

TIME_WAITED

This is the total time (in milliseconds) spent waiting for this wait event in this session.

SID

This is the identifier of the session.

SERIAL

This is the identifier of the wait event.

3.2.73.2 Example

<Example 1> The following SELECT query outputs the total number of waits and the total amount of time spent waiting for each wait event in each session, classified by session, wait event and wait class.

```
select sid, serial, wait_class_id, sum(total_waits), sum(time_waited)
from v$session_wait_class
group by sid, serial, wait_class_id
order by total_waits desc;
```

3.2.74 V$SESSIONMGR

This view displays statistical information about sessions.

Column	Data Type	Description
TASK_COUNT	INTEGER	The number of connected sessions
BASE_TIME	INTEGER	The current time
IDLE_TIMEOUT_COUNT	INTEGER	See below
QUERY_TIMEOUT_COUNT	INTEGER	See below
FETCH_TIMEOUT_COUNT	INTEGER	See below
UTRANS_TIMEOUT_COUNT	INTEGER	See below
SESSION_TERMINATE_COUNT	INTEGER	See below

3.2.74.1 Column Information

TASK_COUNT

This is the total number of currently connected sessions.

BASE_TIME

This is the current time, expressed in seconds.

IDLE_TIMEOUT_COUNT

This is the number of idle timeouts that have occurred since ALTIBASE HDB was started.

QUERY_TIMEOUT_COUNT

This is the number of query timeouts that have occurred since ALTIBASE HDB was started.

FETCH_TIMEOUT_COUNT

This is the number of fetch timeouts that have occurred since ALTIBASE HDB was started.

UTRANS_TIMEOUT_COUNT

This is the number of UPDATE transaction timeouts that have occurred since ALTIBASE HDB was started.

SESSION_TERMINATE_COUNT

This is the number of sessions that have been forcibly disconnected by the sysdba since ALTIBASE HDB was started.

3.2.75 V$SESSTAT

This view shows statistics for all currently connected sessions.

Column name	Type	Description
SID	INTEGER	The identifier of the session.
SEQNUM	INTEGER	The serial number of each statistic
NAME	VARCHAR(128)	The name of the statistic
VALUE	BIGINT	The value returned for the statistic

For information about each status, please refer to V$STATNAME.

3.2.75.1 Column Information

SID

This is the unique identifier for the session.

SEQNUM

This is a serial number for the statistic.

NAME

This is the name of the statistic.

VALUE

This is the value returned for the statistic, expressed as a 64-bit integer.

3.2.76 V$SQLTEXT

This view displays information about SQL that is currently being executed in the server.

Column	Data Type	Description
SID	INTEGER	The identifier of the session
STMT_ID	INTEGER	The identifier of the statement

Column	Data Type	Description
PIECE	INTEGER	The serial number of the text fragment
TEXT	VARCHAR(64)	A fragment of SQL text

3.2.76.1 Column Information

SID

This is a unique number identifying the session in which the SQL text is being executed.

STMT_ID

This is the serial number of the fragment of the SQL statement being executed in the session.

PIECE

The complete SQL statement that is being executed is divided into 64-byte fragments of text and saved. PIECE is a serial number that identifies each line of text, ascending from 0.

TEXT

This is the actual 64-byte fragment of text constituting part of the SQL statement.

3.2.77 V$SQL_PLAN_CACHE

This view shows the current status of the SQL Plan Cache along with some related statistical information.

Column name	Type	Description
MAX_CACHE_SIZE	BIGINT	The maximum size of the SQL Plan Cache (in bytes)
CURRENT_HOT_LRU_SIZE	BIGINT	The current size of the HOT area of an LRU list
CURRENT_COLD_LRU_SIZE	BIGINT	The current size of the COLD area of an LRU list
CURRENT_CACHE_SIZE	BIGINT	The current size of the SQL Plan Cache
CURRENT_CACHE_OBJ_COUNT	INTEGER	The number of plan objects currently registered in the SQL Plan Cache
CACHE_HIT_COUNT	BIGINT	The number of times that plan cache objects registered in the SQL Plan Cache are used
CACHE_MISS_COUNT	BIGINT	The number of failures to find plan objects in the SQL Plan Cache

Column name	Type	Description
CACHE_IN_FAIL_COUNT	BIGINT	The number of failures to insert new plan objects into the SQL Plan Cache due to the maximum size restriction
CACHE_OUT_COUNT	BIGINT	The number of plan objects that have been deleted from the SQL Plan Cache
CACHE_INSERTED_COUNT	BIGINT	The number of plan objects that have been inserted into the SQL Plan Cache
NONE_CACHE_SQL_TRY_COUNT	BIGINT	The number of attempts to execute statements, such as DDL and DCL statements, that do not affect the plan cache

3.2.77.1 Column Information

MAX_CACHE_SIZE

This is the maximum size of the SQL Plan Cache. To reduce or increase this maximum size, execute '`alter system set SQL_PLAN_CACHE_SIZE = `'.

CURRENT_HOT_LRU_SIZE

The plan cache objects on the SQL Plan Cache LRU list that are frequently referred to are managed in a HOT area, the size of which is expressed in bytes.

CURRENT_COLD_LRU_SIZE

The plan cache objects on the SQL Plan Cache LRU list that are not frequently referred to are managed in a COLD area, the size of which is expressed in bytes.

CURRENT_CACHE_SIZE

This is the total size of plan cache objects that are currently in the SQL Plan Cache.

CURRENT_CACHE_OBJ_COUNT

This is the number of plan cache objects that are in the SQL Plan Cache.

CACHE_HIT_COUNT

This is the total number of times that plan cache objects in the SQL Plan Cache have been used.

CACHE_MISS_COUNT

This is the number of attempts to refer to plan cache objects that do not exist in the SQL Plan Cache.

CACHE_IN_FAIL_COUNT

This is the number of times that a plan cache object could not be inserted into the cache due to the maximum memory size restriction of the cache, even though an attempt was made to delete or remove infrequently consulted plan cache objects from the cache.

CACHE_OUT_COUNT

This is the number of plan cache objects that were deleted from the SQL Plan Cache.

CACHE_INSERTED_COUNT

This is the number of plan cache objects that were added to the SQL Plan Cache.

NONE_CACHE_SQL_TRY_COUNT

This is the number of attempts to execute statements that do not affect the plan cache. These statements are usually DDL or DCL statements.

3.2.78 V$SQL_PLAN_CACHE_PCO

This view displays information about plan cache objects registered in the SQL Plan Cache.

Column name	Type	Description
SQL_TEXT_ID	VARCHAR(64)	The identifier of the SQL text object containing the plan cache object
PCO_ID	INTEGER	The identifier of the plan cache object in the SQL text object
CREATE_REASON	VARCHAR(28)	The reason the plan cache object was created
HIT_COUNT	INTEGER	The number of times the plan cache object has been referred to
REBUILD_COUNT	INTEGER	The number of times the plan cache object has been rebuilt
PLAN_STATE	VARCHAR(17)	The state of the plan of the plan cache object

3.2.78.1 Column Information

SQL_TEXT_ID

This is the identifier of the SQL text object to which the plan cache object belongs.

PCO_ID

This is the identifier of the plan cache object in the SQL text object.

CREATE_REASON

This is the reason for creating the plan cache object. It can have the following values:

- CREATE_BY_CACHE_MISS

  The plan cache object was created because no such object existed in the SQL Plan Cache.

- CREATE_BY_PLAN_INVALIDATION

  A plan cache object was found in the SQL Plan Cache during PREPARE work, but a new object was created because the database object referred to in the plan was not valid.

- CREATE_BY_PLAN_TOO_OLD

  A new plan cache object was created, either because the range of statistical information about objects to which the plan refers has changed excessively, or because a DDL statement was executed.

HIT_COUNT

This is the number of times the plan cache object has been referred to.

REBUILD_COUNT

This is the number of times the plan cache object has been recompiled.

PLAN_STATE

This is the status of the plan of the plan cache object. It can have the following values:

- NOT_READY

  This is the state in which a plan and environment have not yet been assigned to the plan cache object.

- READY

  This is the state in which a plan and environment have been assigned to the plan cache object.

- HARD-PREPARE-NEED

  This is the state in which Hard Prepare (forcible plan creation) is necessary because the statement does not affect the plan cache or because there is insufficient plan cache area.

- OLD_PLAN

  This is the state in which the plan is not valid and will not be used in the future.

3.2.79 V$SQL_PLAN_CACHE_SQLTEXT

This view displays information about plan cache objects registered in the SQL Plan Cache.

Column name	Type	Description
SQL_TEXT_ID	VARCHAR(64)	The identifier of the SQL statement in the SQL Plan Cache
SQL_TEXT	VARCHAR(16384)	The SQL statements
CHILD_PCO_COUNT	INTEGER	The number of Child Plan Cache objects
CHILD_PCO_CREATE_COUNT	INTEGER	The number of Child Plan Cache objects that have been created

3.2.79.1 Column Information

SQL_TEXT_ID

This is the identifier of the SQL statement in the SQL Plan Cache. The first 4 digits indicate thenumber of the bucket in which the SQL statement is stored in the SQL Plan Cache. The remaining digits indicate the serial number of the SQL statement in the bucket.

SQL_TEXT

This is the actual SQL statement.

CHILD_PCO_COUNT

This is the number of Child Plan Cache objects that the SQL Text Plan object currently possesses.

CHILD_PCO_CREATE_COUNT

This is the number of Child Plan Caches that have been created in the SQL Text Plan object so far. New Child Plan Cache objects are created in the SQL Text Plan object in the following two cases:

- A Child Plan Cache object is created when the SQL statement is the same but the environment in which the plan was created has changed.
- A new plan cache object is created when objects that refer to the plan cache object have changed, or when the range of statistical information about objects has changed excessively.

3.2.80 V$STABLE_MEM_DATAFILES

This view shows the complete file path of the data files in the database.

Column name	Type	Description
MEM_DATA_FILE	VARCHAR(256)	The full path of the data file

3.2.80.1 Column Information

MEM_DATA_FILE

This is the full path of the data files in the database.

3.2.81 V$STATEMENT

This view shows information about the most recently executed query in each currently connected session.

Column name	Type	Description
ID	INTEGER	The identifier of the statement
PARENT_ID	INTEGER	The identifier of the parent statement
CURSOR_TYPE	INTEGER	The cursor type
SESSION_ID	INTEGER	The ID of the session to which the statement belongs
TX_ID	BIGINT	The identifier of the transaction
QUERY	VARCHAR(16384)	The first 1024 characters of the SQL string that is or was executed
LAST_QUERY_START_TIME	INTEGER	The start time of the most recent query
QUERY_START_TIME	INTEGER	The start time of the current query
FETCH_START_TIME	INTEGER	The start time of the current fetch
STATE	VARCHAR(13)	The state of the current statement
ARRAY_FLAG	INTEGER	The array execution flag
ROW_NUMBER	INTEGER	The number of the current row
EXECUTE_FLAG	INTEGER	The execution flag
BEGIN_FLAG	BIGINT	A flag that shows whether the current statement is opened or not
TOTAL_TIME	BIGINT	The total elapsed time
PARSE_TIME	BIGINT	The time taken to parse the statement
VALIDATE_TIME	BIGINT	The time taken to validate the statement
OPTIMIZE_TIME	BIGINT	The time taken to optimize the statement
EXECUTE_TIME	BIGINT	The time taken to execute the statement
FETCH_TIME	BIGINT	The time taken to perform a fetch operation

Column name	Type	Description
SOFT_PREPARE_TIME	BIGINT	The time taken to search for a plan in the SQL Plan Cache during the Prepare process
SQL_CACHE_TEXT_ID	VARCHAR(64)	The SQL Text identifier of the SQL plan cache object
SQL_CACHE_PCO_ID	INTEGER	The identifier of the plan cache object
OPTIMIZER	BIGINT	The optimization mode
COST	BIGINT	The optimization cost
USED_MEMORY	BIGINT	Reserved for future use
READ_PAGE	BIGINT	The number of disk pages that have been read
WRITE_PAGE	BIGINT	The number of disk pages that have been written to
GET_PAGE	BIGINT	The number of disk pages that have been accessed
CREATE_PAGE	BIGINT	The number of disk pages that have been created
UNDO_READ_PAGE	BIGINT	The number of disk UNDO pages that have been read
UNDO_WRITE_PAGE	BIGINT	The number of disk UNDO pages that have been written to
UNDO_GET_PAGE	BIGINT	The number of disk UNDO pages that have been accessed
UNDO_CREATE_PAGE	BIGINT	The number of disk UNDO pages that have been created
MEM_CURSOR_FULL_SCAN	BIGINT	The number of memory table searches without indexes
MEM_CURSOR_INDEX_SCAN	BIGINT	The number of memory table searches that use indexes
DISK_CURSOR_FULL_SCAN	BIGINT	The number of disk table searches without indexes
DISK_CURSOR_INDEX_SCAN	BIGINT	The number of disk table searches that use indexes
EXECUTE_SUCCESS	BIGINT	The number of successful statement executions
EXECUTE_FAILURE	BIGINT	The number of failed statement executions
PROCESS_ROW	BIGINT	The number of processed records

Column name	Type	Description
MEMORY_TABLE_ACCESS_COUNT	BIGINT	The number of records that a statement retrieves from the target memory table(s)
SEQNUM	INTEGER	The identifier of a wait event
EVENT	VARCHAR(128)	The name of a wait event
P1	BIGINT	Parameter 1 of the wait event
P2	BIGINT	Parameter 2 of the wait event
P3	BIGINT	Parameter 3 of the wait event
WAIT_TIME	BIGINT	The time spent waiting (in milliseconds)
SECOND_IN_TIME	BIGINT	The time spent waiting (in seconds)

3.2.81.1 Column Information

ID

This is a unique identifier that distinguishes the statement within a session.

PARENT_ID

This is the identifier of the parent statement of the given statement.

CURSOR_TYPE

A hex value of 0x02 indicates a memory cursor, whereas a hex value of 0x04 indicates a disk cursor.

SESSION_ID

This is the identifier of the session to which the statement belongs.

TX_ID

This is the identifier of the transaction that is currently being executed.

QUERY

This is a query string that is currently being executed or was executed by the statement.

LAST_QUERY_START_TIME

Indicates the absolute time when the last query started, in seconds.

QUERY_START_TIME

This is the absolute start time of execution of the currently executed query, in seconds.

FETCH_START_TIME

If the current statement is a SELECT statement, this is the time at which the fetch started, in seconds.

STATE

This is the state of the current statement. It can have the following values:

ALLOC: The statement has been initialized and assigned.

PREPARED: The statement is in a prepared state.

FETCH READY: The statement is being prepared for a fetch operation.

FETCH PROCEED: The statement is in the process of performing a fetch operation.

ARRAY_FLAG

This indicates whether or not the current statement is being executed in array or batch mode. It can have the following values:

0: Not executed in array or batch mode

1: Executed in array or batch mode

ROW_NUMBER

If the current statement is being executed in array or batch mode, this is the number of the row currently being processed, starting at 1.

EXECUTE_FLAG

Indicates whether the current statement is being executed. It can have the following values:

0: Not currently being executed

1: Currently being executed

BEGIN_FLAG

Indicates whether the current statement is open, that is, whether it is being executed.

0: Execution of the current statement has not started, or has completed.

1: The current statement is open.

TOTAL_TIME

This is the total execution time of the current statement, in microseconds.

Depending on the type of the statement, the PVO time or fetch time can be added to EXECUTE_TIME.

PARSE_TIME

This is the time taken to check the syntax of the query, in microseconds.

VALIDATE_TIME

This is the time taken to validate the query, in microseconds.

OPTIMIZE_TIME

This is the time taken to optimize the query, in microseconds.

EXECUTE_TIME

This is the time actually taken to execute a query, in microseconds. In the case of a `SELECT` statement, this is the execution time up until the first fetch occurs.

FETCH_TIME

For a `SELECT` query, this is the time that elapses during fetching, in microseconds.

SOFT_PREPARE_TIME

This is the time taken to find an appropriate plan cache object in the SQL Plan Cache when creating an SQL statement and plan as part of a Prepare task.

SQL_CACHE_TEXT_ID

This is the identifier of the SQL Cache Text object when searching for a plan object in the SQL Plan Cache.

SQL_CACHE_PCO_ID

This is the object identifier of a shared plan cache in the SQL Cache Text object.

OPTIMIZER

This is the optimization mode. It can have the following values:

0: Cost-based optimization

1: Rule-based optimization

COST

This is the cost of optimizing the query.

USED_MEMORY

Reserved for future use.

READ_PAGE

This is the number of disk data pages that are physically read when executing a query.

WRITE_PAGE

This is the number of disk data pages that are physically written to when executing a query.

GET_PAGE

This is the number of disk data pages that are accessed when executing a query.

CREATE_PAGE

This is the number of disk data pages that are created when executing a query.

UNDO_READ_PAGE

This is the number of disk UNDO pages that are physically read when executing a query.

UNDO_WRITE_PAGE

This is the number of disk UNDO pages that are physically written to when executing a query.

UNDO_GET_PAGE

This is the number of disk UNDO pages that are physically accessed when a query is executed.

UNDO_CREATE_PAGE

This is the number of disk UNDO pages that are created when executing a query.

MEM_CURSOR_FULL_SCAN

This is the number of times that a memory table is searched without using an index when executing a query.

MEM_CURSOR_INDEX_SCAN

This is the number of times that a memory table is searched using an index when executing a query.

DISK_CURSOR_FULL_SCAN

This is the number of times that a disk table is searched without using an index when executing a query.

DISK_CURSOR_INDEX_SCAN

This is the number of times that a disk table is searched using an index when executing a query.

EXECUTE_SUCCESS

This is the number of successful query executions.

EXECUTE_FAILURE

This is the number of failed query executions.

PROCESS_ROW

This is the number of records that were processed when a query was executed.

MEMORY_TABLE_ACCESS_COUNT

This is the total number of records that are found in memory tables when a statement is executed. It should be the same as the total number of accesses specified in the execution plan of the statement.

SEQNUM

This is the identifier of the wait event.

EVENT

This is the name of the wait event.

P1

This is a parameter used by the wait event.

P2

This is a parameter used by the wait event.

P3

This is a parameter used by the wait event.

WAIT_TIME

This is the time spent waiting (in milliseconds).

SECOND_IN_TIME

This is the time spent waiting (in seconds).

3.2.82 V$STATNAME

This view shows the numeric identifiers and names of statistics, and is the basis for V$SYSSTAT, which shows the overall statistics for the system, and V$SESSTAT, which shows the statistics for individual sessions.

This table alone does not have any meaning; it should be viewed through one of the above two performance views in order to provide meaningful information.

Column name	Type	Description
SEQNUM	INTEGER	The identifier for the particular statistic
NAME	VARCHAR(128)	The name of the statistic

3.2.82.1 Column Information

SEQNUM

This is the identifier of the statistic, which is shown in one of the above performance views.

NAME

This is the name of the statistic, which is shown in one of the above performance views.

The serial number and a brief description of each statistic are provided in the following table. Each statistic value is expressed as a 64-bit integer in the V$SYSSTAT and V$SESSTAT performance views.

SEQNUM	NAME	Description
0	logon current	The number of users that are currently connected
1	logon cumulative	The cumulative number of users who have connected
2	data page read	The number of times that pages were read in the system or session
3	data page write	The number of times that pages were written to in the system or session
4	data page gets	The number of times that pages were accessed in the system or session using latches
5	data page fix	The number of times that pages were accessed in the system or session without using latches
6	data page create	The number of pages that were created in the system or session
7	undo page read	The number of times that UNDO pages were read in the system or session
8	undo page write	The number of times that UNDO pages were written to in the system or session
9	undo page gets	The number of times that UNDO pages were accessed in the system or session using latches

SEQNUM	NAME	Description
10	undo page fix	The number of times that UNDO pages were accessed in the system or session without using latches
11	undo page create	The number of UNDO pages that were created in the system or session
12	base time in second	The internal time that is maintained by the system (in seconds)
13	query timeout	The number of query timeouts that have occurred in the system or session
14	idle timeout	The number of idle timeouts that have occurred in the system or session
15	fetch timeout	The number of fetch timeouts that have occurred in the system or session
16	utrans timeout	The number of utrans timeouts that have occurred in the system or session
17	session terminated	The number of sessions that have been forcibly shut down in the system
18	statement rebuild count	The number of times that a statement has been rebuilt in the system or session
19	unique violation count	The number of times that a unique key constraint has been violated in the system or session
20	update retry count	The number of times that an update operation has been reattempted in the system or session
21	delete retry count	The number of times that a delete operation has been reattempted in the system or session
22	lock row retry count	The number of times that an attempt to lock a row has been repeated in the system or session
23	session commit	The number of commits that have occurred in the system or session
24	session rollback	The number of rollbacks that have occurred in the system or session
25	fetch success count	The number of successful fetches in the system or session
26	fetch failure count	The number of times a fetch failed in the system or session
27	execute success count	The number of times that queries were successfully executed in the system or session
28	execute failure count	The number of failures to execute a query in the system or session

SEQNUM	NAME	Description
29	prepare success count	The number of times that a Prepare operation was successfully conducted in the system or session
30	prepare failure count	The number of times that a Prepare operation failed in the system or session
31	rebuild count	The number of times a plan cache object was rebuilt in the system or session
32	write redo log count	The number of log records that were recorded in the system or session
33	write redo log bytes	The total number of bytes of logs that were recorded in the system or session
34	read socket count	The number of times that data were read from a socket in the system or session
35	write socket count	The number of times that data were written to a socket in the system or session
36	byte received via inet	The number of bytes of data read using an INET socket in the system or session
37	byte sent via inet	The number of bytes of data written using an INET socket in the system or session
38	byte received via unix domain	The number of bytes of data read using the Unix domain protocol in the system or session
39	byte sent via unix domain	The number of bytes of data written using the Unix domain protocol in the system or session
40	semop count for receiving via ipc	The number of semaphore operations for IPC read tasks in the system or session
41	semop count for sending via ipc	The number of semaphore operations for IPC write tasks in the system or session
42	memory table cursor full scan count	The number of full scan cursors (a full scan cursor is a forward-only cursor that scans an entire table) opened on memory tables using sequential read
43	memory table cursor index scan count	The number of index scan cursors opened on memory tables
44	disk table cursor full scan count	The number of full scan cursors opened on disk tables using sequential read
45	disk table cursor index scan count	The number of index scan cursors opened on disk tables

SEQNUM	NAME	Description
46	lock acquired count	The number of table locks that were obtained in the system or session (Caution: For internal reasons, when viewing V$SYSSTAT, this value may not be the same as the number of locks that have been released. However, for V$SESSTAT, the two values should be the same.)
47	lock released count	The number of table locks that have been released in the system or session
48	service thread created count	The number of service threads that have been created in the system or session
49	memory table access count	The number of times that memory tables have been accessed in the system or session
50	elapsed time: query parse	The total amount of time taken to parse a query. This is a cumulative value.
51	elapsed time: query validate	The total amount of time taken to validate a query. This is a cumulative value.
52	elapsed time: query optimize	The total amount of time taken to optimize a query. This is a cumulative value.
53	elapsed time: query execute	The total amount of time taken to execute a query. This is a cumulative value.
54	elapsed time: query fetch	The total amount of time taken for a query to return records. This is a cumulative value.
55	elapsed time: soft prepare	The total amount of time taken for soft prepare. This is a cumulative value.
56	elapsed time: analyze values in DML(disk)	The total amount of time taken to analyze the input column values when executing DML statements (INSERT or UPDATE) in the system or session. This is a cumulative value.
57	elapsed time: record lock validation in DML(disk)	The amount of time taken to check whether or not records can be updated in the system or session. This is a cumulative value.
58	elapsed time: allocate data slot in DML(slot)	The amount of time taken to allocate data slots during a DML operation in the system or session. This is a cumulative value.
59	elapsed time: write undo record in DML(disk)	The amount of time taken to write undo records in the system or session. This is a cumulative value.
60	elapsed time: allocate tss in DML(disk)	The amount of time taken to allocate transaction slots in the system or session. This is a cumulative value.

SEQNUM	NAME	Description
61	elapsed time: allocate undopage in DML(disk)	The amount of time taken to allocate undo pages in the system or session. This is a cumulative value.
62	elapsed time: index operation in DML(disk)	The amount of time taken to add keys to indexes in the system or session. This is a cumulative value.
63	elapsed time: create page(disk)	The amount of time taken to create pages in the system or session. This is a cumulative value.
64	elapsed time: get page(disk)	The amount of time taken to access pages with latches in the system or session. This is a cumulative value.
65	elapsed time: fix page(disk)	The amount of time taken to access pages without latches in the system or session. This is a cumulative value.
66	elapsed time: logical aging by tx in DML(disk)	Not currently used.
67	elapsed time: physical aging by tx in DML(disk)	Not currently used.
68	elapsed time: replace (plan cache)	The time taken to replace one plan with another plan from a list.
69	elapsed time: victim free in replace (plan cache)	The time taken to release a victim while replacing one plan with another plan from a list.
70	elapsed time: hard rebuild	When a plan is found in the plan cache but is determined to be invalid, this is the amount of time taken to re-build it. This is a cumulative value.
71	elapsed time: soft rebuild	When a plan is found in the plan cache but is determined to be invalid and is thus to be rebuilt, this is the amount of time spent waiting for another transaction to re-build the plan. This is a cumulative value.
72	elapsed time: add hard-prepared plan to plan cache	The amount of time taken to add a plan created by hard prepare (i.e. a forcibly created plan) to the plan cache. This is a cumulative value.
73	elapsed time: add hard-built plan to plan cache	The amount of time taken to add a plan created by hard rebuild (refer to #70) to the plan cache. This is a cumulative value.
74	elapsed time: search time for parent PCO	The amount of time taken to find a parent PCO (Plan Cache Object that has SQL text). This is a cumulative value.

SEQNUM	NAME	Description
75	elapsed time: creation time for parent PCO	The amount of time taken to create a new parent PCO. This is a cumulative value.
76	elapsed time: search time for child PCO	The sum of #82 and #83 (i.e. 82 + 83). This is a cumulative value.
77	elapsed time: creation time for child PCO	The amount of time taken to create a new child PCO (Plan Cache Object which has an execution plan). This is a cumulative value.
78	elapsed time: validation time for child PCO	The amount of time taken to validate a child PCO. This is a cumulative value.
79	elapsed time: creation time for new child PCO by rebuild at execution	The amount of time taken to create a new child PCO in the case where a plan is re-built during the execution phase. This is a cumulative value.
80	elapsed time: creation time for new child PCO by rebuild at soft prepare	The amount of time taken to create a new child PCO in the case where a plan is re-built during the soft prepare phase. This is a cumulative value.
81	elapsed time: hard prepare time	The amount of time taken for hard prepare, that is, to create a plan when no plan exists in the plan cache. This is a cumulative value.
82	elapsed time: matching time for child PCO	The amount of time taken to determine which plan is the desired plan in the case where there are two or more child PCOs that have the same SQL text. This is a cumulative value.
83	elapsed time: waiting time for hard prepare	The sum of #81 and #72 (i.e. 81 + 72). This is a cumulative value.
84	elapsed time: moving time from cold region to hot region	The amount of time taken to move a plan from a cold area to a hot area. This is a cumulative value.
85	elapsed time: waiting time for parent PCO when choosing plan cache replacement target	The amount of time spent waiting for a parent PCO latch to check child PCOs when choosing a replacement target. This is a cumulative value.
86	elapsed time: privilege checking time during soft prepare	The amount of time taken to check privileges for access to objects during soft prepare. This is a cumulative value.

3.2.83 V$SYSSTAT

This view shows the status of the system. It should be noted that the shown value may be out of date, because the status values are updated every 3 seconds based on the data for all sessions.

Column name	Type	Description
SEQNUM	INTEGER	The identifier of the statistical category
NAME	CHAR(128)	The name of the statistic
VALUE	BIGINT	The value of the statistic

For information about each statistic, please refer to V$STATNAME.

Note: The timestamps that can be obtained from Windows NT are limited to a maximum resolution of 10 or 15 milliseconds, depending on the underlying hardware. When the TIMED_STATISTICS property is set to 1 in altibase.properties, statistics which show elapsed time, such as "elapsed time: query parse" and "elapsed time: query validate", in the V$SYSSTAT and V$SESSTAT performance views will be multiples of the above maximum resolution.

3.2.83.1 Column Information

SEQNUM

This is the serial number of the system statistic.

NAME

This is the name corresponding to the statistic serial number.

VALUE

This is the current system value corresponding to the statistic serial number, expressed as a 64-bit integer.

3.2.84 V$SYSTEM_CONFLICT_PAGE

This displays conflict information, classified by page type, for use in analyzing bottlenecks caused by page latch contention in disk buffer space.

This information is collected only if the TIMED_STATISTICS property is set to 1.

Column name	Type	Description
PAGE_TYPE	VARCHAR(20)	The type of page
LATCH_MISS_CNT	BIGINT	The number of failures to acquire latches
LATCH_MISS_TIME	BIGINT	The waiting time

3.2.84.1 Column Information

PAGE_TYPE

This is the type of page.

LATCH_MISS_CNT

This is the number of failures to acquire buffer page latches.

LATCH_MISS_TIME

This is the amount of time (in microseconds) spent waiting for failed attempts to acquire buffer page latches.

3.2.85 V$SYSTEM_EVENT

This view shows cumulative statistical information about waits, classified according to wait event, from the time ALTIBASE HDB was started to the present.

Column name	Type	Description
EVENT	VARCHAR(128)	The name of the wait event
TOTAL_WAITS	BIGINT	The total number of waits for this event
TOTAL_TIMEOUTS	BIGINT	The number of failures to gain access to the requested resource within the specified time
TIME_WAITED	BIGINT	The total time spent waiting for this wait event by all sessions (in milliseconds)
AVERAGE_WAIT	BIGINT	The average length of a wait for this event (in milliseconds)
TIME_WAITED_MICRO	BIGINT	(The total time spent waiting for this wait event by all sessions (in microseconds)
EVENT_ID	INTEGER	The identifier of the wait event
WAIT_CLASS_ID	INTEGER	The identifier of the wait class
WAIT_CLASS	VARCHAR(128)	The name of the wait class

3.2.85.1 Column Information

EVENT

This is the name of the wait event.

TOTAL_WAITS

This is the total number of waits for this event.

TOTAL_TIMEOUTS

This is the number of failures to gain access to the requested resource even after the specified time has elapsed.

TIME_WAITED

This is the total amount of time spent waiting for this wait event by all sessions (in milliseconds).

AVERAGE_WAIT

This is the average time spent waiting for this wait event (in milliseconds).

TIME_WAITED_MICRO

This is the total amount of time spent waiting for this event by all sessions (in microseconds).

EVENT_ID

This is the identifier of the wait event.

WAIT_CLASS_ID

This is the identifier of the wait class into which the event being waited for in the session is categorized.

WAIT_CLASS

This is the name of the wait class into which the event being waited for in the session is categorized.

3.2.86 V$SYSTEM_WAIT_CLASS

This view shows cumulative statistical information about waits, classified according to wait class, from the time ALTIBASE HDB was started to the present.

Column name	Type	Description
WAIT_CLASS_ID	INTEGER	The identifier of the wait class
WAIT_CLASS	VHARCHAR(128)	The name of the wait class
TOTAL_WAITS	BIGINT	The total number of waits in this wait class
TIME_WAITED	VACHAR(128)	The total amount of time spent waiting for this wait class by all processes (in milliseconds)

3.2.86.1 Column Information

WAIT_CLASS_ID

This is the identifier of the wait class.

WAIT_CLASS

This is the name of the wait class.

TOTAL_WAITS

This is the total number of waits for this class.

TIME_WAITED

This is the total time (in milliseconds) spent waiting for this wait class by all sessions.

3.2.86.2 Example

<Example 1> The following query outputs the waiting time and the number of waits in each wait class for all current wait events.

```
iSQL> select * from v$system_wait_class order by total_waits desc;
```

<Example 2> The following query outputs the proportion of waits in each wait class to total waits and the proportion of time spent waiting in each wait class to the total amount of time spent waiting, in descending order, starting with the wait class in which the longest waits have occurred.

```
iSQL> select
 WAIT_CLASS,
 TOTAL_WAITS,
 round(100 * (TOTAL_WAITS / SUM_WAITS),2) PCT_WAITS,
 TIME_WAITED,
 round(100 * (TIME_WAITED / SUM_TIME),2) PCT_TIME
from
(select
WAIT_CLASS,
TOTAL_WAITS,
 TIME_WAITED
 from
 V$SYSTEM_WAIT_CLASS
 where
 WAIT_CLASS != 'Idle'),
 (select
 sum(TOTAL_WAITS) SUM_WAITS,
 sum(TIME_WAITED) SUM_TIME
 from
 V$SYSTEM_WAIT_CLASS
 where
 WAIT_CLASS != 'Idle')
order by 5 desc;
```

3.2.87 V$TABLE

This view shows the list of performance views.

Column	Data Type	Description
NAME	VARCHAR(39)	The name of the view
SLOTSIZE	INTEGER	The record size
COLUMNCOUNT	SMALLINT	The number of columns

3.2.87.1 Column Information

NAME

This is the name of the performance view.

SLOTSIZE

This is the size of one record in the performance view.

COLUMNCOUNT

This is the number of columns in the performance view.

3.2.88 V$TABLESPACES

This view shows information about tablespaces.

Column	Data Type	Description
ID	INTEGER	The tablespace identifier
NAME	VARCHAR(40)	The tablespace name
NEXT_FILE_ID	INTEGER	The identifier of the next data file to be created
TYPE	INTEGER	The type of tablespace
STATE	INTEGER	The status of the tablespace
EXTENT_MANAGEMENT	VARCHAR(20)	The method of managing extents, which is set when the user creates a disk tablespace
SEGMENT_MANAGEMENT	VARCHAR(20)	The type of segment in the tablespace
DATA FILE_COUNT	INTEGER	The number of files in the tablespace
TOTAL_PAGE_COUNT	BIGINT	The total number of pages
EXTENT_PAGE_CNT	INTEGER	The size of an extent (number of pages) in the tablespace

Column	Data Type	Description
ALLOCATED_PAGE_CNT	BIGINT	The initial number of pages in the tablespace
PAGE_SIZE	INTEGER	The size of a page in the tablespace
ATTR_LOG_COMPRESS	INTEGER	Whether to compress logs when executing DML statements on tables in the tablespace

3.2.88.1 Column Information

ID

This is the identifier of the tablespace. The identifiers of user tablespaces start at 5 and increment.

NAME

This is the name of the tablespace, which was defined using the CREATE TABLESPACE statement.

NEXT_FILE_ID

This is an identifier that is assigned to a data file when the data file is added to the tablespace. This value increases by 1 for every individual data file that is added.

TYPE

This value indicates the type of tablespace:

0: MEMORY_SYSTEM_DICTIONARY

1: MEMORY_SYSTEM_DATA

2: MEMORY_USER_DATA

3: DISK_SYSTEM_DATA

4: DISK_USER_DATA

5: DISK_SYSTEM_TEMP

6: DISK_USER_TEMP

7: DISK_SYSTEM_UNDO

8: VOLATILE_USER_DATA

STATE

This value indicates the status of the tablespace.

1: OFFLINE

2: ONLINE

5: Offline tablespace that is being backed up

6: Online tablespace that is being backed up

128: DROPPED

1024: Discarded tablespace

1028: Discarded tablespace that is being backed up

EXTENT_MANAGEMENT

This is the method of managing extents, which is set when a user disk tablespace is created. At present, the BITMAP method is supported.

BITMAP: This indicates whether all EXTENTs of a tablespace are allocated.

SEGMENT_MANAGEMENT

When a segment is created in a tablespace, this indicates which type of segment is to be created.

MANUAL: This indicates that a Free list Management Segment (FMS). is to be created.

AUTO: This indicates that a bitmap-based Tree Management Segment (TMS) is to be created.

DATA FILE_COUNT

This is the number of data files in the tablespace.

TOTAL_PAGE_COUNT

This is the size of the tablespace, expressed as the number of pages. The actual size of the tablespace can be calculated by multiplying this value by the page size (TOTAL_PAGE_COUNT * PAGE_SIZE). This is the actual number of usable pages, and does not include the single file header page for each file.

EXTENT_PAGE_COUNT

This is the size of an extent for this tablespace, expressed as the number of pages. An extent has at least 3 pages.

ALLOCATED_PAGE_COUNT

This is the initial number of pages that were allocated to the tablespace.

PAGE_SIZE

This is the size of each of the pages in the tablespace. It is 8 kB for disk tablespaces and 32 kB for memory tablespaces.

ATTR_LOG_COMPRESS

This indicates whether to perform log compression when executing DML statements on tables in the tablespace.

0: do not compress logs

1: compress logs

3.2.89 V$TRACELOG

This view displays information related to message logging, for use in leaving records related to internal database operation.

Column	Type	Description
MODULE_NAME	VARCHAR(8)	The name of the module
TRCLEVEL	INTEGER	The logging level (1~32)
FLAG	VARCHAR(8)	Whether logging is enabled for this module and level. O: Enable X: Disable
POWLEVEL	BIGINT	Two to the power of the level minus one (2^(TRCLEVEL-1))
DESCRIPTION	VARCHAR(64)	A description of this module and level

3.2.89.1 Column Information

MODULE_NAME

This is the name of an ALTIBASE HDB module. At present, ALTIBASE HDB comprises the SERVER, QP, RP and SM modules, each of which can perform message logging.

TRCLEVEL

This is the message logging level. It has a value between 1 and 32.

FLAG

This displays the setting that determines whether history messages for this module and level are output.

X: Not output

O: Output

SUM: This value indicates that the POWLEVEL column for this record contains the sum of POWLEVELs for which the FLAG is set to ‘O’ in each module.

For information on output settings, please refer to the following description.

POWLEVEL

This is 2 to the power of the TRCLEVEL minus one, that is, 2^(TRCLEVEL-1). The stored procedures addTrcLevel() and delTrcLevel() are provided so that users can easily set the logging level. These stored procedures can be created by executing tracelog.sql, which comes with the package.

DESCRIPTION

This is an explanation of the corresponding module and level.

Example

To check the trace logging level currently set for the server module:

```
iSQL> select module_name, trclevel, flag, powlevel, description from
v$tracelog where module_name like '%SER%';
MODULE_NAME TRCLEVEL FLAG POWLEVEL DESCRIPTION
------------------------------------------------
SERVER 1 O 1 [DEFAULT] TimeOut(Query,Fetch,Idle,UTrans) Trace Log
SERVER 2 O 2 [DEFAULT] Network Operation Fail Trace Log
SERVER 3 O 4 [DEFAULT] Memory Operation Warning Trace Log
SERVER 4 X 8 ---
SERVER 5 X 16 ---
SERVER 6 X 32 ---
SERVER 7 X 64 ---
SERVER 8 X 128 ---
SERVER 9 X 256 ---
SERVER 10 X 512 ---
SERVER 11 X 1024 ---
SERVER 12 X 2048 ---
SERVER 13 X 4096 ---
SERVER 14 X 8192 ---
SERVER 15 X 16384 ---
SERVER 16 X 32768 ---
SERVER 17 X 65536 ---
SERVER 18 X 131072 ---
SERVER 19 X 262144 ---
SERVER 20 X 524288 ---
SERVER 21 X 1048576 ---
SERVER 22 X 2097152 ---
SERVER 23 X 4194304 ---
SERVER 24 X 8388608 ---
SERVER 25 X 16777216 ---
SERVER 26 X 33554432 ---
SERVER 27 X 67108864 ---
SERVER 28 X 134217728 ---
SERVER 29 X 268435456 ---
SERVER 30 X 536870912 ---
SERVER 31 X 1073741824 ---
SERVER 32 X 2147483648 ---
SERVER 99 SUM 7 Total Sum of Trace Log Values
33 rows selected.
```

Usage

ALTIBASE HDB provides message logging properties for the SERVER, SM, QP and RP modules.

- SERVER_MSGLOG_FLAG: Communication and server messages
- SM _MSGLOG_FLAG: Storage manager-related messages

- QP_MSGLOG_FLAG: Query processor-related messages
- RP_MSGLOG_FLAG: Replication-related messages

Each property has 32 bits. The message type and description for each bit can be checked by viewing V$TRACELOG.

The message logging details can be changed as follows.

- To disable the output of all server logging messages:

  ```
  alter system set server_msglog_flag=0
  ```

- To enable the output of server logging messages related to the 1st, 2nd and 5th bits (1+2+5):

  ```
  alter system set server_msglog_flag=8
  ```

- To disable the output of all replication logging messages except conflict-related messages:

  ```
  alter system set rp_msglog_flag=2
  ```

- To enable stored procedure error line logging (the 1st bit) and details pertaining to the execution of DDL statements (the 2nd bit) for the query processor (1+2):

  ```
  alter system set qp_msglog_flag=3
  ```

3.2.90 V$TRANSACTION

This view displays information about transaction objects.

Column	Data Type	Description
ID	BIGINT	The transaction identifier
SESSION_ID	INTEGER	See below
MEMORY_VIEW_SCN	VARCHAR(29)	See below
MIN_MEMORY_LOB_VIEW_SCN	VARCHAR(29)	See below
DISK_VIEW_SCN	VARCHAR(29)	See below
MIN_DISK_LOB_VIEW_SCN	VARCHAR(29)	See below
COMMIT_SCN	VARCHAR(29)	See below
STATUS	BIGINT	See below
UPDATE_STATUS	BIGINT	See below
LOG_TYPE	INTEGER	See below
XA_COMMIT_STATUS	BIGINT	See below
XA_PREPARED_TIME	VARCHAR(64)	See below
FIRST_UNDO_NEXT_LSN_LFGID	INTEGER	See below

Column	Data Type	Description
FIRST_UNDO_NEXT_LSN_FILENO	INTEGER	See below
FIRST_UNDO_NEXT_LSN_OFFSET	INTEGER	See below
CURRENT_UNDO_NEXT_SN	BIGINT	For internal use
CURRENT_UNDO_NEXT_LSN_LFGID	INTEGER	For internal use
CURRENT_UNDO_NEXT_LSN_FILENO	INTEGER	For internal use
CURRENT_UNDO_NEXT_LSN_OFFSET	INTEGER	For internal use
LAST_UNDO_NEXT_LSN_LFGID	INTEGER	See below
LAST_UNDO_NEXT_LSN_FILENO	INTEGER	See below
LAST_UNDO_NEXT_LSN_OFFSET	INTEGER	See below
LAST_UNDO_NEXT_SN	BIGINT	See below
SLOT_NO	INTEGER	See below
UPDATE_SIZE	BIGINT	See below
ENABLE_ROLLBACK	BIGINT	For internal use
FIRST_UPDATE_TIME	INTEGER	See below
LOG_BUF_SIZE	INTEGER	For internal use
LOG_OFFSET	INTEGER	For internal use
SKIP_CHECK_FLAG	BIGINT	For internal use
SKIP_CHECK_SCN_FLAG	BIGINT	For internal use
DDL_FLAG	BIGINT	See below
TSS_RID	BIGINT	See below
UNDO_NO	INTEGER	See below
RESOURCE_GROUP_ID	INTEGER	The log file group identifier

3.2.90.1 Column Information

ID

This is a number for classifying the transaction, ranging from 0 to $2^{32} - 1$. These values can be reused.

SESSION_ID

This is the identifier of the session in which the transaction is executing. If no session is associated with the transaction, this value is -1, which indicates that the transaction branch is in a prepared state in an XA environment.

MEMORY_VIEW_SCN

Because ALTIBASE HDB uses MVCC, it has an SCN that indicates the relative point in time at which each cursor for a table was opened. This value is the smallest value of the View SCNs for memory table cursors for the transaction. A value of 2^{63} means that no cursor is open.

MIN_MEMORY_LOB_VIEW_SCN

This is the SCN of the oldest of the currently open memory LOB cursors for the present transaction. A value of 2^{63} means that no cursors are open.

DISK_VIEW_SCN

This is the lowest of the View SCN values for cursors that are currently open for disk tables for the present transaction. The range of values is the same as for MEMORY_VIEW_SCN.

MIN_DISK_LOB_VIEW_SCN

This is the SCN of the oldest of the currently open disk LOB cursors for the present transaction. A value of 2^{63} means that no cursors are open.

COMMIT_SCN

This is the system SCN at the point in time at which the transaction is committed. A value of 2^{63} means that the transaction has not been committed yet.

STATUS

This is the status of the current transaction. The possible values are:

0: BEGIN

1: PRECOMMIT

2: COMMIT_IN_MEMORY

3: COMMIT

4: ABORT

5: BLOCKED

6: END

UPDATE_STATUS

This indicates whether the transaction is a transaction that is still updating or a read-only transaction.

0: Read-only

1: Updating

LOG_TYPE

This indicates whether the transaction updates tables related to replication. The possible values are:

0: General

1: Replication-related

XA_COMMIT_STATUS

This is the status of a local transaction that is caused by a global transaction. It can have the following values:

0: BEGIN

1: PREPARED

2: COMPLETE

XA_PREPARED_TIME

This is the point in time at which a PREPARE command was received from the global transaction manager as the result of a global transaction.

FIRST_UNDO_NEXT_LSN_LFGID

This is the log file group identifier portion of the LSN, which indicates the location of the first log recorded for the transaction.

FIRST_UNDO_NEXT_LSN_FILENO

This is the file number portion of the LSN, which indicates the location of the first log recorded for the transaction.

FIRST_UNDO_NEXT_LSN_OFFSET

This is the offset portion of the LSN, which indicates the location of the first log recorded for the transaction. The offset indicates the location of the log within a file.

LAST_UNDO_NEXT_LSN_LFGID

This is the log file group identifier portion of the LSN, which indicates the location of the last log recorded for the transaction.

LAST_UNDO_NEXT_LSN_FILENO

This is the file number portion of the LSN, which indicates the location of the last log recorded for the transaction.

LAST_UNDO_NEXT_LSN_OFFSET

This is the offset portion of the LSN, which indicates the location of the last log recorded for the transaction. The offset indicates the location of the log within a file.

LAST_UNDO_NEXT_SN

This is the sequence number (SN) of the last log recorded for the transaction.

SLOT_NO

This is the location of the transaction object in the transaction pool.

UPDATE_SIZE

This is the size of the data created as the result of an UPDATE operation executed by the transaction. If this value is greater than the value of the LOCK_ESCALATION_MEMORY_SIZE property, the table is locked with an X-lock and updates are performed according to the in-place update method.

FIRST_UPDATE_TIME

This is the point in time at which the database was first updated.

DDL_FLAG

This indicates whether the transaction is one that executes a DLL statement:

0: non-DDL

1: DDL

TSS_RID

This is the physical location of the Transaction Status Slot (TSL), which is obtained in order to perform an UPDATE operation on a disk table. A nonzero value means that the transaction has executed at least one update operation on a disk table.

3.2.91 V$TRANSACTION_MGR

This value displays information about the Transaction Manager of ALTIBASE HDB.

Column	Data Type	Description
TOTAL_COUNT	INTEGER	The total number of transactions
FREE_LIST_COUNT	INTEGER	The number of free lists
BEGIN_ENABLE	BIGINT	Indicates whether a new transaction can be commenced
ACTIVE_COUNT	INTEGER	The number of active transactions
SYS_MIN_DISK_VIEWSCN	VARCHAR(29)	The lowest transaction disk view SCN

3.2.91.1 Column Information

TOTAL_COUNT

When ALTIBASE HDB is started, it creates a number of transaction objects equal to the number defined in this property, and uses these objects as the transaction pool. TOTAL_COUNT is the total number of transactions that have been created.

FREE_LIST_COUNT

This is the number of lists used to separately manage the transaction pool.

BEGIN_ENABLE

This indicates whether a new transaction can begin.

0: Disabled

1: Enabled

ACTIVE_COUNT

This is the number of transaction objects that have been assigned to tasks and are currently executing them.

SYS_MIN_DISK_VIEWSCN

This is the lowest transaction disk view SCN (System Change Number).

3.2.92 V$TSSEGS

This view outputs a list of all TSS segments that exist in UNDO tablespace.

Column name	Type	Description
SPACE_ID	INTEGER	The identifier of the UNDO tablespace
SEG_PID	INTEGER	The identifier of the TSS segment page
TXSEG_ENTRY_ID	INTEGER	The identifier of the transaction segment
CUR_ALLOC_EXTENT_RID	BIGINT	The RID of the extent currently being used in the TSS segment
CUR_ALLOC_PAGE_ID	INTEGER	The identifier of the page currently being used in the TSS segment
TOTAL_EXTENT_COUNT	BIGINT	The total number of extents in the TSS segment
TOTAL_EXTDIR_COUNT	BIGINT	The total number of extent directories in the TSS segment

Column name	Type	Description
PAGE_COUNT_IN_EXTENT	INTEGER	The total number of pages in one extent

3.2.92.1 Column Information

SPACE_ID

This is the identifier of the UNDO tablespace.

SEG_PID

This is the identifier of the TSS segment page.

TXSEG_ENTRY_ID

This is the identifier of the transaction segment.

CUR_ALLOC_EXTENT_RID

This is the RID (resource identifier) of the extent currently being used in the TSS segment.

CUR_ALLOC_PAGE_ID

This is the identifier of the page currently being used in the TSS segment.

TOTAL_EXTENT_COUNT

This is the total number of extents in the TSS segment.

TOTAL_EXTDIR_COUNT

This is the total number of extent directories in the TSS segment.

PAGE_COUNT_IN_EXTENT

This is the total number of pages in one extent.

3.2.93 V$TXSEGS

This view outputs the list of transaction segments that are bound to transactions, and thus online (active).

Column name	Type	Description
ID	INTEGER	The identifier of the transaction segment

Column name	Type	Description
TRANS_ID	INTEGER	The identifier of the transaction to which the segment is bound
MIN_DISK_VIEW_SCN	VARCHAR(29)	The lowest disk view SCN of the transaction
COMMIT_SCN	VARCHAR(29)	The commit SCN of the transaction
FIRST_DISK_VIEW_SCN	VARCHAR(29)	The first disk view SCN of the transaction
TSS_RID	BIGINT	The RID of the TSS for the transaction
TSSEG_EXTENT_RID	BIGINT	The RID of the extent of the TSS segment allocated to the TSS
FST_UDSEG_EXTENT_RID	BIGINT	The RID of the first extent of the UNDO segment used by the transaction
LST_UDSEG_EXTENT_RID	BIGINT	The RID of the last extent of the UNDO segment used by the transaction
FST_UNDO_PAGEID	INTEGER	The identifier of the page containing the first UNDO record written by the transaction
FST_UNDO_SLOTNUM	SMALLINT	The slot number of the first UNDO record written by the transaction
LST_UNDO_PAGEID	INTEGER	The identifier of the page containing the last UNDO record written by the transaction
LST_UNDO_SLOTNUM	SMALLINT	The slot number of the last UNDO record written by the transaction

3.2.93.1 Column Information

ID

This is the identifier of the transaction segment.

TRANS_ID

This is the identifier of the transaction to which the segment is bound.

MIN_DISK_VIEW_SCN

This is the lowest disk view SCN for the transaction.

COMMIT_SCN

This is the commit SCN for the transaction.

FIRST_DISK_VIEW_SCN

This is the first disk view SCN for the transaction.

TSS_RID

This is the RID (resource identifier) of the TSS (Transaction Status Slot) allocated to the transaction.

TSSEG_EXTENT_RID

This is the RID (resource identifier) of the extent of the TSS segment allocated to the TSS.

FST_UDSEG_EXTENT_RID

This is the RID (resource identifier) of the first extent of the UNDO segment used by the transaction.

LST_UDSEG_EXTENT_RID

This is the RID (resource identifier) of the last extent of the UNDO segment used by the transaction.

FST_UNDO_PAGEID

This is the identifier of the page containing the first UNDO record written when the transaction is updated.

FST_UNDO_SLOTNUM

This is the slot number in the page containing the first UNDO record written when the transaction is updated.

LST_UNDO_PAGEID

This is the identifier of the page containing the last UNDO record written when the transaction is updated.

LST_UNDO_SLOTNUM

This is the slot number in the page containing the last UNDO record written when the transaction is updated.

3.2.94 V$UDSEGS

This view outputs a list of all UNDO segments existing in undo tablespace.

Column name	Type	Description
SPACE_ID	INTEGER	The UNDO tablespace identifier
SEG_PID	INTEGER	The UNDO segment page identifier
TXSEG_ENTRY_ID	INTEGER	The transaction segment identifier
CUR_ALLOC_EXTENT_RID	BIGINT	The RID of the extent currently used in the UNDO segment

Column name	Type	Description
CUR_ALLOC_PAGE_ID	INTEGER	The identifier of the page currently used in the UNDO segment
TOTAL_EXTENT_COUNT	BIGINT	The total number of extents in the UNDO segment
TOTAL_EXTDIR_COUNT	BIGINT	The total number of extent directories in the UNDO segment
PAGE_COUNT_IN_EXTENT	INTEGER	The total number of pages in one extent

3.2.94.1 Column Information

SPACE_ID

This is the identifier of the UNDO tablespace.

SEG_PID

This is the identifier of the page associated with the UNDO segment.

TXSEG_ENTRY_ID

This is the identifier of the segment used by the transaction.

CUR_ALLOC_EXTENT_RID

This is the RID of the extent that is currently being used in the UNDO segment.

CUR_ALLOC_PAGE_ID

This is the identifier of the page that is currently being used in the UNDO segment.

TOTAL_EXTENT_COUNT

This is the total number of extents in the UNDO segment.

TOTAL_EXTDIR_COUNT

This is the total number of extent directories in the UNDO segment.

PAGE_COUNT_IN_EXTENT

This is the total number of pages in one extent.

3.2.95 V$UNDO_BUFF_STAT

This view displays buffer pool statistics related to the UNDO tablespace.

Column	Data Type	Description
READ_PAGE_COUNT	BIGINT	See below
GET_PAGE_COUNT	BIGINT	The number of page requests made to the buffer manager
FIX_PAGE_COUNT	BIGINT	The number of UNDO page requests made to the buffer manager
CREATE_PAGE_COUNT	BIGINT	See below
HIT_RATIO	DOUBLE	The hit ratio of the buffer frame

3.2.95.1 Column Information

READ_PAGE_COUNT

The total number of pages read from disk since the buffer was initialized.

GET_PAGE_COUNT

This is the total number of page requests made to the buffer manager since the buffer was initialized. If the page is in the buffer, the buffer manager returns the requested page, otherwise the page is read from disk and then returned.

FIX_PAGE_COUNT

This is the total number of UNDO page requests made without latches to the buffer manager since the buffer was initialized.

CREATE_PAGE_COUNT

This is the total number of page creation requests made by transactions to the buffer manager since the buffer was initialized. The buffer manager responds to such requests by obtaining a free BCB from the buffer and then creating and returning a page. This operation does not incur any disk I/O.

3.2.96 V$VERSION

This view displays information about the version of the database.

Column	Data Type	Description
PRODUCT_VERSION	VARCHAR(128)	The product version, e.g. 6.1.1.1
PKG_BUILD_PLATFORM_INFO	VARCHAR(128)	The platform on which the package was built
PRODUCT_TIME	VARCHAR(128)	The date on which the package was built

Column	Data Type	Description
SM_VERSION	VARCHAR(128)	The version of the Storage Manager
META_VERSION	VARCHAR(128)	The meta table version
PROTOCOL_VERSION	VARCHAR(128)	The communication protocol version
REPL_PROTOCOL_VERSION	VARCHAR(128)	The replication protocol version

3.2.96.1 Column Information

PRODUCT_VERSION

This is the version of the ALTIBASE HDB.

PKG_BUILD_PLATFORM_INFO

This is information about the platform on which the package was built.

PRODUCT_TIME

This is the date and time when the current package was built on the platform.

SM_VERSION

This is the version of the Storage Manager. This version information changes every time the storage structure changes.

META_VERSION

This is the version of the meta tables, in which database information is managed.

PROTOCOL_VERSION

This is the version of the protocols used for database communication.

REPL_PROTOCOL_VERSION

This is the version of the protocol used for replication.

3.2.97 V$VOL_TABLESPACES

This view shows information about volatile tablespaces, which exist in memory.

Column name	Type	Description
SPACE_ID	INTEGER	The identifier of the tablespace

Column name	Type	Description
SPACE_NAME	VARCHAR(512)	The name of the tablespace
SPACE_STATUS	INTEGER	The status of the tablespace
INIT_SIZE	BIGINT	The initial size of the tablespace (in bytes)
AUTOEXTEND_MODE	INTEGER	The auto extension mode of the tablespace
AUTOEXTEND_NEXT_SIZE	BIGINT	The auto extension increment size (in bytes)
MAXSIZE	BIGINT	The maximum size of the tablespace (in bytes)
CURRENT_SIZE	BIGINT	The current size of the tablespace (in bytes)

3.2.97.1 Column Information

SPACE_STATUS

This is a value indicating the status of the tablespace. Please refer to V$MEM_TABLESPACE_STATUS_DESC for details.

AUTOEXTEND_MODE

This indicates the Autoextend mode. If it is set to 1, Autoextend is enabled; if not, Autoextend is disabled.

AUTOEXTEND_NEXTSIZE

This is the incremental size used for auto extension (in bytes).

MAXSIZE

This is the maximum size of the tablespace (in bytes).

CURRENT_SIZE

This is the current size of the tablespace (in bytes).

3.2.98 V$WAIT_CLASS_NAME

This view shows information for classifying ALTIBASE HDB server wait events. This performance view can be used to check wait classes, which are a higher concept for classifying the various kinds of wait events.

Column name	Type	Description
WAIT_CLASS_ID	INTEGER	The identifier of the wait class

Column name	Type	Description
WAIT_CLASS	VARCHAR(128)	The name of the wait class

3.2.98.1 Column Information

WAIT_CLASS_ID

This is the class identifier of the wait event.

WAIT_CLASS

This is the wait class, which is a higher concept for classifying and grouping wait events. In ALTIBASE HDB, wait events are classified into the following 8 wait event classes:

WAIT_CLASS_ID	WAIT_CLASS	Description
0	Other	This wait class includes all wait events not included in any of the following classes.
1	Administrative	This class includes wait events that make the user wait due to the execution of a command with SYSDBA privileges.
2	Configuration	This class includes wait events pertaining to unsuitable settings for database resources.
3	Concurrency	This class includes wait events pertaining to internal database resources.
4	Commit	This class includes wait events pertaining to the synchronization of REDO logs in log files
5	Idle	This class includes wait events pertaining to requested tasks in sessions.
6	User I/O	This class includes wait events pertaining to user I/O.
7	System I/O	This class includes wait events pertaining to system I/O.

3.2.99 V$XID

This view displays a list of XIDs, which are identifiers for distributed transactions in the DBMS. In compliance with XA, the distributed transaction identifier is generated internally by the TM (Transaction Manager) and sent to the RM (Resource Manager), that is, to other database nodes, when a distributed transaction commences.

Column	Data Type	Description
XID_VALUE	VARCHAR(256)	This returns the XID value as a character string
ASSOC_SESSION_ID	INTEGER	The identifier of the session connected to the XID object
TRANS_ID	INTEGER	The identifier of the distributed transaction within the XID object
STATE	VARCHAR(24)	The state of the XID object
STATE_START_TIME	INTEGER	The time at which the state of the XID object was determined
STATE_DURATION	BIGINT	The amount of time that has elapsed since the state of the XID was determined
TX_BEGIN_FLAG	VARCHAR(9)	A flag within the XID object indicating whether the transaction has begun
REF_COUNT	INTEGER	The number of current references to the XID object

3.2.99.1 Column Information

XID_VALUE

This is the XID value, expressed as a character string.

ASSOC_SESSION_ID

This is the identifier of the session related to the XID object, that is, the session which executed XA_START for this XID.

TRANS_ID

This is the internal identifier of the distributed transaction within the XID object.

STATE

This is the state of execution of the XID object. The possible values for this state are as follows:

- IDLE: This means that no sessions are connected to the XID.
- ACTIVE: This means that there is a session connected to the XID. In other words, XA_START has been executed for this XID.
- PREPARED: This means that a Prepare command has been received for a 2PC (Phase Commit) task.
- HEURISTICALLY_COMMITED: This means that the DBMS has forcefully committed the transaction branch of the XID.

- HEURISTICALLY_ROLLBACKED: This means that the DBMS has forcefully rolled back the transaction branch of the XID.
- NO_TX: This means that the XID has just been initialized, or that the transaction branch related to the XID has been committed or rolled back.

STATE_START_TIME

This is the time at which the state of the XID object was determined.

STATE_DURATION

This is the amount of time that has elapsed since the state of the XID object was determined.

TX_BEGIN_FLAG

This is an internal flag within the XID object that indicates whether the transaction branch has been started in the RM.

- BEGIN: The transaction has started
- NOT BEGIN: The transaction has not started

REF_COUNT

This is the number of current references to the XID object.

4 The Sample Schema

This appendix provides information about the schemas and data used in the examples in the ALTIBASE HDB Manuals.

4.1 Information about the Sample Schema

4.1.1 Script Files

A schema creation file is provided at $ALTIBASE_HOME/sample/APRE/schema/schema.sql.

Executing this file creates the tables referenced in the manuals and populates them with sample data.

Therefore, if you would like to work with the examples described in the manuals, first execute the schema creation file, after which it will be possible to follow the provided examples.

4.1.2 The Sample Schema

Purpose: Managing Customers and Orders

Tables: *employees*, *departments*, *customers*, *orders*, *goods*

4.1.2.1 *employees* Table

Primary Key: Employee Number (eno)

Column Name	Data Type	Description	Other
eno	INTEGER	Employee Number	Primary Key
e_lastname	CHAR(20)	Employee Last Name	NOT NULL
e_firstname	CHAR(20)	Employee First Name	NOT NULL
emp_job	VARCHAR(15)	Title	NULL allowed
emp_tel	CHAR(15)	Telephone Number	NULL allowed
dno	SMALLINT	Department Number	NULL allowed, INDEX ASC
salary	NUMBER(10,2)	Monthly Salary	NULL allowed, DEFAULT 0
sex	CHAR(1)	Gender	NULL allowed
birth	CHAR(6)	Birthday	NULL allowed
join_date	DATE	Hiring Date	NULL allowed
status	CHAR(1)	Position	NULL allowed, DEFAULT 'H'

4.1.2.2 *departments* Table

Primary Key: Department Number (dno)

Column Name	Data Type	Description	Other
dno	SMALLINT	Department Number	Primary Key
dname	CHAR(30)	Department Name	NOT NULL
dep_location	CHAR(15)	Department Location	NULL allowed
mgr_no	INTEGER	Administrator Number	NULL allowed, INDEX ASC

4.1.2.3 *customers* Table

Primary Key: Resident Registration Number (cno)

Column Name	Data Type	Description	Other
cno	BIGINT	Customer Number	Primary Key
c_lastname	CHAR(20)	Customer Last Name	NOT NULL
c_firstname	CHAR(20)	Customer First Name	NOT NULL
cus_job	VARCHAR(20)	Occupation	NULL allowed
cus_tel	CHAR(15)	Telephone Number	NOT NULL
sex	CHAR(1)	Gender	NULL allowed
birth	CHAR(6)	Birthday	NULL allowed
postal_cd	VARCHAR(9)	Postal Code	NULL allowed
address	VARCHAR(60)	Address	NULL allowed

4.1.2.4 *orders* Table

Primary Keys: Order Number & Order Date (ono, order_date)

Column Name	Data Type	Description	Other
ono	BIGINT	Order Number	Primary Key
order_date	DATE	Order Date	Primary Key
eno	INTEGER	Sales Clerk	NOT NULL,INDEX ASC
cno	BIGINT	Customer Number	NOT NULL,INDEX DESC
gno	CHAR(10)	Product No.	NOT NULL,INDEX ASC
qty	INTEGER	Order Quantity	NULL allowed, DEFAULT 1

Column Name	Data Type	Description	Other
arrival_date	DATE	Expected Arrival Date	NULL allowed
processing	CHAR(1)	Order Status	NULL allowed (typical values: O: ordered, P: being prepared, D: being delivered, C: complete)

4.1.2.5 *goods* Table

Primary Key: Product No. (gno)

Column Name	Data Type	Description	Other
gno	CHAR(10)	Product Number	Primary Key
gname	CHAR(20)	Product Name	NOT NULL, Unique
goods_location	CHAR(9)	Storage Location	NULL allowed
stock	INTEGER	Stored Quantity	NULL allowed, DEFAULT 0
price	NUMERIC(10,2)	Item Price	NULL allowed

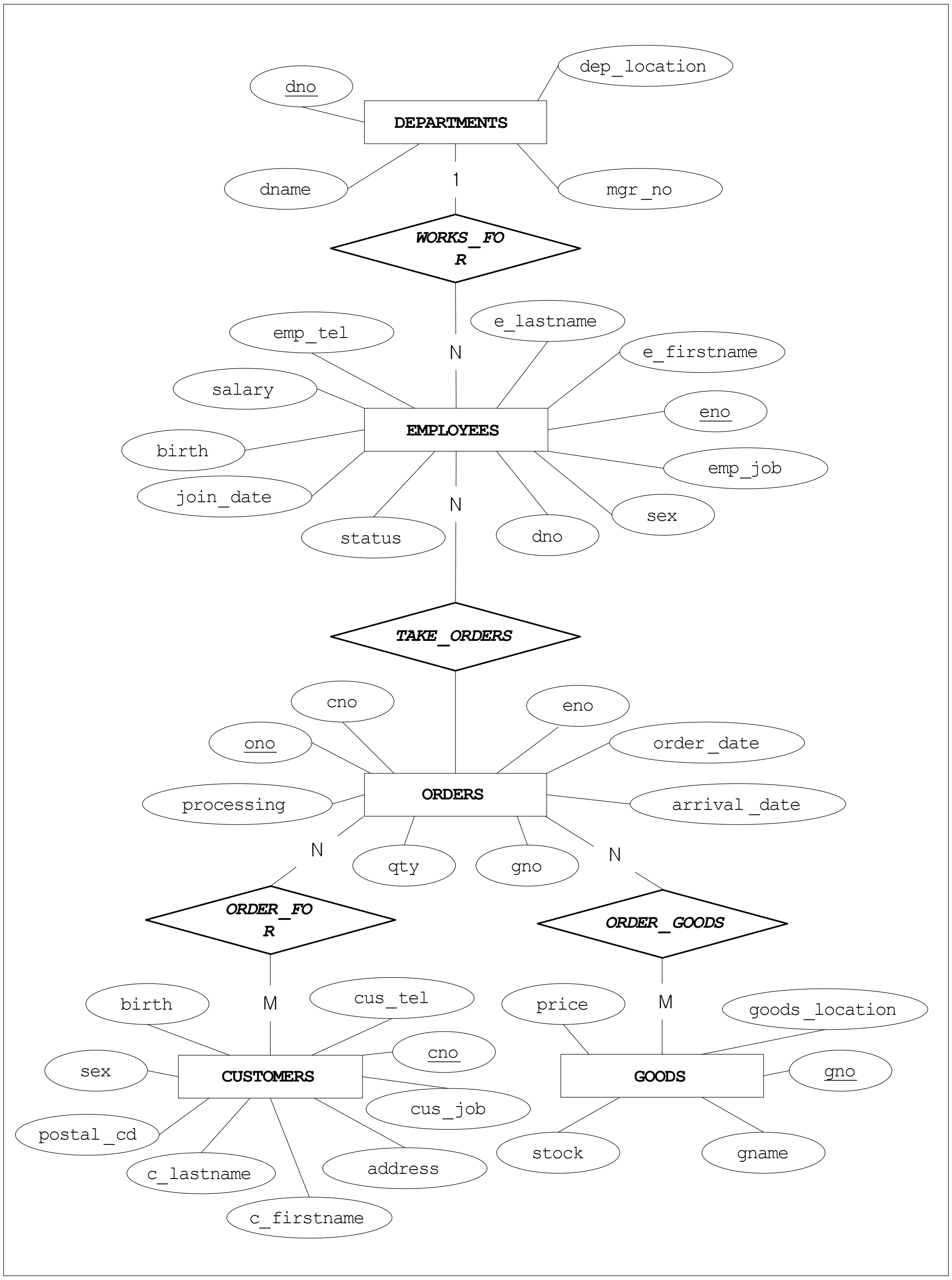
4.1.2.6 *dual* Table

Record Size: 1

Column Name	Data Type	Description	Other
DUMMY	CHAR(1)		DEFAULT ‘X’

4.2 Entity-Relationship (ER) Diagram and Sample Data

4.2.1 ER Diagram



4.2.2 Sample Data

employees Table

```
iSQL> select * from employees;
ENO         E_LASTNAME             E_FIRSTNAME           EMP_JOB
----------------------------------------------------------------------------
EMP_TEL          DNO         SALARY      SEX  BIRTH   JOIN_DATE    STATUS
----------------------------------------------------------------------------
1           Moon                   Chan-seung            CEO
01195662365      3002                    M                        R
2           Davenport              Susan                 designer
0113654540              1500        F  721219  18-NOV-2009  H
3           Kobain                 Ken                   engineer
0162581369       1001        2000        M  650226  11-JAN-2010  H
4           Foster                 Aaron                 PL
0182563984       3001        1800        M  820730               H
5           Ghorbani               Farhad                PL
01145582310      3002        2500        M          20-DEC-2009  H
6           Momoi                  Ryu                   programmer
0197853222       1002        1700        M  790822  09-SEP-2010  H
7           Fleischer              Gottlieb              manager
0175221002       4002        500         M  840417  24-JAN-2004  H
8           Wang                   Xiong                 manager
0178829663       4001                    M  810726  29-NOV-2009  H
9           Diaz                   Curtis                planner
0165293668       4001        1200        M  660102  14-JUN-2010  H
10          Bae                    Elizabeth             programmer
0167452000       1003        4000        F  710213  05-JAN-2010  H
11          Liu                    Zhen                  webmaster
0114553206       1003        2750        M          28-APR-2011  H
12          Hammond                Sandra                sales rep
0174562330       4002        1890        F  810211  14-DEC-2009  H
13          Jones                  Mitch                 PM
0187636550       1002        980         M  801102               H
14          Miura                  Yuu                   PM
0197664120       1003        2003        M                       H
15          Davenport              Jason                 webmaster
0119556884       1003        1000        M  901212               H
16          Chen                   Wei-Wei               manager
0195562100       1001        2300        F  780509               H
17          Fubuki                 Takahiro              PM
0165293886       2001        1400        M  781026  07-MAY-2010  H
18          Huxley                 John                  planner
01755231044      4001        1900        M          30-OCT-2007  H
19          Marquez                Alvar                 sales rep
0185698550       4002        1800        M          18-NOV-2010  H
20          Blake                  William               sales rep
01154112366      4002                    M          18-NOV-2006  H
20 rows selected.
```

departments Table

```
iSQL> select * from departments;
DNO         DNAME                         DEP_LOCATION     MGR_NO
----------------------------------------------------------------------------
1001        RESEARCH DEVELOPMENT DEPT 1     New York         16
1002        RESEARCH DEVELOPMENT DEPT 2     Sydney           13
1003        SOLUTION DEVELOPMENT DEPT       Osaka            14
2001        QUALITY ASSURANCE DEPT          Seoul            17
3001        CUSTOMERS SUPPORT DEPT          London           4
```

```
3002        PRESALES DEPT                  Peking          5
4001        MARKETING DEPT                 Brasilia        8
4002        BUSINESS DEPT                  Palo Alto       7
8 rows selected.
```

customers Table

```
iSQL> select * from customers;
CNO                  C_LASTNAME            C_FIRSTNAME
----------------------------------------------------------------------
CUS_JOB               CUS_TEL          SEX  BIRTH   POSTAL_CD
----------------------------------------------------------------------
ADDRESS
-----------------------------------------------------------------
1                    Sanchez               Estevan
engineer              0514685282       M  720828  90021
2100 Exposition Boulevard Los Angeles USA
2                    Martin                Pierre
doctor                023242121        M  821215  V6T 1F2
4712 West 10th Avenue Vancouver BC Canada
3                    Morris                Gabriel
designer              023442542        M  811111  75010
D914 Puteaux Ile-de-France France
4                    Park                  Soo-jung
engineer              022326393        F  840305  609-735
Geumjeong-Gu Busan South Korea
5                    Stone                 James
webmaster             0233452141       M  821012  6060
142 Francis Street Western Australia AUS
6                    Dureault              Phil
WEBPD                 025743215        M  810209  H1R-2W1
1000 Rue Rachel Est Montreal Canada
7                    Lalani                Yasmin
planner               023143366        F  821225  156772
176 Robinson Road Singapore
8                    Kanazawa              Tsubasa
PD                    024721114        M  730801  141-0031
2-4-6 Nishi-Gotanda Shinagawa-ku Tokyo JP
9                    Yuan                  Ai
designer              0512543734       F  690211  200020
10th Floor No. 334 Jiujiang Road Shanghai
10                   Nguyen                Anh Dung
                      0516232256       M  790815  70000
8A Ton Duc Thang Street District 1 HCMC Vietnam
11                   Sato                  Naoki
manager               027664545        M  810101  455-8205
3-23 Oye-cho Minato-ku Nagoya Aichi Japan
12                   Rodriguez             Aida
banker                023343214        F  810905  76152
3484 Taylor Street Dallas TX USA
13                   White                 Crystal
engineer              022320119        F  801230  WC2B 4BM
12th Floor Five Kemble Street London UK
14                   Kim                   Cheol-soo
banker                024720112        M  660508  135-740
222-55 Samsung-dong Gangnam-gu Seoul Korea
15                   Fedorov               Fyodor
manager               0518064398       M  750625  50696
No 6 Leboh Ampang 50100 Kuala Lumpur Malaysia
16                   Lefebvre              Daniel
planner               027544147        M  761225  21004
Chaussee de Wavre 114a 1050 Brussels Belgium
17                   Yoshida               Daichi
                      023543541        M  811001  530-0100
```

```
2-7 3-Chome-Kita Tenjinbashi Kita-ku Osaka
18                   Zhang                 Bao
engineer              024560207        F  840419  100008
2 Chaoyang Men Wai Street Chaoyang Beijing
19                   Pahlavi               Saeed
                      022371234        M  741231  20037
3300 L Street NW Washington DC USA
20                   Dubois                Alisee
webmaster             024560002        F  860405  1357
Chemin de Messidor 7-6 CH-1006 Lausanne Suisse
20 rows selected.
```

orders Table

```
iSQL> select * from orders;
ONO                  ORDER_DATE   ENO         CNO
------------------------------------------------------------------------
GNO         QTY         ARRIVAL_DATE PROCESSING
------------------------------------------------------
11290007             29-NOV-2011  12          3
A111100002  70          02-DEC-2011  C
11290011             29-NOV-2011  12          17
E111100001  1000        05-DEC-2011  D
11290100             29-NOV-2011  19          11
E111100001  500         07-DEC-2011  D
12100277             10-DEC-2011  19          5
D111100008  2500        12-DEC-2011  C
12300001             01-DEC-2011  19          1
D111100004  1000        02-JAN-2012  P
12300002             29-DEC-2011  12          2
C111100001  300         02-JAN-2012  P
12300003             29-DEC-2011  20          14
E111100002  900         02-JAN-2012  P
12300004             30-DEC-2011  20          15
D111100002  1000        02-JAN-2012  P
12300005             30-DEC-2011  19          4
D111100008  4000        02-JAN-2012  P
12300006             30-DEC-2011  20          13
A111100002  20          02-JAN-2012  P
12300007             30-DEC-2011  12          7
D111100002  2500        02-JAN-2012  P
12300008             30-DEC-2011  20          11
D111100011  300         02-JAN-2012  P
12300009             30-DEC-2011  20          19
D111100003  500         02-JAN-2012  P
12300010             30-DEC-2011  19          16
D111100010  2000        02-JAN-2012  P
12300011             30-DEC-2011  20          15
C111100001  1000        02-JAN-2012  P
12300012             30-DEC-2011  12          3
E111100012  1300        02-JAN-2012  P
12300013             30-DEC-2011  20          6
C111100001  5000        02-JAN-2012  P
12300014             30-DEC-2011  12          12
F111100001  800         02-JAN-2012  P
12310001             31-DEC-2011  20          15
A111100002  50          09-DEC-2011  O
12310002             31-DEC-2011  12          10
D111100008  10000       03-JAN-2012  O
12310003             31-DEC-2011  20          18
E111100009  1500        03-JAN-2012  O
12310004             31-DEC-2011  19          5
```

```
E111100010  5000        08-JAN-2012  O
12310005             31-DEC-2011  20          14
E111100007  940         03-JAN-2012  O
12310006             31-DEC-2011  20          2
D111100004  500         03-JAN-2012  O
12310007             31-DEC-2011  12          19
E111100012  1400        03-JAN-2012  O
12310008             31-DEC-2011  19          1
D111100003  100         03-JAN-2012  O
12310009             31-DEC-2011  12          5
E111100013  500         03-JAN-2012  O
12310010             31-DEC-2011  20          6
D111100010  1500        03-JAN-2012  O
12310011             31-DEC-2011  19          15
E111100012  10000       03-JAN-2012  O
12310012             31-DEC-2011  19          1
C111100001  250         03-JAN-2012  O
30 rows selected.
```

goods Table

```
iSQL> SELECT * FROM goods;
GOODS.GNO      GOODS.GNAME      GOODS.GOODS_LOCATION      GOODS.STOCK
-------------------------------------------------------------------------
GOODS.PRICE
--------------
A111100001     IM-300           AC0001                    1000
78000
A111100002     IM-310           DD0001                    100
98000
B111100001     NT-H5000         AC0002                    780
35800
C111100001     IT-U950          FA0001                    35000
7820.55
C111100002     IT-U200          AC0003                    1000
9455.21
D111100001     TM-H5000         AC0004                    7800
12000
D111100002     TM-T88           BF0001                    10000
72000


D111100003     TM-L60           BF0002                    650
45100
D111100004     TM-U950          DD0002                    8000
96200
D111100005     TM-U925          AC0005                    9800
23000
D111100006     TM-U375          EB0001                    1200
57400
D111100007     TM-U325          EB0002                    20000
84500
D111100008     TM-U200          AC0006                    61000
10000
D111100009     TM-U300          DD0003                    9000
50000
D111100010     TM-U590          DD0004                    7900
36800
D111100011     TM-U295          FA0002                    1000
45600
E111100001     M-T245           AC0007                    900
2290.54
```

```
E111100002     M-150           FD0001                         4300
7527.35
E111100003     M-180           BF0003                         1000
2300.55
E111100004     M-190G          CE0001                         88000
5638.76
E111100005     M-U310          CE0002                         11200
1450.5
E111100006     M-T153          FD0002                         900
2338.62
E111100007     M-T102          BF0004                         7890
966.99
E111100008     M-T500          EB0003                         5000
1000.54
E111100009     M-T300          FA0003                         7000
3099.88
E111100010     M-T260          AC0008                         4000
9200.5
E111100011     M-780           AC0009                         9800
9832.98
E111100012     M-U420          CE0003                         43200
3566.78
E111100013     M-U290          FD0003                         12000
1295.44
F111100001     AU-100          AC0010                         10000
100000
30 rows selected.
```

dual Table

```
iSQL> SELECT * FROM dual;
DUAL.X
------------
X
selected row count [1]
```

Index