ORACLE SERVICE CLOUD GUIDE: HOW TO IMPROVE REPORTING PERFORMANCE

Best Practices to Scale Oracle Service Cloud Analytics for High Performance

ORACLE WHITE PAPER | MARCH 2015

Table of Contents

Target Audience

This Oracle Service Cloud Guide provides best practices to assist report developers, technical analysts and business analysts configure Oracle Service Cloud Analytics reports to run quickly and efficiently, particularly when scaling to handle large data volumes or highly intensive processing scenarios. After reading this Guide, the audience will understand tools, techniques and other technical best practices to employ for optimizing reporting performance and related scalability.

Executive Overview / Abstract

As organizations scale their use of Oracle Service Cloud, demands placed on Oracle Service Cloud's reporting and analytics capabilities increase. Reports process larger data volumes as more data is gathered within the Oracle Service Cloud database, reporting requirements become increasingly sophisticated, and processing needs become more intensive overall. Also, regardless of scale, report designs can simply be inefficient. As a result, reports may sometimes run slowly, or may be stopped from running altogether due to built-in system thresholds. Fortunately, many best practices and report tuning approaches are available to build reports that scale with your Oracle Service Cloud deployment.

This paper discusses best practices for developing reports and analyses that run fast and efficiently within Oracle Service Cloud. Particular challenges addressed include scaling reports to handle large data volumes and highly-intensive report queries, identifying and fixing inefficient report design, and how to overcome reporting threshold governors when encountered. While the primary focus of this discussion is Oracle Service Cloud Analytics, related products and tools will be addressed when applicable.

The best practices presented in this Guide can be applied to develop new reports or to tune existing reports. As applicable, any differences in functionality among versions of Oracle Service Cloud will be addressed, as will the benefits and any potential drawbacks of a best practice.

Before reading further, be aware of the following points about some material presented throughout this Guide:

- Screenshot illustrations and menu paths referenced reflect the Oracle Service Cloud February 2015 release
- Several references to Oracle Service Cloud Analytics documentation are made in order to explore additional information about certain topics. Documentation for your release of Oracle Service Cloud is available by viewing Answer #5168 Documentation for Oracle Service Cloud Products, or from the "Help" link within your Oracle Service Cloud application. Most Oracle Service Cloud documentation, including the previous Oracle Service Cloud Analytics Manual, is consolidated into one comprehensive online Oracle Service Cloud documentation solution.

Understanding the Problem

Oracle Service Cloud Analytics delivers real-time operational analyses, reporting and monitoring from activity recorded in the Oracle Service Cloud database. This provides executives, managers and individual contributors actionable insights to help them make timely, informed and effective business decisions for improving business performance and delivering an outstanding customer experience across all interaction channels.

Naturally, when reports and analyses are run in Oracle Service Cloud Analytics, the insights they provide should be delivered to users quickly, particularly when high performance is a critical business requirement for the reporting need. Additionally, solution capabilities used to deliver these insights must increasingly scale as the size and usage of a Oracle Service Cloud deployment increases. Consequently, achieving high performance and scalability in Oracle Service Cloud Analytics is a key goal of any Oracle Service Cloud deployment.

If the goals of high performance and scalability aren't achieved, then several undesirable ramifications result. In particular, poorly performing reports – reports that run slowly or take a long time to return results – can:

» Frustrate the user running the report, causing a poor user experience

» Degrade the performance of your overall Oracle Service Cloud production system, which your customers, agents and other employees are using, causing a poor experience for them. This occurs because your live production system shares the same resources as your reporting system. If too many shared resources are consumed for reporting, then your live production site can slow down.

» Be susceptible to queuing, meaning they may be put in a queue to run later on a different server so they don't timeout. This means the end-user has to wait for the report to run, which again can provide for a poor user experience.

A report can perform poorly for several reasons. Examples include –

» The report isn't designed or optimized to run as efficiently as possible

» The report must analyze and process a large amount of data to return results

» The report requires highly-intensive processing to compute calculations, massage data and otherwise return results

» The report requires significant resources being shared with or consumed by other system processes

» The report encounters limits from the structure of the database schema it's run against, which can be designed to balance the need for reporting with competing needs

» The report is leveraging a platform optimized for purposes other than reporting

A common driver of most poorly performing reports is that their queries result in highly intensive processing. Therefore, to make these reports run faster, their processing requirements must be reduced if possible.

In some cases, a tradeoff must be made between the degree of value a report provides and the time it takes to execute the report. For example, sometimes reports with the most intensive processing requirements are those which uncover the most hidden, obscure or least intuitive insights that otherwise would be difficult, if not impossible, to know. These reports, despite their processing requirements, provide valuable information supporting better, smarter and more impactful decision-making.

Fortunately, there are several tools, configuration techniques and other best practices available from Oracle Service Cloud to help improve reporting performance and scalability. Before discussing them some key terminology and concepts should be defined.

Key Terminology and Concepts

To understand the best practices outlined in this Oracle Service Cloud Guide, it's important to first become familiar with some relevant terminology and concepts referenced by the best practices later in this document. Key terms and concepts are outlined below.

Note: Readers at least moderately familiar Oracle Service Cloud Analytics may already know much of the terminology and concepts presented in this section. These readers should feel free to skip this section and move on to the next. However, reviewing this section will provide helpful context for the rest of this document.

1. Database Types

Oracle Service Cloud provides three types of databases for reporting. Each has a different purpose.

» **The Operational Database**, also called the Oracle Service Cloud database, is the live production database recording all interactions processed from customers, agents, other employees, etc., in real-time. When Oracle Service Cloud Analytics points to this database, it's reporting on live, real-time production data.

» The Operational Database is **a transactional database**. Transactional databases are designed to process individual transactions quickly and efficiently. This design allows you to interact with your customers quickly when they contact you, provide them fast answers, and to process their related interaction records quickly.

» **The Report Database** (also referred to as the replication database or replication server) is a replicated near-live copy of the Operational Database. It shifts the processing of some Oracle Service Cloud Analytics reports off the Operational Database so they run faster and don't impact performance of the live production system. Even though the Report Database is a replication of the Operational Database, it isn't used to process transactions like the Operational Database. Rather, it's specifically reserved to perform processing-intensive activities that shouldn't be performed on the Operational Database.

  The Report Database is constantly being updated by the Operational Database. Typically it is only a few seconds behind the Operational Database, but longer periods of data latency can sometimes occur temporarily. (See Answer #2159 on the Oracle Service Cloud Customer Support site for more information).

Reporting Implications: Transactional databases are primarily designed for transactional processing. They process single transactions fast and efficiently. Reporting and analyses, on the other hand, process large volumes of data (rather than single transactions), aggregate the data, and then summarize it into meaningful insights or metrics. Consequently, reporting systems sometimes don't use a transactional database because reporting on very large volumes of data is not what transactional databases are designed to provide. As a result, reporting systems sometimes report from a database designed specifically for reporting, such as a data warehouse.

Regardless, a great deal of reporting is still performed from transactional databases without noticeable performance impact. In fact, when reports reflecting real-time data are a necessity, a transactional database is likely the best data source.

At a certain point however, running reports from a transactional database can impact performance, either of the report itself or of the operational system. Factors impacting performance include the volume of data the report is analyzing, the volume of data in the entire system, the efficiency of the query generated to run the report, schema design, the calculations and metrics the report is returning, activity taking place in the system, system resource allocation at report run-time and more.

When performance begins to be an issue, it's necessary to make decisions as to how to handle such a report. For example, you could decide to run the report from an offline replication of the live production database (e.g. the Report Database). While this provides the advantages of shifting processing off the live production system to another system, it's still ultimately a transactional database and won't resolve all performance situations.

You could also utilize a data warehouse for large volume reporting as long as the minor data latency reflected in the data warehouse is acceptable.

Alternatively, you could decide to run the report as is, using as many best practices addressed in this document as possible, if the advantages of running the report outweigh the potential performance impact of doing so.

Ultimately, delivering high-performance reporting, particularly when large data volumes are involved, requires trade-offs, and weighing the advantages gained in one area for perhaps sacrifices forfeited elsewhere.

Best practices for choosing the right database to use for a report are presented in the following best practice section "Use the Report Database to Report on Large Data Volumes".

2. Queued and Deferred Reports

Reports can take a long time to run if their underlying database queries will access a large amount of information or otherwise require intensive processing. Oracle Service Cloud Analytics provides a report queuing feature for reports with intense queries so users can continue to work in Oracle Service Cloud while the reports run in the background on a different server. Queuing a report can avoid a time-out error message and permit the report to run when it requires significant processing resources.

When a queued report has finished running the user receives a notification and can open the report immediately to view it. This is performed either directly from the notification message, or via the “My Queued Reports” standard report accessed from the Reports Explorer > Public Reports > Common > Site Administration > Reports.

A user can also see which reports are still in the queue, and their status in the queue, via the “Report – Waiting in Queue” report accessed from the same location referenced above.

Reports can be subject to automatic queuing, manual queuing, or deferred execution.

Automatic Queuing. Typically a report is queued when the user permits it to be queued at report run-time in response to a prompt. Specifically, when an attempt is made to open a report, the database query for the report is checked. If the system determines that the query will likely take too long to process, the user may be prompted with the message “Unable to process report. This query requires too much processing time to complete. Would you like to queue the report with the current search criteria?” If the query exceeds the number of rows analyzed threshold the user may be prompted with the message “Unable to Process Report. The report could not be processed because it exceeds the query size limitation. Please reduce the amount of data processed by the report. Would you like to queue the report with the current search criteria? If you choose to cancel the report no data will be returned.” If the user responds in the affirmative in either situation, the report will be queued. Otherwise, more restrictive filtering criteria must be specified to narrow analysis so that the report will immediately run. Once a report is queued, a message displays an estimate of how long it should take to process the report. As previously mentioned, once the report is finished running, the user can open the report via the “My Queued Reports” report accessed either directly from the notification message or Reports Explorer

Specific criteria used to determine whether or not Oracle Service Cloud will display the prompt that permits a report to be queued a report are outlined in the following best practice section “Use Reporting Thresholds to Your Advantage.” Generally speaking, if the number of rows required to process the report is between a certain range, and the amount of time required to process the report exceeds a certain threshold, then the report will be prompted for queuing. However, if the number of rows required to process the report exceeds a maximum threshold, then the report can neither be queued nor run, and more restrictive filtering is required.

Once a report is automatically queued, it is also automatically set for deferred execution every time it is run thereafter, since it is assumed the report’s filters and data set result in long run times. Deferred execution is described in more detail below, including how to clear the deferred execution state if preferred.

Manual Queuing. Reports can be queued manually before attempting to run them if the user thinks they will query or return large amounts of data. Manual queuing is also helpful if the user wants to view a number of reports but doesn’t want to wait for each to generate. For instructions on how to manually queue a report, see the “Manual queuing” section in Oracle Service Cloud Analytics documentation.

Deferred Execution. A report set for deferred execution is one that is pre-specified as likely to require queuing when run. If the report is deemed to require queuing at run time, then the user will be prompted to permit the queuing just like in any other report that requires queuing. More about this is described in the best practices section “Use Reporting Thresholds to Your Advantage,” and in the following section “Reporting Thresholds and Governors.” Note that a deferred execution report may run immediately, without prompting the user to permit queuing, if Oracle Service Cloud determines at run-time that the report can be executed without significant processing resources.

The advantage of setting a report for deferred execution is that Oracle Service Cloud Analytics requires less time and processing resources at report run-time to determine if it needs queuing.

Best practices pertaining to queued and deferred reports are presented in the following best practices section “Queue Large Reports” and “Use Reporting Thresholds to Your Advantage.”

More Information. For other information about queued and deferred reports, see the “Queuing Reports” section of the Oracle Service Cloud Analytics documentation, or Answer ID #2776 “Common questions regarding queued reports” on the Oracle Service Cloud Customer Support site.

Reporting Thresholds and Governors

In order to avoid long-running reports, and to ensure system resources aren't overly allocated to certain reporting functions at the expense of other functions, Oracle Service Cloud applies processing threshold limits to reports. The application of thresholds balances the need to run large or processing-intensive reports against the competing need to provide a highly performant Oracle Service Cloud operational environment for customers, agents and other system users.

Thresholds can govern many functions, including whether a report will run immediately, will be queued to run later when more processing capacity is available, or cannot run at all. Other threshold limits guide the number of rows that can return to the client for viewing, and the number of rows that can be exported into a file.

A brief definition of each threshold is discussed below. *Most of these are referenced later in the document, so it's a good idea to get familiar with them now.*

Note: Some threshold limits can be modified, either by customers or by Oracle Service Cloud, when appropriate. To discuss the potential to modify thresholds for your Oracle Service Cloud implementation to achieve certain reporting goals, please contact Oracle Service Cloud Customer Care. Note that increasing a threshold can impact how other parts of your Oracle Service Cloud application operate, including performance, so changing a threshold must be done with careful consideration and usually should only be done on a temporary basis.

1. Database Row Thresholds

Database row thresholds determine the maximum *estimated* number of database rows that can be *analyzed* to return a result set that will be displayed in a report. Note that this threshold is not necessarily a function of the actual number of rows returned for the report. Rather, it is the number of database rows Oracle Service Cloud will *analyze* to return a result set that will eventually show on a report. There are different thresholds (also known in more technical terms as the maximum join size) for different databases.

» The **operational threshold** is the database row threshold applied when executing a report against the Oracle Service Cloud Operational Database. In Oracle Service Cloud Analytics the operational threshold is 2 million rows.

» The **reporting threshold** is the database row threshold applied when executing a report against the Oracle Service Cloud Report Database. In Oracle Service Cloud Analytics the reporting threshold is 5 million rows.

Additionally, the **deferred report threshold** is the database row threshold applied to deferred execution reports, regardless of whether they are executed against the Operational Database or Report Database. In Oracle Service Cloud Analytics, the deferred report threshold is 200,000 rows.

Implications. Since the reporting threshold is greater than the operational threshold, Oracle Service Cloud Analytics reports processing larger data volumes are more likely to run on the Report Database than on the Operational Database. The report threshold is higher because use of the Report Database has no impact on performance of the Oracle Service Cloud production system, and because fewer activities are competing for resources on the Report Database. Consequently, it's typically advantageous to run larger or intensive Oracle Service Cloud Analytics reports from the Report Database rather than from the Operational Database. However, for even greater scalability, use of a data warehouse should be considered.

Warning Messages: If a report cannot run because it will exceed a database row threshold, a message such as "Query processes too much data" will display.

See the following best practice section "Use Reporting Thresholds to Your Advantage" for information about how to leverage your knowledge of thresholds in report building.

2. Deferred Report Execution Time Threshold

The deferred report execution time threshold governs how much time Oracle Service Cloud Analytics will spend processing a report in real-time. If the time to process a report exceeds the threshold, the report query is stopped and no data is returned. The deferred report execution time threshold is **60 seconds**.

See the following best practice section "Use Reporting Thresholds to Your Advantage" for information about how to leverage your knowledge of thresholds in report building.

3. Row Export Threshold

The row export threshold governs how many report rows can be exported from Oracle Service Cloud Analytics to a file. In Oracle Service Cloud, the maximum row export threshold is **100,000 rows**. This threshold can be lower depending on the hardware limitations of a client's machine, the number of columns exported, and the type of columns exported.

Beginning in the February 2014 release an option was added to allow export to a compressed CSV file. This export option allows a user to export 1 million rows or 100 Mb of data (whichever limit is reached first). If the 100 Mb file size is reached first the data in the file is truncated exactly at the 100 Mb file size. The data set truncated at the end of the row where 100 MB file size is reached. The entire set of data for that row is included in the file (i.e. not truncated mid-row). For this reason, this last line of the data file can be considered complete if the data is being parsed or used for import to another system.

See the following best practice section "Use of Data Export Tools" for related information.

4. Report Row Viewing Threshold

The report row viewing threshold determines the number of rows a report can return to the Oracle Service Cloud client desktop to be viewed and displayed online. (Note this differs from the row export threshold which pertains to exporting output as opposed to online viewing of output). The default threshold is **10,000 rows**. If the number of rows returned in a report exceeds this number, paging will automatically be enabled with a rows-per-page value of 10,000.

Limiting the number of rows displayed prevents inefficient reports from returning more rows than can be quickly processed and displayed.

If needed, this threshold can be increased via the VRL_HARD system configuration setting. This is found in Configurations > Site Configuration > Configuration Settings > Search "Key" for VRL_HARD. Oracle Service Cloud recommends that any increase be temporary only, and then returned to the 10,000 row default after the larger report has been consumed. Note that increasing this threshold can impact performance elsewhere.

See the following best practice section "Use of Data Export Tools" for related information.

5. Maximum Analytics File Size Threshold

The maximum analytics file size threshold governs the maximum data set size that can be returned to the Oracle Service Cloud client desktop when running a report. The default threshold is 15MB, and the maximum threshold is 2 GB. If exceeded, a message will display stating that the maximum data set size is exceeded, and the report will not execute. More information about this setting is available in Answer ID #2223 "Analytics Error: Data Set Has Exceeded Maximum Size" on the Oracle Service Cloud Customer Support site. This threshold is configurable via the MAX_ANALYTICS_FILE_SZ setting. This is found in Configurations > Site Configuration > Configuration Settings > Search "Key" for MAX_ANALYTICS_FILE_SZ. Note: This configuration setting does not impact the 1 million row/100 Mb limit for the CSV export option discussed above and expanded upon in the section "Use of Data Export Tools".

6. Summary

This section described how types of databases, queued reports, deferred reports, and threshold governors impact your reports and analyses. The next section will apply this knowledge, and more, in outlining best practices to optimize reporting performance.

Addressing Performance Problems through Best Practices

This section outlines the top 15 best practices to configure Oracle Service Cloud Analytics reports for improved performance and scalability. These best practices particularly apply for reports accessing larger data volumes, that require intensive processing, or that are designed to run less efficiently than possible. These best practices are based on Oracle Service Cloud's observations in working with thousands of customers over several years. Adhering to these best practices will help your reports run faster and otherwise improve their scalability.

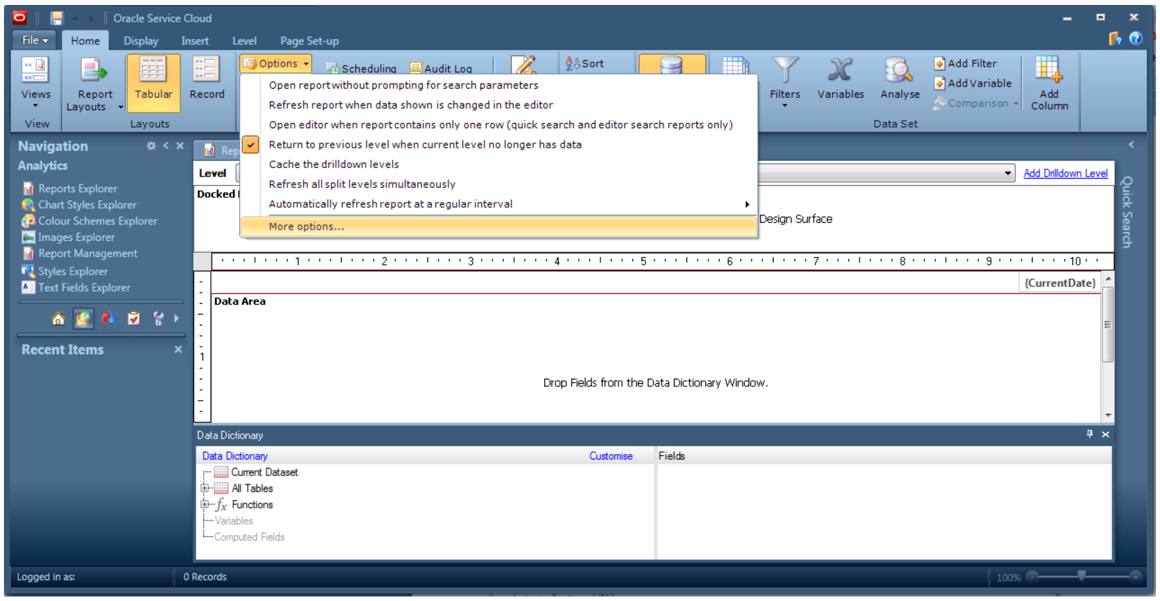
At A Glance – Top Best Practices to Improve Reporting Performance and Scalability

1. Use the Report Database to Report On Large Data Volumes

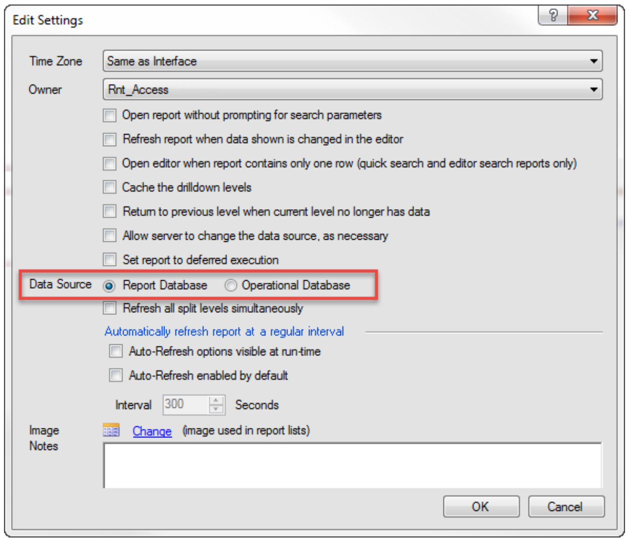
Oracle Service Cloud provides two types of databases to report from: the Operational Database and the Report Database. These databases, and the implications of reporting on each, are described in the prior section on "Key Terminology and Concepts."

When possible, configure Oracle Service Cloud Analytics custom reports examining large data sets to run against the Report Database. This especially applies for custom reports accessing larger data volumes and that in other ways require significant processing power. Use of the Report Database shifts report processing off live Operational Database to a replication database so that reports will run faster and not potentially impact performance of the live production system.

You can specify a custom report to run against the Report Database in the report design center's Edit Settings window, accessed from the "Option > More options" menu within the Home tab of the report designer. See the screenshots below. For more information about this configuration setting, see the "Report databases" section of the Oracle Service Cloud Analytics documentation.



Clicking “More options…” will open the Edit Settings window, where “Report Database” should be selected as the Data Source:



Edit Settings

Time Zone: Same as Interface

Owner: Rnt_Access

- ☐ Open report without prompting for search parameters
- ☐ Refresh report when data shown is changed in the editor
- ☐ Open editor when report contains only one row (quick search and editor search reports only)
- ☐ Cache the drilldown levels
- ☐ Return to previous level when current level no longer has data
- ☐ Allow server to change the data source, as necessary
- ☐ Set report to deferred execution

Data Source: [✓ Report Database] Operational Database

- ☐ Refresh all split levels simultaneously

Automatically refresh report at a regular interval

- ☐ Auto-Refresh options visible at run-time
- ☐ Auto-Refresh enabled by default

Interval 300 Seconds

Image: Change (image used in report lists)

Notes

OK Cancel

While most of us inherently prefer reports reflecting data from the Operational Database, the performance costs and other implications of doing so must also be considered. Use of the Report Database is often a better overall option. By reporting against the Report Database, you a) get near real-time information; b) benefit from higher reporting thresholds which allow larger data volumes to be processed; c) reduce the load on your live production system; and d) typically will achieve faster running reports.

When to Use the Operational Database

Report against the Operational Database when reporting on lower data volumes or when reports use less processing-intensive report queries. Also, as mentioned in a prior section, while the Report Database typically lags the Operational Database by only a few seconds, it can be longer at times. Therefore, report against the Operational Database when up-to-the-second, real-time information is essential for the report.

As a general rule of thumb, when a report query will analyze less than 2 million rows to generate report output, the Operational Database can be used for reporting (2 million rows is the operational database row threshold). Otherwise, the Report Database is ideal. This is a general rule of thumb however, and in some instances the Report Database may be better, or even required, even if less than 2 million rows are analyzed by a report query.

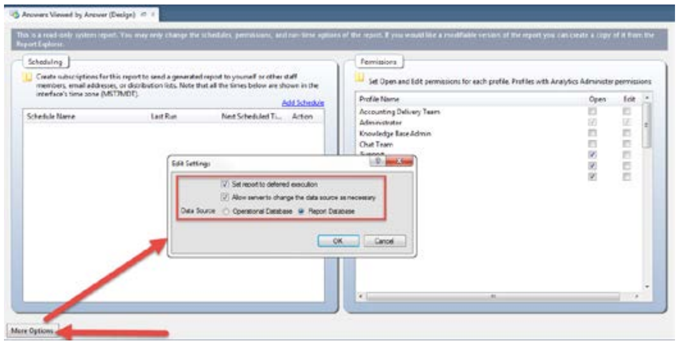
If preferred, custom reports can be configured to point to the Operational Database by default but shifted to the Report Database if necessary. Doing this tells Oracle Service Cloud to dynamically determine, at report run-time, whether the Operational Database can be used or if the Report Database is required. This configuration is achieved by first selecting the Operational Database as the default data source, and then checking the “Allow server to change the data source as necessary” box (see “Data Source” section of prior “Edit Settings” window screenshot). Selecting this option is ideal when use of the Operational Database is generally preferred for reports.

Oracle Service Cloud will shift reports to the Report Database if Oracle Service Cloud determines that the report query will examine more than the operational database reporting threshold of 2 million rows.

Oracle Service Cloud automatically checks the “Allow server to change the data source as necessary” box when a custom report is created. Report developers can uncheck this box to force the report to run only against the data source selected by the developer if desired. For example, if the report should only run against the Operational Database, then the data source should be set as the Operational Database, and this box should be unchecked.

Standard Reports

Standard reports in Oracle Service Cloud point to the Operational Database by default. Beginning with the Oracle Service Cloud November 2009 release, standard reports are automatically shifted to the Report Database if the reports will examine more than 2 million rows (see process described immediately above). If a standard report is set to deferred execution the report can be edited to modify this setting. Edit the standard report in the reports explorer and select the More button in the lower left hand corner of the edit screen. This brings up the settings window for the standard reports that allows you to modify settings for standard reports regarding which database server to use for the report, whether it is deferred, and if the server can automatically allow the report to be deferred if it exceeds the specific database server thresholds. This is show in the following screen shot.



This is a read-only system report. You may only change the schedules, permissions, and run-time options of the report. If you would like a modifiable version of the report you can create a copy of it from the Report Explorer.

Create subscriptions for this report to send a generated report to yourself or other staff members, email addresses, or distribution lists. Note that all the times below are shown in the interface's time zone (MST7MDT).

Set Open and Edit permissions for each profile. Profiles with Analytics Administer permissions

Edit Settings

☑ Set report to deferred execution

☑ Allow server to change the data source as necessary

Data Source: Operational Database [✓ Report Database]

Standard reports can also be configured differently by report developers if preferred. This is achieved by copying a standard report to a custom report, configuring the custom report as needed, and then running the custom report in lieu of the standard report.

Report Examples

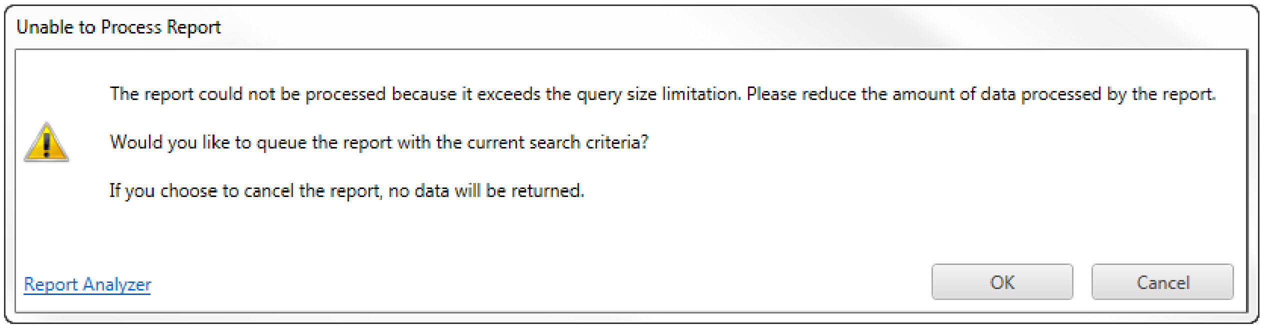
The “My Inbox” report is an example of a report that should run on the Operational Database, because its purpose is to provide the latest Inbox data to the report recipient. In this scenario, there is an important need for real-time information, and, it also runs on a small amount of data.

In contrast, a report that analyzes incident creation rates over several months is better implemented with the Report Database, because this report not only processes a large amount of data, but is providing a historical trend where real-time up-to-the-second data isn’t essential.

2. Queue Large Reports

Reports that require intensive processing should be queued so that they run in the background (on a different server) and don't timeout. Consequently, as a general rule, when Oracle Service Cloud Analytics prompts a user to permit that a report be queued, the user should accept this request. Otherwise, the user will need to apply more restrictive filter criteria to the report to narrow the analysis delivered.

Reports that require intensive processing typically include large reports (e.g. reports with many output rows), reports that analyze a large amount of data to produce output (regardless of the number of rows output), reports with complex calculations, inefficiently designed reports, and more. Any report that generates the following message "Unable to process report. This query requires too much processing time to complete. Would you like to queue the report with the current search criteria?" (see below) is considered a processing-intensive report and should be permitted to queue.


Unable to Process Report

The report could not be processed because it exceeds the query size limitation. Please reduce the amount of data processed by the report.

Would you like to queue the report with the current search criteria?

If you choose to cancel the report, no data will be returned.

More information about queuing, including its advantages and types of queuing, is described in the prior section "Key Terminology and Concepts," subsection "Queued and Deferred Reports."

Other information about queuing is contained in the "Queuing Reports" section of the Oracle Service Cloud Analytics documentation, and in Answer ID #2776 "Common questions about queued reports" on the Oracle Service Cloud Customer Support site.

How to Avoid Queuing

The following best practice section "Use Reporting Thresholds to Your Advantage" outlines how Oracle Service Cloud determines whether or not a report must be queued by examining report characteristics in comparison to thresholds. See Figure A, "Executing an Oracle Service Cloud Analytics Report" in particular. As **Figure A** indicates, to avoid queuing, reports should analyze data sizes that don't exceed the thresholds associated with the database the report is pointing to. The report must also require less than 60 seconds of processing time (when pointing to the Report Database).

More generally, the smaller the data set analyzed (as determined primarily by the level of report filtering), and the more efficiently the report is designed to run, the less likely the report is to require queuing. Best practices to design reports that run the most efficiently are outlined by this Guide.

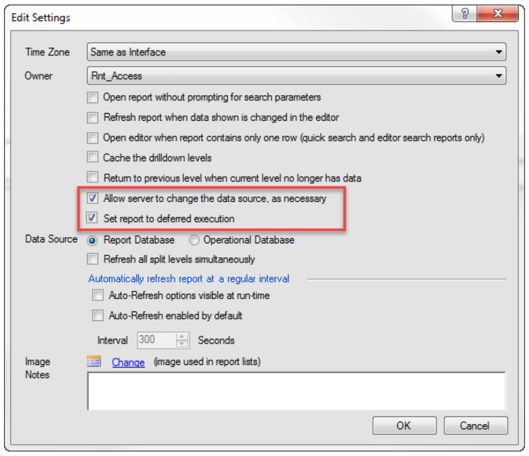
Note that a report can be queued from either the Operational Database or Report Database, depending on how the report is configured. Typically, reports are queued from the Report Database. However, if a report is configured to run only against the Operational Database, then it will be queued to run against that database if it is queued manually, or if it is set for deferred execution.

Keep in mind that queuing does not provide a means to run a report if the volume of data it analyzes exceeds the row threshold of the database it is run against. It simply shifts processing-intensive reports to the background so that they don't timeout, as long as they conform to database row thresholds and other criteria. Reports that exceed these thresholds won't run, and will require more restrictive filtering criteria.

Configuring Reports for Deferred Execution

As previously mentioned in the “Key Terminology and Concepts” section of this Guide, a report set for deferred execution is one that is pre-specified ahead-of-time as likely to require queuing when run. The advantage of setting a report for deferred execution is that Oracle Service Cloud Analytics requires less time and processing resources to determine if it needs queuing at report run-time.

Typically, Oracle Service Cloud Analytics automatically configures reports for deferred execution when they’ve been queued in the past. Reports can manually be configured for deferred execution also using the “Edit Settings” box of the report designer. Simply check the “Set report to deferred execution” as indicated in the following screenshot.



Edit Settings

Time Zone: Same as Interface

Owner: Rnt_Access

- [ ] Open report without prompting for search parameters
- [ ] Refresh report when data shown is changed in the editor
- [ ] Open editor when report contains only one row (quick search and editor search reports only)
- [ ] Cache the drilldown levels
- [ ] Return to previous level when current level no longer has data
- [✓] Allow server to change the data source, as necessary
- [✓] Set report to deferred execution

Data Source: [✓ Report Database] Operational Database

- [ ] Refresh all split levels simultaneously

Automatically refresh report at a regular interval

- [ ] Auto-Refresh options visible at run-time
- [ ] Auto-Refresh enabled by default

Interval 300 Seconds

Image Change (image used in report lists)

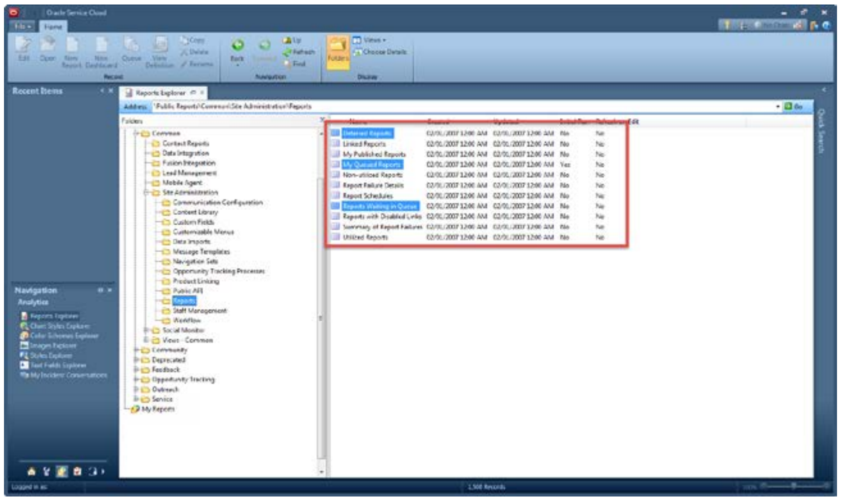
Notes

To keep the report from running in a deferred state, uncheck the “Set report to deferred execution” box. Keep in mind that if the report is queued in the future, Oracle Service Cloud will automatically recheck the box.

Once a report is marked for “deferred execution,” Oracle Service Cloud Analytics will use the smaller deferred maximum join size of 200,000 rows (see prior section on Key Terminology and Concepts”, and next best practice section “Use Reporting Thresholds to Your Advantage”) for the report. If Oracle Service Cloud Analytics deems the report will run without exceeding this threshold, then it will run immediately and not be prompted for queuing.

Finding Queued Reports

Oracle Service Cloud provides a set of reports to identify all of your implementation’s queued reports. Within the Report Explorer, they are available within Public Reports > Common > Site Administration > Reports. There are three reports: Deferred Reports, My Queued Reports, and Reports Waiting in Queue. These are highlighted in the screenshot below.



Name	Created	Updated	Initial Run	Refresh on Edit
Deferred Reports	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Linked Reports	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
My Published Reports	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
My Queued Reports	02/01/2007 12:00 AM	02/01/2007 12:00 AM	Yes	No
Non-utilized Reports	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Report Failure Details	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Report Schedules	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Reports Waiting in Queue	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Reports with Disabled Links	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Summary of Report Failures	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No
Utilized Reports	02/01/2007 12:00 AM	02/01/2007 12:00 AM	No	No

Alternative: The analytics_core table in Oracle Service Cloud shows all options mentioned above for each report. You can build an ad-hoc report showing these options using the analytics_core table. Specific information about this table and its columns is available in the Oracle Service Cloud data dictionary.

3. <u>Use Reporting Thresholds to Your Advantage</u>

Reporting thresholds have a key influence on how your report is run. Thresholds can dynamically shift a report from running on one database to another (i.e. from the Operational Database to the Report Database), determine if the report requires queuing, or if it must be filtered more granularly before running. A description of the key threshold governors impacting reports is outlined in the prior section "Reporting Thresholds and Governors". Become familiar with these concepts to understand the factors influencing your report. In particular, know -

- » The maximum join size (also referred to as the database row threshold) for:
  - The Operational Database is **2 million rows**
  - The Report Database is **5 million rows**
  - Deferred reports is **200,000 rows** (regardless of database report points to)
- » The report execution time threshold is **60 seconds**.

It is important to understand that the database thresholds are not necessarily a function of the actual number of rows returned for the report. Rather, these are the number of database rows Oracle Service Cloud will <u>*analyze*</u> to return a result set that will eventually show on a report. Since the database row threshold for the Report Database (5 million rows) is much larger than that for the Operational Database (2 million rows), it is more advantageous to run a report against the Report Database for reports that look at larger data volumes. For example, if the set of rows to be analyzed after all joins are performed to process the report will exceed 2 million rows, the Report Database should be selected as the report's data source.

Exceeding thresholds can sometimes result in reports being processed in ways the report developer or user doesn't prefer. To overcome this, make sure that as many best practices outlined in this Guide are applied so the report runs as efficiently as possible.

Understanding How Oracle Service Cloud Analytics Executes Reports

The flowchart depicted in Figure A illustrates how Oracle Service Cloud Analytics processes reports. Factors impacting how Oracle Service Cloud Analytics will optimally execute a report include the report's data source, data source row thresholds, queuing, expected execution time, and other parameters. In particular, it illustrates the impact of threshold governors, and highlights what happens when thresholds are exceeded. Note that the diagram is a general flowchart showing the most important flows, but isn't exhaustive.

Key takeaways of this diagram are:

» When running larger reports, it's more advantageous to use the Report Database than the Operational Database, and to allow queuing

» Smaller reports are more likely to run on the Operational Database than larger reports. Therefore, if the Operational Database is the preferred data source to report from, the report should be configured to analyze a small data set. Several best practices outlined in this document discuss how to configure a report to analyze smaller data volumes.

» Larger reports, and reports that require more than 60 seconds to run, are likely to require queuing as long as they don't exceed the maximum join size for the database they are run against.

» Reports that exceed the maximum join size won't run and will require more extensive filtering.

Understanding this process will help report developers configure reports to deliver the user experience desired.

Executing a Oracle Service Cloud Analytics Report

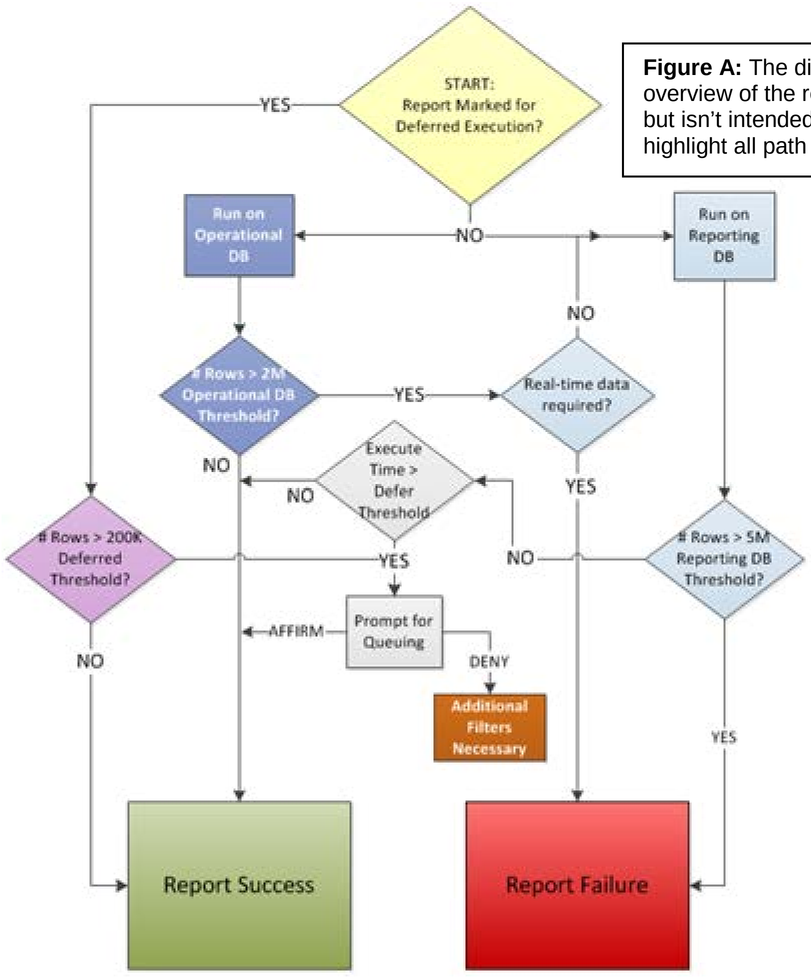

START: Report Marked for Deferred Execution?
YES
NO
Run on Operational DB
Run on Reporting DB
NO
# Rows > 2M Operational DB Threshold?
YES
Real-time data required?
NO
Execute Time > Defer Threshold
NO
YES
# Rows > 200K Deferred Threshold?
YES
NO
# Rows > 5M Reporting DB Threshold?
Prompt for Queuing
AFFIRM
DENY
Additional Filters Necessary
NO
YES
Report Success
Report Failure

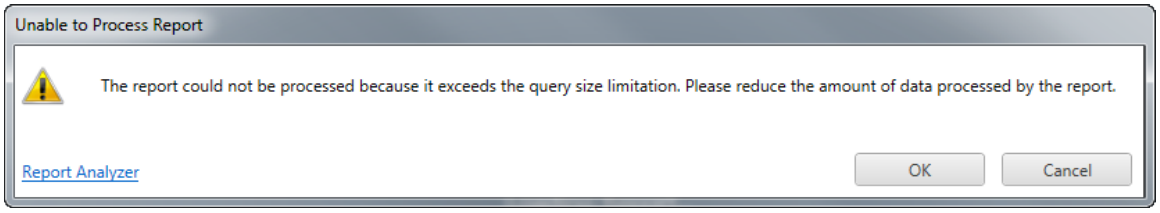
Figure A: The diagram provides an overview of the report's processing path, but isn't intended to exhaustively highlight all path conditions.

1) If the report is pre-configured for deferred execution, then depending on how large the data set to be processed is estimated to be, the report will either run right away or be prompted for queuing. In most cases, the user will be prompted to place the report in the queued state. Only where the report is estimated to run very quickly by not exceeding the deferred report threshold of 200,000 rows will the report be run immediately.

2) If the report is not pre-configured for deferred execution, then system will initially assign the report to run from the Operational Database unless the report is pre-configured to run from the Report Database.

3) If the report is assigned to run against the Operational Database, the system compares the number of database rows to be analyzed against the Operational Threshold. If this number is below Operational Threshold, the report succeeds.

4) If the report query exceeds the Operational Threshold, then the system will attempt to divert the report to execute against the Report Database. However, if the report is configured to run against the Operational Database, and the report developer has not selected the option to allow Oracle Service Cloud to change the data source as necessary (so that it can run against the Report Database), then the report will fail, and the following message will display.

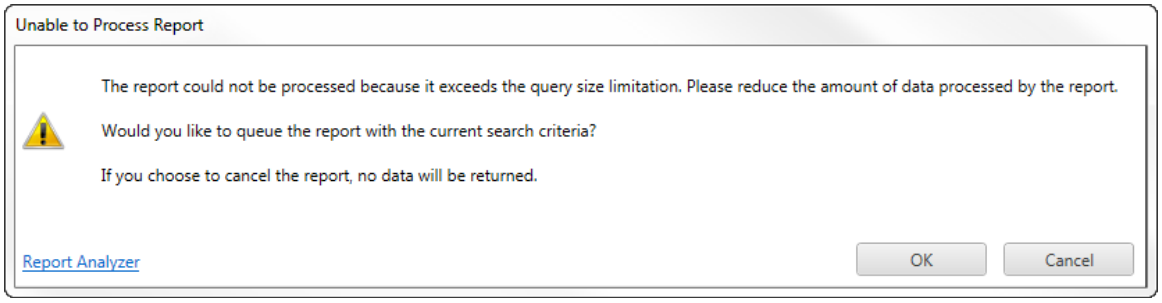

Unable to Process Report

The report could not be processed because it exceeds the query size limitation. Please reduce the amount of data processed by the report.

5) If the report is allowed to run against the Report Database, the system will then determine if the report will analyze fewer rows than the Reporting Threshold. If the number exceeds this threshold, the report will also fail, with a message such as "Query Processes too Much Data" similar to the screen shot in #4 above.

6) If the number of rows to analyze is fewer than the Reporting Threshold, a final check is made to determine if the likely reporting execution time will exceed the Execution Time Threshold of 60 seconds. If it won't, the report will run and the results will be returned from the Reporting Database.

7) If the expected execution time is longer than the deferred report Execution Time Threshold, then the user is prompted to queue the report as shown in the message below. If the user affirms, the report will be executed in the background and a notification will be sent to the user when the report generation has completed. If the user declines the queuing request, the user must narrow reporting criteria or modify the report to add filters, change joins or otherwise reduce data set that is processed by the report.


Unable to Process Report

The report could not be processed because it exceeds the query size limitation. Please reduce the amount of data processed by the report.

Would you like to queue the report with the current search criteria?

If you choose to cancel the report, no data will be returned.

8) Reports that succeed remain subject to the report row viewing threshold of 10,000 rows, and the row export threshold of 100,000 rows. If either of these thresholds is exceeded, a relevant message will display, although report results will still be provided up to the threshold limit.

Increasing the Reporting Database Maximum Join Size Threshold

If your report won't run even after applying **all** applicable best practices in this guide, you may want to consider requesting Oracle Service Cloud to increase the maximum join size for your Report Database. This can be applied to your Oracle Service Cloud deployment as a whole, or to individual reports[1] , on either a temporary or permanent basis.

Any potential increase must be requested by contacting Oracle Service Cloud Customer Care to initiate the process. Oracle Service Cloud will investigate the situation and discuss with you the circumstances surrounding the request to arrive at a recommended action plan. Alternative recommendations may be suggested as a first next step. Keep in mind that increasing the maximum join size for the Report Database may impact performance of other parts of your Oracle Service Cloud deployment also using the Report Database.

The maximum join size for the Operational Database **cannot** be modified due to the impact that can have on the performance of your live production system.

1 The maximum join size for a report is contained in the Maximum Join Size field (max_join_sz), within the Reports table (analytics_core). A custom report can be built to access this table and provide the maximum join size for each report.

4. Use Report Filters to Optimize Performance and Utilization

Report filters help improve reporting performance by narrowing the report's analysis. The more restrictive the analysis (i.e. the higher the degree of filtering) the less data must be analyzed, and therefore the faster the report will run. Use report filters as much as possible, and configure them so that the report returns the smallest data set possible. For example, rather than reporting on an entire year's worth of data, report only on the time period with the most recent activity of interest.

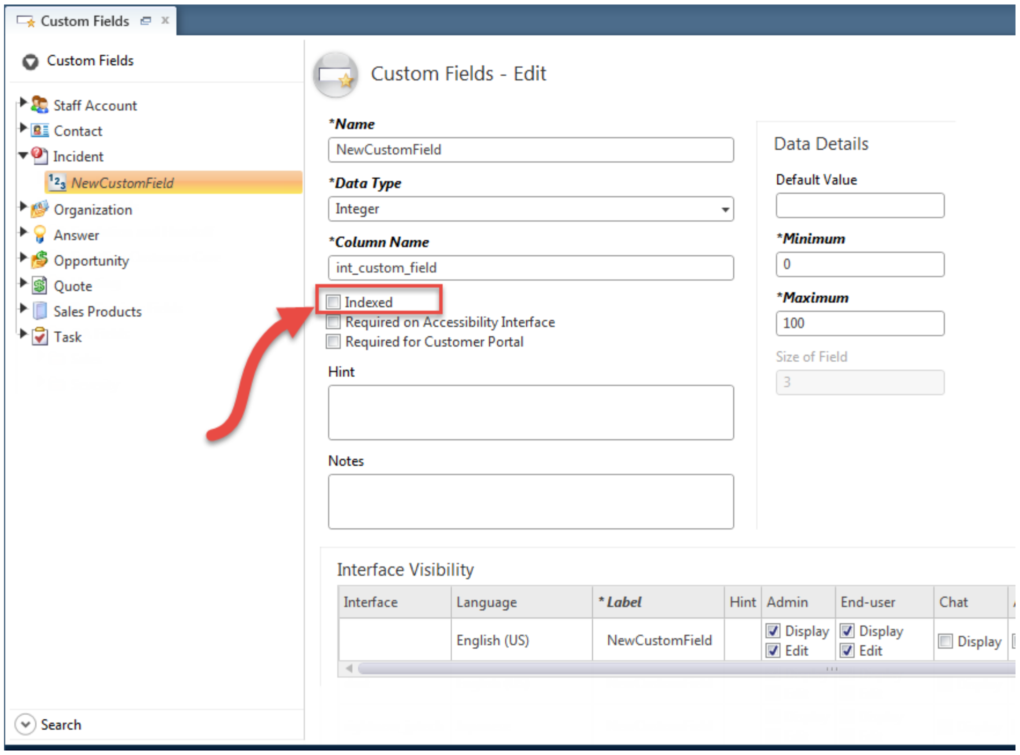
Optimizing Filter Configuration

While filtering typically improves reporting performance, configuring filters incorrectly can degrade performance. The following best practices describe how to configure report filters properly to deliver maximum reporting performance, and how to avoid typical cases where report filters can result in poorly performing reports.

a) Index custom fields used as filters. One of the most common reasons why reports perform poorly is that they are filtered on custom fields which aren't indexed. Generally speaking, an administrator should index custom fields used as report filters (when indexing is possible). Keep in mind that the more highly selective a filter is (i.e. the narrower it restricts the analysis), the greater impact adding an index to a field it references will typically have on improving performance.

   Note: Indexing is not available for fields using the text area, opt-in, or Yes/No data types.

   You can determine if a custom field is indexed by finding the field in Common Configuration > Custom Fields, and then examining whether or not the "Indexed" box is checked. See example below. To index the field, simply check the "Indexed" checkbox.



*Name: NewCustomField
*Data Type: Integer
*Column Name: int_custom_field
☐ Indexed
☐ Required on Accessibility Interface
☐ Required for Customer Portal
Hint:
Notes:

Data Details
Default Value:
*Minimum: 0
*Maximum: 100
Size of Field: 3

Interface Visibility

Interface	Language	*Label	Hint	Admin	End-user	Chat
	English (US)	NewCustomField		✓ Display ✓ Edit	✓ Display ✓ Edit	☐ Display

   More information about indexing a custom field can be found in Answer ID #1266 "Impact of indexing a custom field" on the Oracle Service Cloud Customer Support site.

b) Filter on standard fields which are indexed. Similar to the above best practice, ensure that standard fields (non-custom fields) used as filters are indexed. You can determine if a standard field is indexed by looking in the Common Configuration -> Database Administration -> Data Dictionary. Avoid filtering on standard fields that aren't indexed.

   If you're filtering on a standard field that isn't indexed, adding an index to that field may help improve the report's performance. To determine if adding an index is a right approach, contact Oracle Service Cloud Customer Care. Oracle Service Cloud can evaluate the need, examine the report query, and facilitate adding an index on that standard field if deemed necessary (or recommend another action). Again, as previously mentioned, keep in mind that the more highly selective a filter is, the greater impact adding an index to a field it references will typically have on improving performance.

c) Filter on simple columns, not on expressions. Filter on simple columns rather than on functions or expressions. Report queries can't utilize indexes when filters contain functions or expressions on the left hand side of the filter designer box. Therefore, if you need to filter on a function or expression, make sure this is absolutely necessary.

d) Use the "Like" instead of the Complex Expression operator. Whenever possible, use the Like operator instead of the Complex Expression operator. The Like operator requires the usage of wildcard characters (% or *) to perform partial searches. The Complex Expression operator automatically appends a wildcard character to search strings. Since the presence of wildcard characters reduces the performance of the report query, it is best to use them only when necessary. The Like operator helps to enforce this best practice. When using the Like operator be sure to place the wildcard only on the right side of the filter string being used (e.g. "table.column Like abc%"). If a wildcard exists on the left side of the string (e.g. "table.column Like %abc%"), indexing is ignored for this filter. This reduces performance of the report query.

e) Use fixed filters whenever possible. Where possible, use fixed filters to restrict records returned to a specific value or range determined by the report developer. Fixed filters cannot be changed or edited by the user when the report is run. This introduced greater control over the data set returned and therefore over performance of the report query.

   For example, reports or views that list all of today's solved or updated incidents could use a fixed filter for the status and last updated fields. For sites with multiple interfaces, consider adding fixed filters for the interface or language so that records from only that interface are accessible.

f) Provide default values for run-time filters. In cases where a fixed filter is not appropriate, you should still set a default value or range for the run-time filters. For example if your report uses the incident Date Created as a filter, set the default range to be within the last week or the last month so that the default search will return the most recent incidents that are typically of greatest interest.

   Configuring default values for your run-time filters reduces the number of rows that the query needs to evaluate. You can also configure default search values with text run-time filters such as "enter search text here". This forces staff members to type in what they are actually looking for before doing a search.

Optimizing Runtime Filter Layout and Utilization

Filter location and size is an important consideration to make when creating a report. This does not have performance impact at in a traditional database perspective during the report run, but rather it will have impact in how efficient your users are in running the report and selecting filter values. Improvements to the search window in the May 2014 release will help report administrators/creators to organize their filters as desired so that report users can be more efficient when running reports. Perhaps the administrator prefers the most often used filters at the top left of the filter window or maybe there are good logical filter groupings within the report.

The additions to the search designer (both in the report editor and at runtime) include the ability to:

- Reorder search filters.
- Resize search filters.
- Stack/group search filters together.
- Modify orientation of date range filters to be stacked or side by side.
- Save layouts for all users (report creator in the report editor).
- Save local changes to layouts (individual users on their machines).

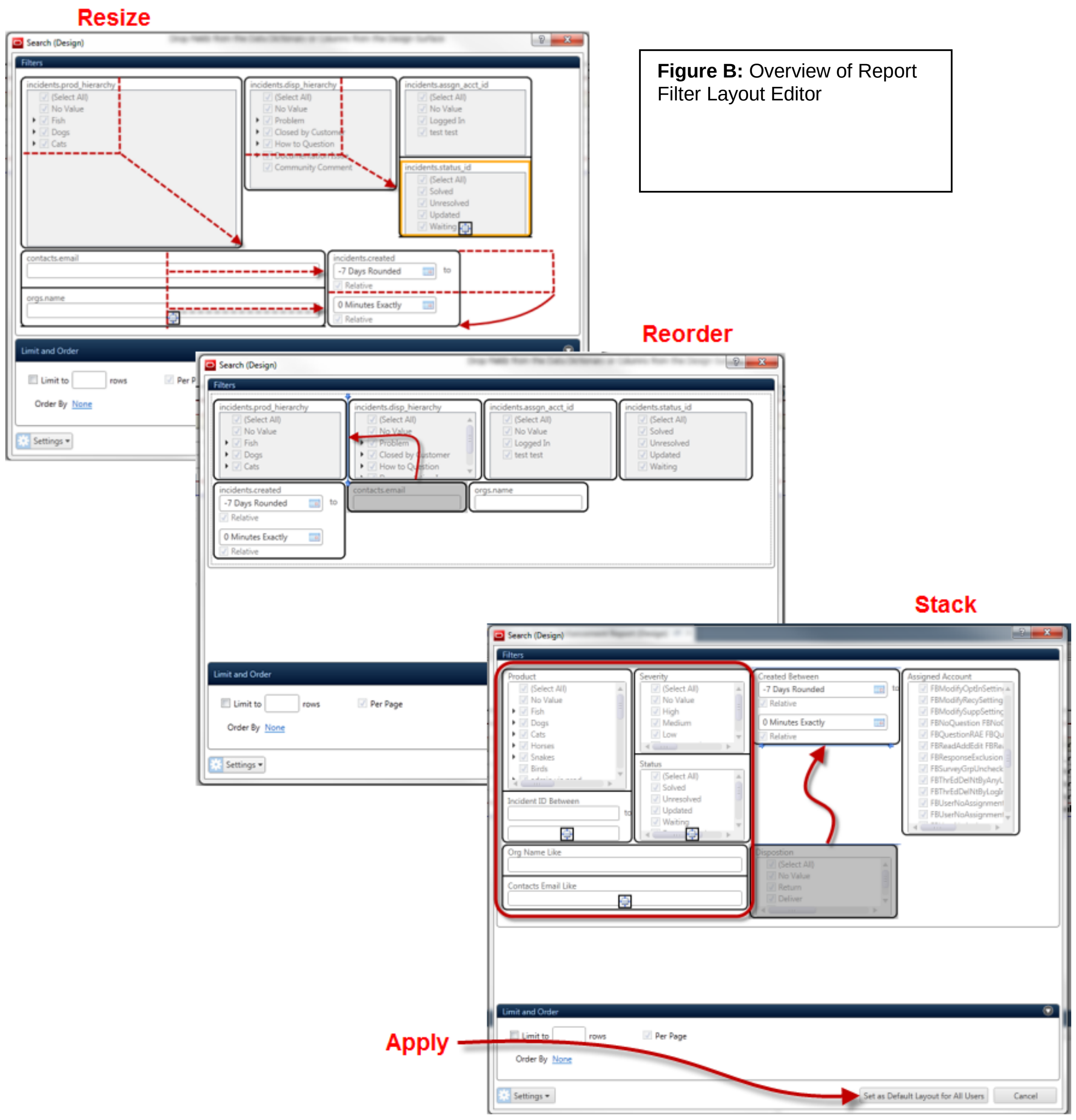
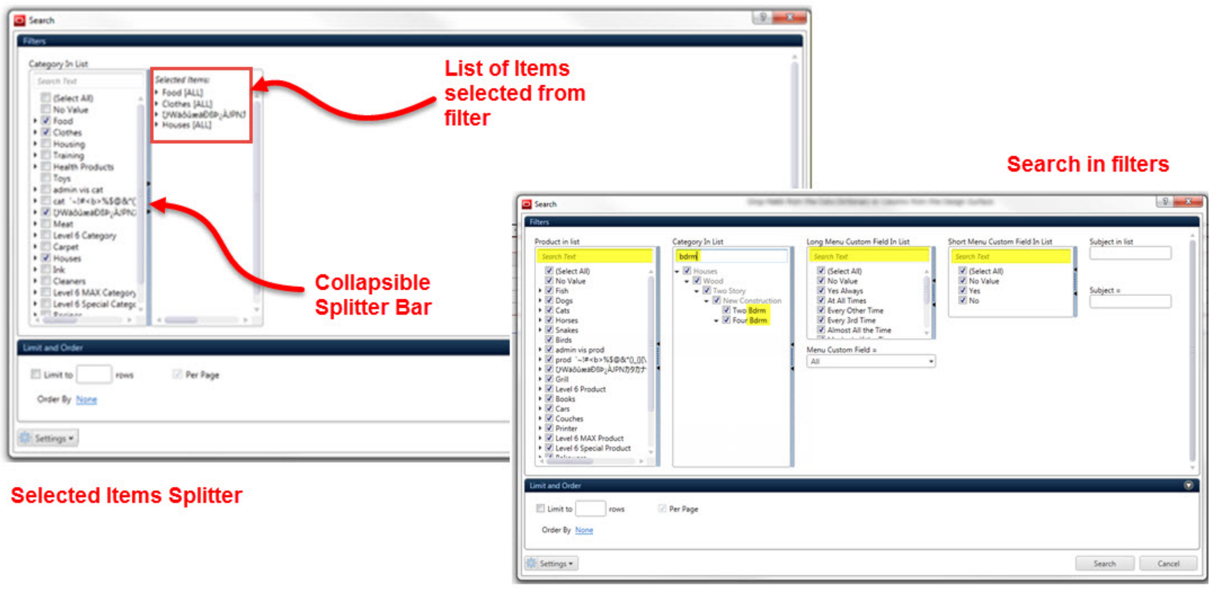


Figure B: Overview of Report Filter Layout Editor

In the Aug 2014 release the ability to search through large filters (e.g. long product/category/menu item custom fields) to be able to quickly and easily find individual filter items was added. In November 2014 release the selected items splitter was added so that a user can see items that are selected in these large/hierarchical lists).



The search window designer mode can be found by clicking on the gear/Settings button at the bottom left of the search window in both the report editor and at runtime. More information on the new functionality can be found in the Oracle Service Cloud documentation for May 14 and Aug 14 releases by searching for "customizing the search window".

All of this functionality adds up to a much better customizable layout for report creators and report users.

5. Avoid Use of Inefficient Functions

As previously mentioned in the report filters section, some functions are less efficient than others. Use of such functions can degrade performance if improperly used, so be aware of inefficient functions and look for alternatives if possible.

For example, sum_distinct is a powerful function, but it can have an adverse impact on reporting performance. Consider using alternative approaches to sum_distinct when possible. In cases where sum_distinct is used in a drill-down, you may be able to remove sum_distinct by converting drill-down levels to linked reports (e.g. use of report linking). For more information about report linking, see the following best practice item “Use Report Linking Rather Than Drill-Down Levels”, or the Oracle Service Cloud Analytics documentation.

6. Setup Table Join to Return the Smallest Result Set Possible

Table joins should be configured to return the smallest data set possible. The smaller the data set Oracle Service Cloud Analytics analyzes to process a report, the faster the report will run. The following best practices provide guidelines about how to achieve this, and how to avoid typical cases where join configurations can result in unnecessarily large data sets. A sound understanding of relational databases and SQL can benefit here.

a) Only join to tables needed for the report. For various reasons, sometimes report configurations include joins to tables that aren't actually used in the report. This should be avoided. When possible, eliminate joins and references to tables not used in the report.

Sometimes there is a valid reason why a report references columns from tables not used in the report. Keep in mind this will increase report processing time, and there may be ways to avoid doing this. For example, if you need to filter out rows in the parent table that do not have rows in another table, consider adding a NOT NULL filter on the join field in the parent table.

Sometimes extraneous joins exist because they were inherited from the definition of a report that was copied to build a new report. Make sure that when you re-use one report to build another, that you review the report configuration and eliminate tables, joins, columns, etc. that aren't required for the newer report. See the following related best practices:

- Use the Report Analyzer for Performance Tuning
- Section "Other Important Reporting Best Practices," item "Include Only Pertinent Functionality Within Your Report"

b) Use as few tables as possible. Design reports to require the fewest number of tables possible. The fewer tables required for a report, the fewer joins will be needed. This will reduce the number of rows to be analyzed to generate the report, and decrease the related processing required, which contributes to faster report processing. This objective must be balanced however with the insights the report must provide and related filtering needs.

c) Avoid joining large tables that aren't directly related to each other. Sometimes performance suffers when attempting to join data from very large tables that aren't directly related to each other (i.e. through a common key). If you join such tables together, add as many filters as possible to limit the result set.

An example is joining the transactions and inc_performance tables in the same report via the Incident record. This will create a very large dataset as the number of rows for a given incident will be equal to the number of transactions multiplied by the number of Inc_Performance records. If such joins are required, make sure it is absolutely necessary. To improve performance, consider adding a filter that provides one or more additional joins between the two tables. An example is adding a filter joining transactions.created to inc_performance.time_start.

d) Configure one-to-many joins properly. If the join between tables is a one-to-many join, use the table with the "one" record as the first table, and then join the table with the "many" records to it.

e) Configure one-to-one joins properly. If the join between tables is a one-to-one join, use the table with the fewer records (based on the filtering you intend to do) as the first table and the table with the more records as the second table.

f) Use inner joins rather than outer joins when possible. Inner joins return less data than outer joins. When possible, use inner joins. Oracle Service Cloud Analytics typically defaults to inner joins, but where configuration is applied or required, use inner joins when appropriate.

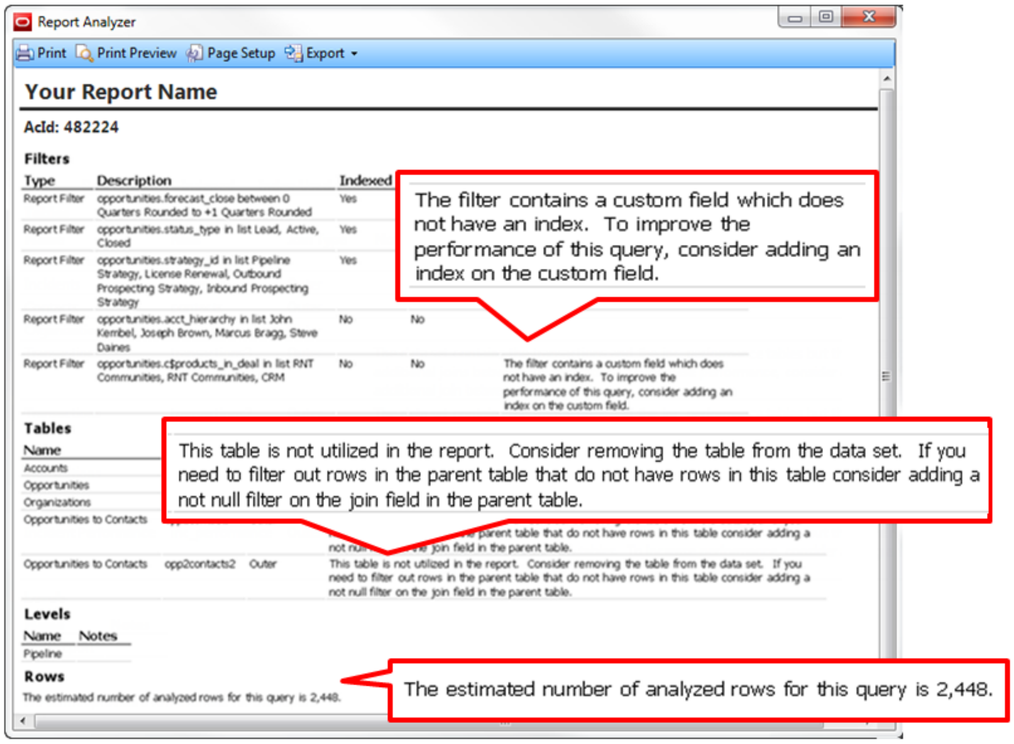
For example, suppose you need a list of incidents with associated organization information. If you join the Incidents table to the Organizations table using an outer join, the result set returned will be those incidents that have organization information as well as incidents that don't have organization information. However, if you use an inner join, the data returned will only be those incident records that contain organization information. Therefore, the resulting data set using an inner join will be smaller than with an outer join, and more targeted. If using an outer join, make sure you need it.

7. Use the Report Analyzer for Performance Tuning

The Report Analyzer examines a custom report and suggests how it can be tuned to run as quickly and efficiently as possible. Tables, rows, filters, joins, levels and other report components are examined and then compared against performance best practices. Using this information the Report Analyzer suggests how performance can be optimized when possible. While designing a report, you can interactively use the Report Analyzer while modifying the report design to see the impact of report on performance in real-time.

The Report Analyzer also shows the estimated number of rows that the report's query to the database will examine. This tells the report developer whether or not the report will exceed row thresholds and limits previously discussed in this document (see prior section "Reporting Thresholds and Governors," and prior best practice "Use Reporting Thresholds to Your Advantage.")

The screenshot below illustrates Report Analyzer output for one custom report. Suggested improvements are provided in the "Notes" column. Other key information provided by the Report Analyzer is contained within the highlighted red boxes.



Your Report Name

AcId: 482224

Filters

Type	Description	Indexed		
Report Filter	opportunities.forecast_close between 0 Quarters Rounded to +1 Quarters Rounded	Yes		
Report Filter	opportunities.status_type in list Lead, Active, Closed	Yes		
Report Filter	opportunities.strategy_id in list Pipeline Strategy, License Renewal, Outbound Prospecting Strategy, Inbound Prospecting Strategy	Yes		
Report Filter	opportunities.acct_hierarchy in list John Kembel, Joseph Brown, Marcus Bragg, Steve Daines	No	No	
Report Filter	opportunities.c$products_in_deal in list RNT Communities, RNT Communities, CRM	No	No	The filter contains a custom field which does not have an index. To improve the performance of this query, consider adding an index on the custom field.

The filter contains a custom field which does not have an index. To improve the performance of this query, consider adding an index on the custom field.

Tables

Name			
Accounts			
Opportunities			
Organizations			
Opportunities to Contacts			parent table that do not have rows in this table consider adding a not null join field in the parent table.
Opportunities to Contacts	opp2contacts2	Outer	This table is not utilized in the report. Consider removing the table from the data set. If you need to filter out rows in the parent table that do not have rows in this table consider adding a not null filter on the join field in the parent table.

This table is not utilized in the report. Consider removing the table from the data set. If you need to filter out rows in the parent table that do not have rows in this table consider adding a not null filter on the join field in the parent table.

Levels

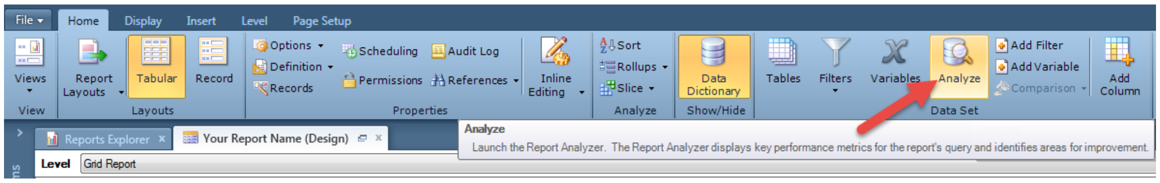
Name	Notes
Pipeline	

Rows

The estimated number of analyzed rows for this query is 2,448.

The estimated number of analyzed rows for this query is 2,448.

To use the Report Analyzer, open a custom report in the report design center, then click the "Analyze" button on the ribbon's Home tab as highlighted in the screenshot below.



Analyze

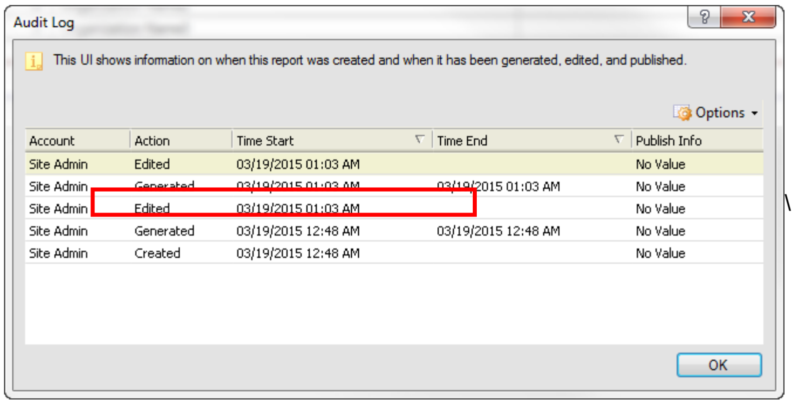
Launch the Report Analyzer. The Report Analyzer displays key performance metrics for the report's query and identifies areas for improvement.

More information about the Report Analyzer is contained in the "Previewing reports with the report analyzer" section of the Oracle Service Cloud Analytics documentation, and in Answer ID #2844 "Using the report analyzer with custom reports" on the Oracle Service Cloud Customer Support site.

Other Tools Available to Identify Tuning Opportunities

Oracle Service Cloud offers other tools to identify long-running queries which can underlie slow running reports.

» The Analytics Audit Log. The Analytics Audit Log is another tool available to assist with report optimization. The Analytics Audit Log can be used to look for query length and other activities pertaining to a report being edited. The audit log is accessed directly from the report design center's Home tab for a particular report (select "Audit Log" button). An example of the audit log is pictured below.


Audit Log

This UI shows information on when this report was created and when it has been generated, edited, and published.

Account	Action	Time Start	Time End	Publish Info
Site Admin	Edited	03/19/2015 01:03 AM		No Value
Site Admin	Generated	03/19/2015 01:03 AM	03/19/2015 01:03 AM	No Value
Site Admin	Edited	03/19/2015 01:03 AM		No Value
Site Admin	Generated	03/19/2015 12:48 AM	03/19/2015 12:48 AM	No Value
Site Admin	Created	03/19/2015 12:48 AM		No Value

» More information about the audit log is contained in the Oracle Service Cloud Analytics documentation. See section "Selecting report options," subsection "Viewing the audit log."

» The AC_Audit_Log Table. Data used to generate the Analytics Audit Log is contained in the AC_Audit_Log table. This table logs the start and stop time of all reports run in a Oracle Service Cloud implementation, so can be used as a powerful tool for locating long-running reports as candidates for tuning opportunities. By building a custom report against the AC_Audit_Log table, and filtering on reports exceeding a certain query length, such reports can be located.

The AC_Audit_Log table is fairly straightforward and contains only a few fields, including the report ID, the type of action taken on the report (Create, Edit, Generate, Publish), who executed the action, when that action started and ended, and what the source of the report request was. An additional important field to look at is the result_code in the ac_audit_log table. This will tell you the type of failure that was returned. The Data Dictionary contains more specific information about the table definition and properties.

» The Report Management component (added in the Nov 13 release) has 2 reports that are if interest when it comes to performance. These are the "Reports with No Filters" and "Deferred Reports" reports. Additional detail of the Report management component can be found in Oracle Service Cloud Documentation.

» There are a number of useful reports in the \Public Reports\Common\Site Administration\Reports directory within the reports explorer. The "Summary of Report Failures" report in this directory is an important tool to find out which reports have failed for the performance reasons described in this document (e.g. Too much data, Time out, etc…). This report will be helpful in tracking down problem reports and the performance issue that was encountered.

8. Use Report Linking Rather Than Drill-Down Levels

For analyses that drill down from one level of analysis to another, use Oracle Service Cloud Analytics' report linking capabilities when doing so improves performance compared to the traditional drill-down level capabilities. Report linking can drill more efficiently when it eliminates unnecessary data and tables from the drill path. Report linking also extends traditional drilling capabilities with more powerful features and improved performance.

Report linking can improve drilling performance in the following manner. The traditional drill down level methodology drills by navigating through one data set common among all drill levels. This can result in a large data set to be drilled through among all levels. In contrast, report linking drills by linking separate individual reports to each other, and executing each in the chain independently. Since each individual linked report has its own result set, the size of the result set can be smaller, because it's limited only to what's needed for that particular report. By drilling on the smaller underlying data sets, performance is improved.

The performance gains achieved with report linking are especially applicable when a table in the drill path is required for only one level, or a subset of drill-levels, rather than all drill levels. For example, a summary report need not include tables that are only used in subsequent detail reports that are linked.

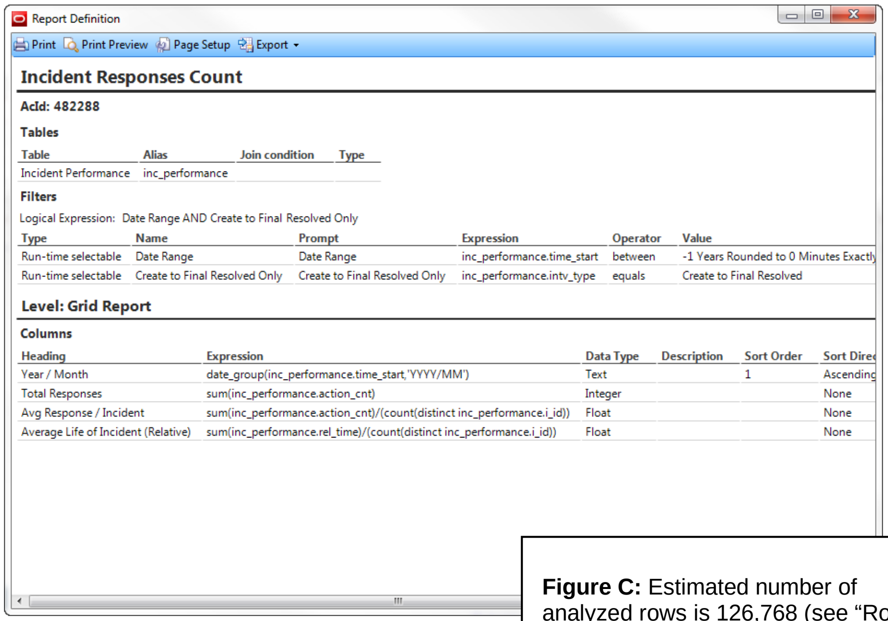
The following callout box illustrates, via the Report Analyzer mentioned previously, how report linking reduces the data set size among report drilling levels and thus improves reporting performance. Other benefits of report linking are mentioned as well.

More information about report linking is found in the "Linking reports" section of the Oracle Service Cloud Analytics documentation, and in Answer ID #2789 "Report Linking in Oracle Service Cloud Analytics" on the Oracle Service Cloud Customer Support site.

EXAMPLE OF IMPROVED PERFORMANCE WITH REPORT LINKING

The performance benefits achieved by using report linking over traditional drill downs is illustrated by the Report Analyzer results listed below for a drill example.

In Figure C, the report is looking at a few statistics from the inc_performance table to generate summary metrics for incident responses[2]. This report on the single inc_performance table is estimated to return 126,788 rows.



Report Definition

Print | Print Preview | Page Setup | Export

Incident Responses Count

AcId: 482288

Tables

Table	Alias	Join condition	Type
Incident Performance	inc_performance		

Filters

Logical Expression: Date Range AND Create to Final Resolved Only

Type	Name	Prompt	Expression	Operator	Value
Run-time selectable	Date Range	Date Range	inc_performance.time_start	between	-1 Years Rounded to 0 Minutes Exactly
Run-time selectable	Create to Final Resolved Only	Create to Final Resolved Only	inc_performance.intv_type	equals	Create to Final Resolved

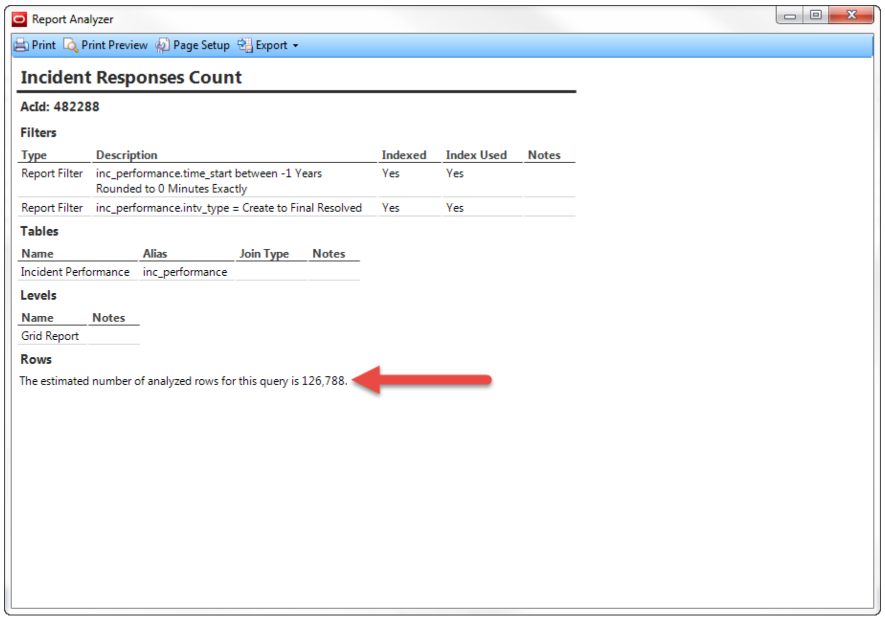
Level: Grid Report

Columns

Heading	Expression	Data Type	Description	Sort Order	Sort Direc
Year / Month	date_group(inc_performance.time_start,'YYYY/MM')	Text		1	Ascending
Total Responses	sum(inc_performance.action_cnt)	Integer			None
Avg Response / Incident	sum(inc_performance.action_cnt)/(count(distinct inc_performance.i_id))	Float			None
Average Life of Incident (Relative)	sum(inc_performance.rel_time)/(count(distinct inc_performance.i_id))	Float			None

Figure C: Estimated number of analyzed rows is 126,768 (see "Rows" section in bottom screen shot).

2 Metrics include the total number of responses sent to incidents created over a given time period, the average number of responses per incident, and the work hour relative length of time that incidents are open (the inc_performance table is required since it is the only place work hour relative time is calculated).



Report Analyzer

Incident Responses Count

AcId: 482288

Filters

Type	Description	Indexed	Index Used	Notes
Report Filter	inc_performance.time_start between -1 Years Rounded to 0 Minutes Exactly	Yes	Yes	
Report Filter	inc_performance.intv_type = Create to Final Resolved	Yes	Yes	

Tables

Name	Alias	Join Type	Notes
Incident Performance	inc_performance		

Levels

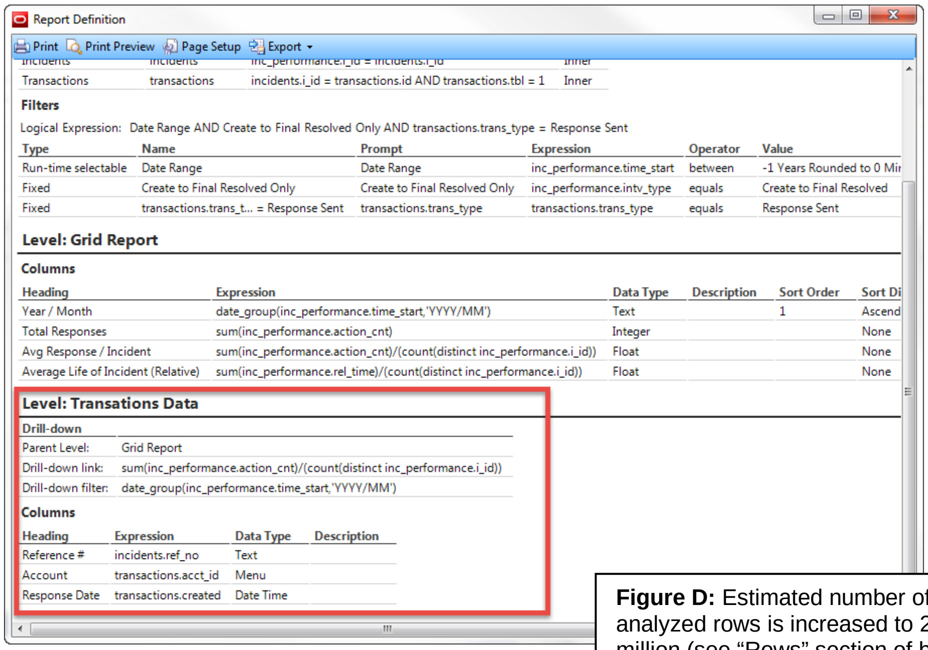
Name	Notes
Grid Report	

Rows

The estimated number of analyzed rows for this query is 126,788.

Figure D then adds a drilldown level to show the specific transactions for each incident response so that anomalies can be investigated. As configured, the number of rows analyzed increases significantly. Also, it becomes necessary to use the sum_distinct function in the top level formulas since each inc_performance record must now be repeated for each response for an incident, so it may appear several times per individual incident.

As you can see, the Analyzer shows that the number of rows analyzed will jump from about 52,200 to over 2.6 million. The Report Analyzer also provides some suggestions that might make the report perform better.



Report Definition

Incidents	incidents	inc_performance.i_id = incidents.i_id	Inner
Transactions	transactions	incidents.i_id = transactions.id AND transactions.tbl = 1	Inner

Filters

Logical Expression: Date Range AND Create to Final Resolved Only AND transactions.trans_type = Response Sent

Type	Name	Prompt	Expression	Operator	Value
Run-time selectable	Date Range	Date Range	inc_performance.time_start	between	-1 Years Rounded to 0 Mi
Fixed	Create to Final Resolved Only	Create to Final Resolved Only	inc_performance.intv_type	equals	Create to Final Resolved
Fixed	transactions.trans_t... = Response Sent	transactions.trans_type	transactions.trans_type	equals	Response Sent

Level: Grid Report

Columns

Heading	Expression	Data Type	Description	Sort Order	Sort Di
Year / Month	date_group(inc_performance.time_start,'YYYY/MM')	Text		1	Ascend
Total Responses	sum(inc_performance.action_cnt)	Integer			None
Avg Response / Incident	sum(inc_performance.action_cnt)/(count(distinct inc_performance.i_id))	Float			None
Average Life of Incident (Relative)	sum(inc_performance.rel_time)/(count(distinct inc_performance.i_id))	Float			None

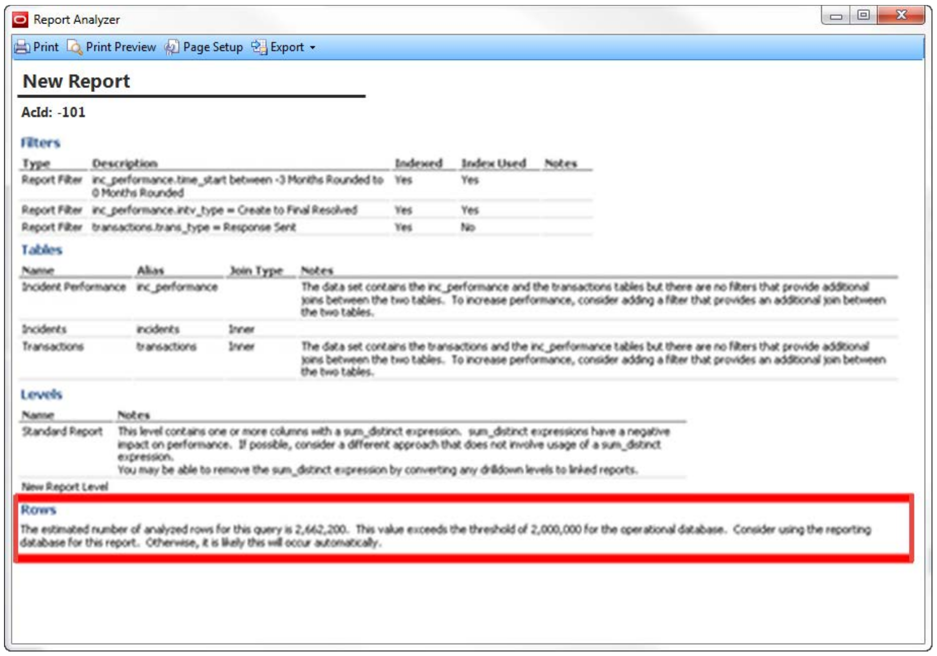
Level: Transations Data

Drill-down	
Parent Level:	Grid Report
Drill-down link:	sum(inc_performance.action_cnt)/(count(distinct inc_performance.i_id))
Drill-down filter:	date_group(inc_performance.time_start,'YYYY/MM')

Columns

Heading	Expression	Data Type	Description
Reference #	incidents.ref_no	Text	
Account	transactions.acct_id	Menu	
Response Date	transactions.created	Date Time	

Figure D: Estimated number of analyzed rows is increased to 2.7 million (see "Rows" section of bottom screen shot).



New Report

AcId: -101

Filters

Type	Description	Indexed	Index Used	Notes
Report Filter	inc_performance.time_start between -3 Months Rounded to 0 Months Rounded	Yes	Yes	
Report Filter	inc_performance.intv_type = Create to Final Resolved	Yes	Yes	
Report Filter	transactions.trans_type = Response Sent	Yes	No	

Tables

Name	Alias	Join Type	Notes
Incident Performance	inc_performance		The data set contains the inc_performance and the transactions tables but there are no filters that provide additional joins between the two tables. To increase performance, consider adding a filter that provides an additional join between the two tables.
Incidents	incidents	Inner	
Transactions	transactions	Inner	The data set contains the transactions and the inc_performance tables but there are no filters that provide additional joins between the two tables. To increase performance, consider adding a filter that provides an additional join between the two tables.

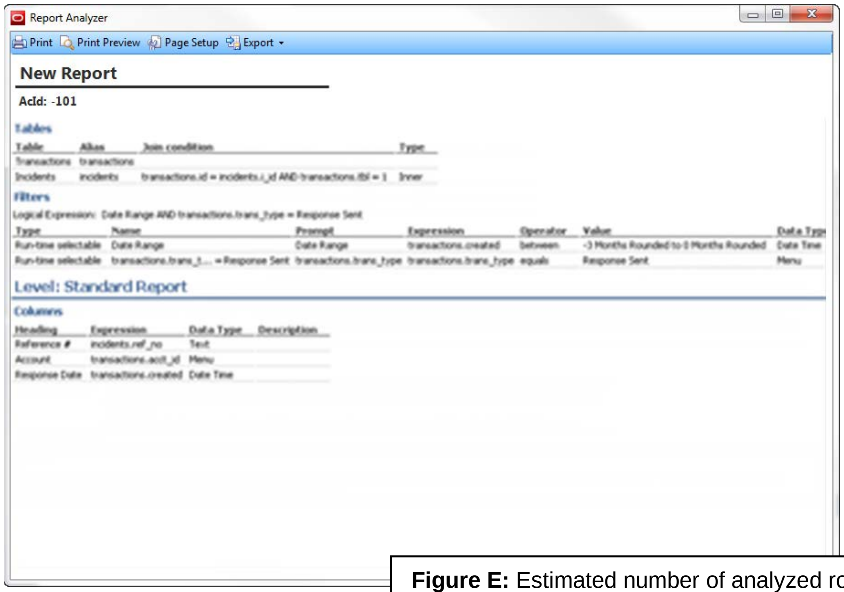
Levels

Name	Notes
Standard Report	This level contains one or more columns with a sum_distinct expression. sum_distinct expressions have a negative impact on performance. If possible, consider a different approach that does not involve usage of a sum_distinct expression. You may be able to remove the sum_distinct expression by converting any drilldown levels to linked reports.
New Report Level	

Rows

The estimated number of analyzed rows for this query is 2,662,200. This value exceeds the threshold of 2,000,000 for the operational database. Consider using the reporting database for this report. Otherwise, it is likely this will occur automatically.

In contrast, Figure E illustrates an alternate report configured with report linking. This report provides the exact same information as the drilldown level in Figure D, but is a stand-alone report that can be linked to from the original inc_performance only report illustrated by Figure C. As you can see, even running over the entire date range, this second level report now examines only about 62,000 rows, rather than the prior 2.7 million rows depicted in Figure D.



New Report

AcId: -101

Tables

Table	Alias	Join condition	Type
Transactions	transactions		
Incidents	incidents	transactions.id = incidents.i_id AND transactions.tbl = 1	Inner

Filters

Logical Expression: Date Range AND transactions.trans_type = Response Sent

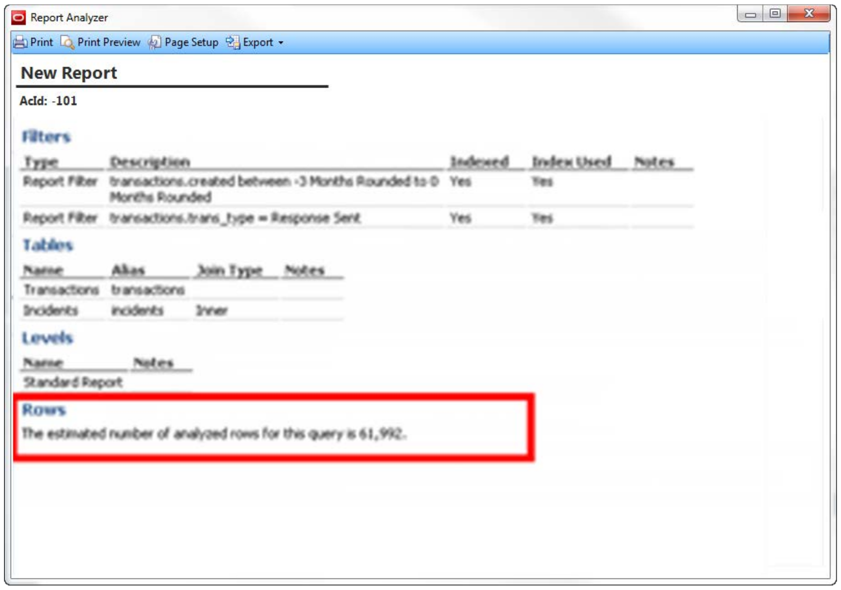
Type	Name	Prompt	Expression	Operator	Value	Data Type
Run-time selectable	Date Range	Date Range	transactions.created	between	-3 Months Rounded to 0 Months Rounded	Date Time
Run-time selectable	transactions.trans_t... = Response Sent	transactions.trans_type	transactions.trans_type	equals	Response Sent	Menu

Level: Standard Report

Columns

Heading	Expression	Data Type	Description
Reference #	incidents.ref_no	Text	
Account	transactions.acct_id	Menu	
Response Date	transactions.created	Date Time	

Figure E: Estimated number of analyzed rows is 61,900 (see "Rows" section of bottom screen shot).



New Report

AcId: -101

Filters

Type	Description	Indexed	Index Used	Notes
Report Filter	transactions.created between -3 Months Rounded to 0 Months Rounded	Yes	Yes	
Report Filter	transactions.trans_type = Response Sent	Yes	Yes	

Tables

Name	Alias	Join Type	Notes
Transactions	transactions		
Incidents	incidents	Inner	

Levels

Name	Notes
Standard Report	

Rows

The estimated number of analyzed rows for this query is 61,992.

In summary, Figure E illustrates that the use of report linking significantly reduces the number of rows analyzed – in this case by 98 % – to get the same information. This reduction in rows will likely increase performance of the report.

Other Benefits of Report Linking

Besides improving performance, report linking offers many other benefits. This includes the ability to re-use reports and dashboards, so that a single report or dashboard can be used as a drilldown for a number of separate reports. In other words, drill-down reports don't need to be created each time from scratch from the parent report. Linking reports is particularly beneficial when you need to create multiple reports with identical drill-down levels, since you can create a single report and link to it instead of re-creating identical levels in multiple reports. This saves the report developer a significant amount of time, and eases report administration.

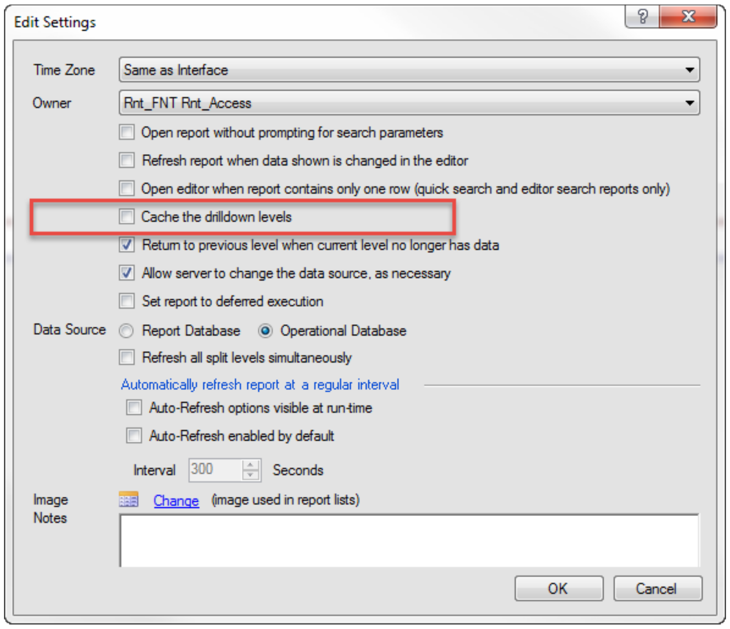
You can also use report linking to drill into disparate data sets, such as drilling from survey questions to different survey response reports. Again, even here, report linking improves performance, because instead of creating a drilldown referencing many tables and expressions that must be used in the drill path, with report linking, each distinct level in the drill path has a corresponding table with only the data needed for that drill level. For example, you could have a separate report for separate question times.

Report linking also allows you to drill into dashboards, such as drilling from an organization record to an organization details dashboard.

Finally, report linking allows you to combine and analyze data which is not otherwise directly related to each other in the Oracle Service Cloud schema. For example, the Stats table and Incidents table are not directly joined, or related, in the Oracle Service Cloud schema. But with report linking, a report on the Stats table can be created, then linked to a report on the Incidents table that displays specific incident details from the same period as the summary numbers recorded in Stats. This is possible because the link can be based on the date in the first report to determine what displays in the second report even though no actual join exists between the two tables at the database level.

9. Use Drilldown Level Caching

For report drilling, use drilldown level caching. This applies to both report linking and the more traditional drilldown capabilities. To use drilldown level caching, select the "Cache the drilldown levels" option from the Edit Settings window, accessed from the report designer's Options > More options menu (see screenshot below). This tells the console to save the levels of a report once they are generated rather than re-generating each level as the user moves through the drilldown levels. This way, the report just pulls the already viewed data back from the cache rather than going back to the server and re-executing the report with the same search criteria. Keep in mind that one can always get a new refreshed copy of the report by pressing the "Refresh" button in the report.



Edit Settings

Time Zone: Same as Interface
Owner: Rnt_FNT Rnt_Access

- [ ] Open report without prompting for search parameters
- [ ] Refresh report when data shown is changed in the editor
- [ ] Open editor when report contains only one row (quick search and editor search reports only)
- [ ] Cache the drilldown levels
- [✓] Return to previous level when current level no longer has data
- [✓] Allow server to change the data source, as necessary
- [ ] Set report to deferred execution

Data Source: Report Database [● Operational Database]

- [ ] Refresh all split levels simultaneously

Automatically refresh report at a regular interval

- [ ] Auto-Refresh options visible at run-time
- [ ] Auto-Refresh enabled by default

Interval 300 Seconds

Image: Change (image used in report lists)
Notes

10. Leverage Cached Data for Historical Insights

To provide users easy access to key historical insights on Oracle Service Cloud data, Oracle Service Cloud offers special tables containing *cached data*[3] to report from. Cached tables provide an easy way to gain deep insights on certain historical trends, activity and metrics, in a highly performant manner, that otherwise would be difficult to achieve. For historical reporting, use tables containing cached data whenever possible.

The table below lists database tables in the Oracle Service Cloud schema with cached data, the type of data each contains, and a relevant example. You can develop custom reports referencing these tables. Many standard reports reference these tables also. You can see which tables a standard report references by looking at the report definition.

Tables in Oracle Service Cloud Schema Using Cached Data

Table	Data	Example
Incident Performance	Incident intervals	Time from incident creation to first

3 Report caching is a process where each day, Oracle Service Cloud runs an "Age Database" utility that takes the detailed data, aggregates it, summarizes it, and transforms it into a meaningful pre-computed metrics immediately available from the cached tables for reporting. Caching occurs daily, so information in the cached tables is always current through midnight of the previous day.

(inc_performance)		agent response
Session Summary (session_summary)	End-user session data	# sessions ending on the answer display page
Stats (stats)	End-user and Admin summary data	Incident backlog at a given time
Opportunity Performance (opp_performance)	Opportunity strategy and stage tracking	Amount of time opportunities are spending in various stages
Opportunity Snapshots (opp_snapshots)	Historical opportunity data	What a given opportunity looked like at specific points in the past
Revenue Snapshots (revenue_snapshots)	Opportunity revenue summary data	# opportunities closed and total value for a given timeframe
Pipeline Snapshots (pipeline_snapshots)	Opportunity pipeline summary data	Total weighted value of opportunities in the pipeline over time

More information about cached data is available in the “Using cached data” section of the Oracle Service Cloud Analytics documentation, and from Answer ID #1917, “Cached data used in reports,” on the Oracle Service Cloud Customer Support site. Detailed information about specific tables is available in the Oracle Service Cloud data dictionary.

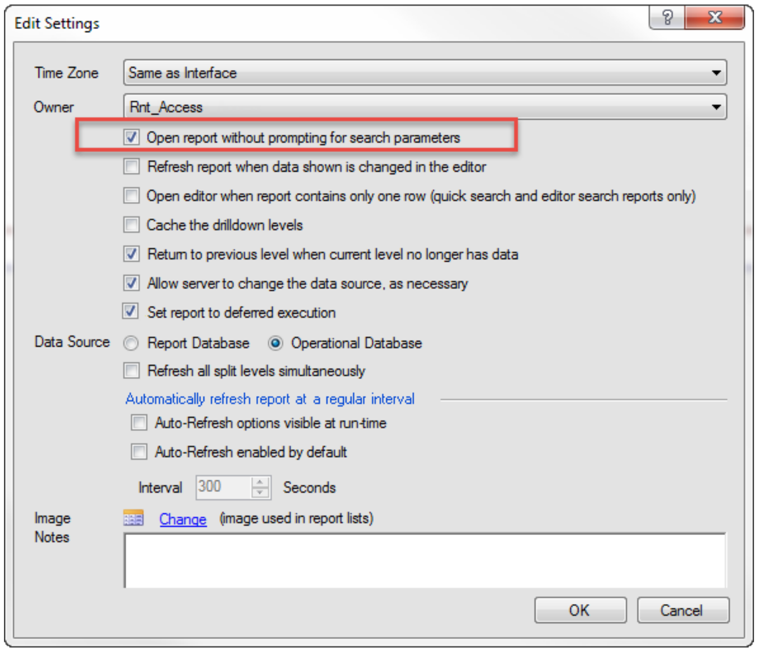
11. Use Initial Run for Small Data Volumes Only

Reports can be configured to make an initial run without prompting the user for search parameters. This is referred to as “initial run” or “initial search.” Allow this only when you know that a small volume of data will be returned. Depending on the default settings for the fixed and run-time filters, the initial search may result in querying an entire database table which may include a significantly large number of records, and therefore impact performance.

For example, if the initial search is enabled for a report on the Contacts table, while there are no fixed filters and none of the run-time filters have default values pre-selected, each time the report is selected, the query will run through every contact record in your database. This may be in the hundreds of thousands of records, or even millions of records. This can significantly contribute to a slow query error on your site, and a slow running report.

The initial search capability should be disabled for reports that have no filters or where the default value of the filters is not restricted. That way, the user can set the filters when they perform their first search.

The initial search is set from the Properties > Report Options panel by enabling or disabling the “Open report without prompting for search parameters” check box. See screenshot below.



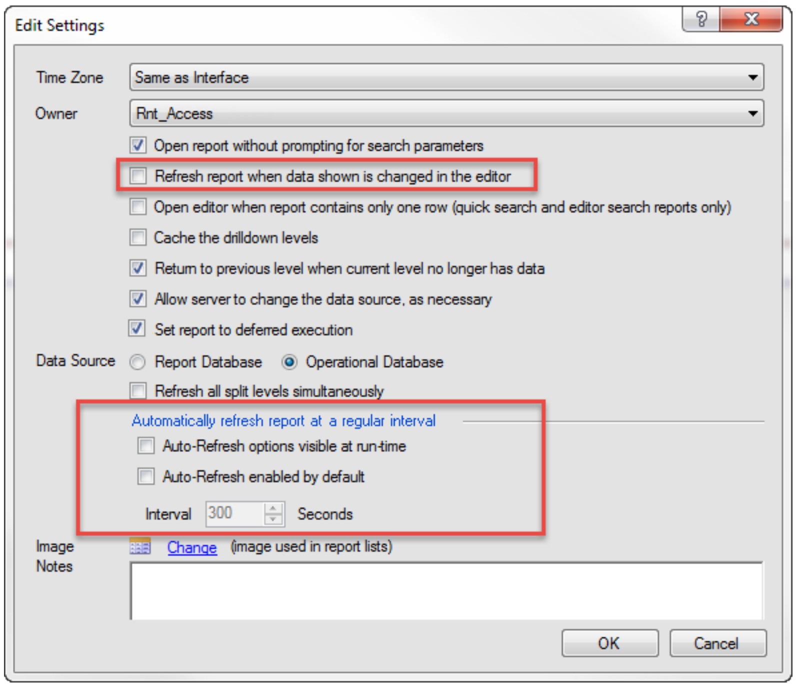
Time Zone: Same as Interface
Owner: Rnt_Access
[✓] Open report without prompting for search parameters
[ ] Refresh report when data shown is changed in the editor
[ ] Open editor when report contains only one row (quick search and editor search reports only)
[ ] Cache the drilldown levels
[✓] Return to previous level when current level no longer has data
[✓] Allow server to change the data source, as necessary
[✓] Set report to deferred execution
Data Source: Report Database [✓ Operational Database]
[ ] Refresh all split levels simultaneously
Automatically refresh report at a regular interval
[ ] Auto-Refresh options visible at run-time
[ ] Auto-Refresh enabled by default
Interval 300 Seconds
Image: Change (image used in report lists)
Notes

12. Use Auto Refresh for Small Data Volumes Only

Reports should only use auto-refresh features if there are adequate fixed or run-time filters in place to return a limited number of records at report run-time each time it is refreshed.

When the “Refresh report when data is changed due to action initiated from this report” checkbox is selected (see screenshot below), each time a record from that report or view is saved, another query is made to the database to refresh the report or view grid. As with the initial search option, if adequate search filters are not in use, a large number of records may be analyzed which impacts the speed of the query that is performed.

When either of the "Automatically refresh report at a regular interval" checkboxes are selected (see screenshot below), then the report is automatically refreshed based on a timer. Again, the performance of the refresh may suffer if adequate filtering isn't specified.


Edit Settings
Time Zone
Same as Interface
Owner
Rnt_Access
☑ Open report without prompting for search parameters
☐ Refresh report when data shown is changed in the editor
☐ Open editor when report contains only one row (quick search and editor search reports only)
☐ Cache the drilldown levels
☑ Return to previous level when current level no longer has data
☑ Allow server to change the data source, as necessary
☑ Set report to deferred execution
Data Source
○ Report Database ● Operational Database
☐ Refresh all split levels simultaneously
Automatically refresh report at a regular interval
☐ Auto-Refresh options visible at run-time
☐ Auto-Refresh enabled by default
Interval 300 Seconds
Image
Change (image used in report lists)
Notes
OK
Cancel

The auto-refresh features are enabled or disabled from Properties tab > Report Options panel. See screenshot above.

More information about auto refresh features is available in the Oracle Service Cloud Analytics documentation.

13. <u>Publish or Schedule Large Reports</u>

Report publishing and report scheduling provide other means of running larger reports in a manner that best preserves system resources while providing your reporting audience the insights they need.

a. Use report publishing for large reports targeted to broad audiences. A published report is one that is run, saved in a static state, and then made available for offline viewing. In other words, the report is run once, the output is saved off-line, and then a broad base of users can have access to the saved version. Publishing reports is an ideal way to make larger or processing-intensive reports available to a large audience while preserving system resources so that the report isn't rerun each time it needs to be viewed.

   Because the published report is in a static state, the report administrator must decide how often to refresh the report data by rerunning and republishing the report. This will depend on how rapidly the data changes, and how important data currency is to the users.

   Report administrators can also leverage the following report publishing capabilities:

   - Between the time a report is run and then eventually published, the report output and display options can be modified so the audience is presented with the report as the developers wants them to see it
   - Reports publishing can be a scheduled function achieved via report scheduling (see next section)

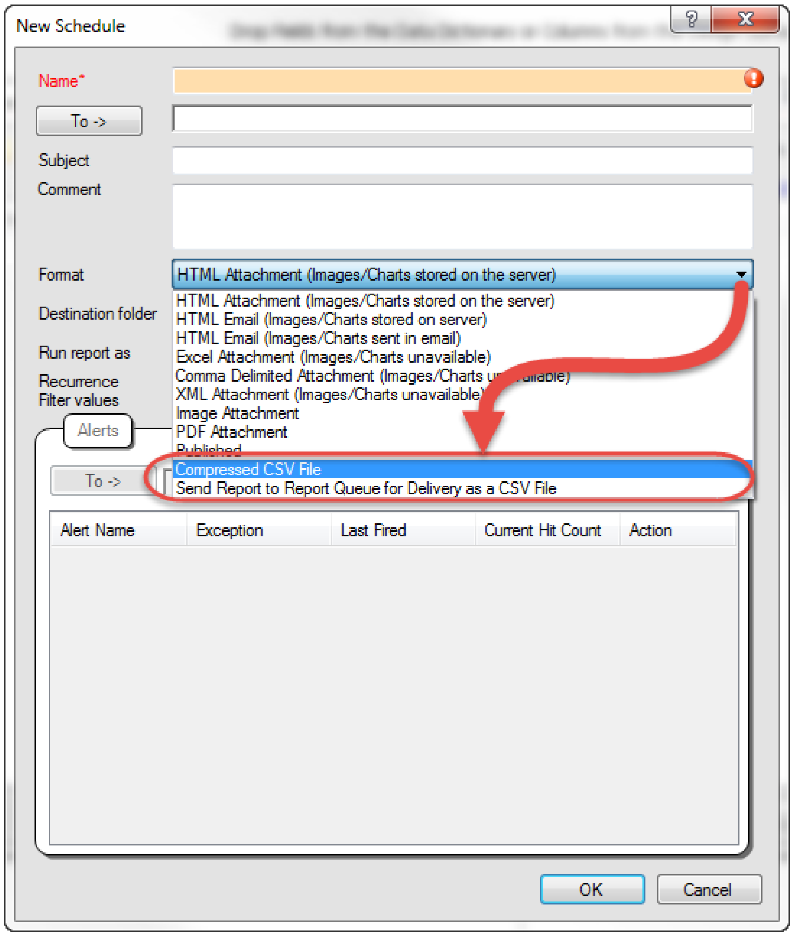
   See the "Publishing reports" section of the Oracle Service Cloud Analytics documentation for more information about report publishing.

b. Use report scheduling to process large reports during off-hours. Report scheduling lets you specify a report to run at a scheduled time in the future. Once run at the scheduled time, the report is sent to the intended audience via email, or is published.

Beginning in the Feb 14 version of Oracle Service Cloud, two new report schedule delivery options were added that can be used to deliver large data sets from Oracle Service Cloud reports. This option allows up to 1 million rows of data (or 100Mb, whichever comes first) to be sent via compressed CSV file to the recipients. This can be done in one of two ways -- "Compressed CSV File" and "Send Report to Report Queue for Delivery as a CSV File".

When the user chooses the "Compressed CSV File" option the scheduled report will be delivered as a compressed CSV file attached to an e-mail for the designated recipients. This is similar to the existing export formats. If sending a compressed CSV file to an email address, the mail server file size delivery limits still exist. These limitations are not controlled within the Oracle Service Cloud product, but rather on each individual mail server.

When the user chooses the “Send Report to Report Queue for Delivery as a CSV File” option the report output can be retrieved from the “My Queued Reports” report in a CSV format after the designated schedule time. This 2nd option will allow users to work around any e-mail file delivery limits by effectively allowing them to download the file created from the My Queued Reports report (Public Reports > Common > Site Administration > Reports > My Queued Reports).


New Schedule
Name*
To ->
Subject
Comment
Format
HTML Attachment (Images/Charts stored on the server)
Destination folder
Run report as
Recurrence
Filter values
Alerts
To ->
HTML Attachment (Images/Charts stored on the server)
HTML Email (Images/Charts stored on server)
HTML Email (Images/Charts sent in email)
Excel Attachment (Images/Charts unavailable)
Comma Delimited Attachment (Images/Charts unavailable)
XML Attachment (Images/Charts unavailable)
Image Attachment
PDF Attachment
Published
Compressed CSV File
Send Report to Report Queue for Delivery as a CSV File
Alert Name
Exception
Last Fired
Current Hit Count
Action
OK
Cancel

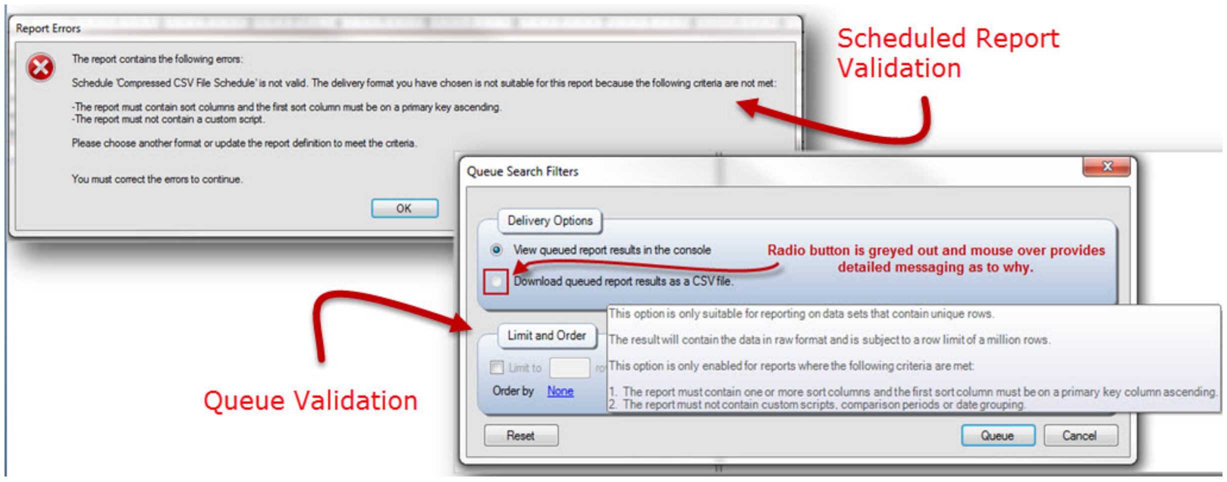
Report scheduling is an ideal way to run large reports during non-peak hours, when more processing capacity is available, and where there will be less impact on the live operational database.

More information about report scheduling is available in the “Scheduled reports” section of the Oracle Service Cloud Analytics documentation.

CSV Export Requirements

The same CSV export requirements exist for reports have a scheduled CSV option selected. There is validation for these requirements to ensure that these exist in the report. If the report fails any of these requirement checks the user is prevented from being able to save the report schedule with the CSV format options selected.

The following are the validation windows that appear in if these checks fail:



The report contains the following errors:

Schedule 'Compressed CSV File Schedule' is not valid. The delivery format you have chosen is not suitable for this report because the following criteria are not met:

-The report must contain sort columns and the first sort column must be on a primary key ascending.
-The report must not contain a custom script.

Please choose another format or update the report definition to meet the criteria.

You must correct the errors to continue.

[✓ View queued report results in the console]
Download queued report results as a CSV file.

Radio button is greyed out and mouse over provides detailed messaging as to why.

This option is only suitable for reporting on data sets that contain unique rows.

The result will contain the data in raw format and is subject to a row limit of a million rows.

This option is only enabled for reports where the following criteria are met:

1. The report must contain one or more sort columns and the first sort column must be on a primary key column ascending.
2. The report must not contain custom scripts, comparison periods or date grouping.

For more information on the CSV export functionality, please see the “Compressed CSV Report Exports” section below.

14. <u>Archive Old Incident Data From Your Operational Database</u>

Removing incident data from your production system (i.e. the Operational Database) will make reports, analytics and dashboards that access incident data run faster[4]. You can remove old, obsolete incident data[5] from your production system by using Oracle Service Cloud’s incident archiving capabilities. Afterwards, your reports should run faster because there is less data in the system.

Incident archiving permanently removes closed incident records from your Operational Database that meet certain criteria. Specifically, archived incidents are those that closed out a certain number of days ago, where this number of days is configurable. Once archived, key data about these incidents is retained offline in searchable XML files. Staff members can access archived incidents from the Oracle Service Cloud Console once the Archived Incidents component has been added to their navigation set.

For more information about incident archiving, see the “Accessing archived incidents” section of the Oracle Service Cloud Service User Manual, and Answer ID #1234 (“Archiving older incidents without deleting them from our site”) on the Oracle Service Cloud Customer Support site.

Reporting Implications and Best Practices

While incident archiving will help improve the performance of your production system, and related reporting performance, once an incident is archived, it’s no longer available for reporting in Oracle Service Cloud Analytics, because it no longer resides in either

4 Incident archiving offers other advantages also, such as faster searches, faster overall system performance, faster upgrades, and a faster time to clone sites.

5 Currently, archiving is supported for incident records only. Records archived include incidents, incident threads and incident transactions.

the Operational Database or Report Database. While each individual archived record stored offline can be individually viewed via the Archive Console in Oracle Service Cloud, archived records cannot be commingled together and reported on together (e.g. to provide trending reports) within Oracle Service Cloud Analytics.

15. Use of Data Export Tools

To perform a large data extraction of Oracle Service Cloud data, use alternative exporting solutions rather than Oracle Service Cloud Analytics when the required extraction exceeds Oracle Service Cloud Analytics thresholds. Example alternative solutions include the Oracle Service Cloud Data Export Service, the Data Direct ODBC/JDBC drivers, or Oracle Service Cloud Connect tools. More information about these alternatives is available in the following section "Using Data Warehouses and Other Specialized Analytics Tools," subsection "Exporting Oracle Service Cloud Data".

Explanation

Oracle Service Cloud Analytics is a business analytics and reporting tool, and optimized to deliver that capability. In particular, it provides actionable reporting and analyses on your Oracle Service Cloud data. This typically involves presenting summarized, actionable insights with drill-downs or report links for more granular analyses.

While Oracle Service Cloud Analytics can be used to obtain a detailed listing of transactions and database tables (i.e. a "data extraction") available for extraction from the system, this isn't the primary purpose of Oracle Service Cloud Analytics. Therefore Oracle Service Cloud Analytics isn't optimized to provide this, and applies thresholds to the number of report output rows that can be returned, viewed or exported. These thresholds (previously discussed in the "Reporting Thresholds and Governors" section of this Oracle Service Cloud Guide) help Oracle Service Cloud Analytics perform its primary functions optimally.

Using Data Warehouses and Other Specialized Analytics Tools

Sometimes reports are so large, or require so much processing, that they are best delivered on platforms specifically designed and purpose-built for highly intensive analytics. These systems often use a data warehouse, OLAP (on-line analytical processing) technologies, and other advanced capabilities. If you're looking for improved performance and scalability for your Oracle Service Cloud reports beyond that provided by applying the best practices outlined in this Guide, consider using a data warehouse for this reporting purpose. A data warehouse can also be used to combine Oracle Service Cloud data with data from other corporate systems into a central data repository maintained in a way that delivers high-performance analyses to broad audiences.

Oracle is currently working on an OTBI Enterprise for CRM Cloud Service adapter to help pull Oracle Service Cloud data into on premise or cloud (OTBI Enterprise) Oracle BI systems. This will come with prebuilt dashboards that can be used to report on Service Cloud data in very powerful Oracle BI data warehouse environments. This adapter will be available in a future release. Please note that not all data from Oracle Service Cloud will be available through the adapter into the OTBI Enterprise for CRM Cloud Service.

Exporting Oracle Service Cloud Data

In some scenarios, Oracle Service Cloud data may need to be exported to a customer data center for use in other applications. Several options are available to achieve this depending on your needs and data volumes. Contact your Oracle Service Cloud account team for more information about any of these options.

» Oracle Service Cloud ODBC/JDBC Export Access provided via our partner Data Direct. The Oracle Service Cloud ODBC/JDBC connector allows you to pull data out of your Oracle Service Cloud site using Service Cloud API functionality.

» Oracle Service Cloud Connect Tools. Oracle Service Cloud Connect provides integration capabilities between Oracle Service Cloud and other applications via an application programming interface (API). This capability can be leveraged to export data from Oracle Service Cloud to be used in other data sources. There are a number of Oracle Service Cloud Connect tools that provide these capabilities including but not limited to Oracle Service Cloud Connect Data Integration, and Oracle Service Cloud Connect Web Services for SOAP, and Oracle Service Cloud Connect PHP API. More details on the connect tools can be found in Answer #5169: "Technical Documentation and Sample Code". NOTE: Using these tools for bulk exports of large data volumes may require special considerations depending on the need.

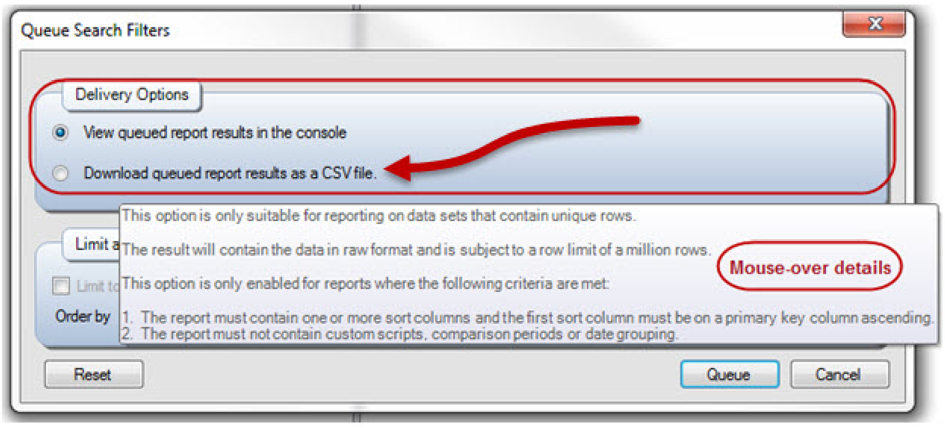
» Oracle Service Cloud Analytics. Output from an Oracle Service Cloud Analytics report can be exported in any of the following formats: Excel, XML, Delimited, HTML, PDF, Image and CSV. Up to 100,000 rows can be exported at a time (see "Row Export Threshold" paragraph in prior section "Reporting Thresholds and Governors" for additional information). As of February 2014, an additional functionality was added that allows the export of up to 1,000,000 rows of data or 100 Mb (whichever limit is reached first). This functionality is described in the following "Compressed CSV Report Exports" section. If an export of more than 1,000,000 rows is required, one of the alternative extract methods listed above should be used, or you can run multiple instances of the report with appropriate filters to produce output in 1,000,000 row segments. For more information, see the "Exporting report output" section of the Oracle Service Cloud Analytics documentation.

  Note: While you can export up to 100,000 rows to a file, the maximum number of rows you can display online for Oracle Service Cloud Analytics report output is 10,000 rows per page by default. This setting can be increased, however. See prior "Report Row Viewing Threshold" section for information about changing this via the VRL_HARD setting.

Compressed CSV Report Exports

Beginning in the February 2014 version of Oracle Service Cloud, a new report delivery option was added that can be used to export data out of the Oracle Service Cloud. This includes the ability to export up to 1 million rows of data (or 100Mb, whichever comes first). This can be done in one of two ways:

Queued CSV Export
Queuing the report in the report editor (or via the queue option if the report is large).
This option allows you to select the report in the reports explorer and select the queue button from the ribbon. You will then be prompted to select a delivery option. The download compressed CSV file provides you with the ability to report on the expanded data set as a single file output.



This option is only suitable for reporting on data sets that contain unique rows.
The result will contain the data in raw format and is subject to a row limit of a million rows.
This option is only enabled for reports where the following criteria are met:
1. The report must contain one or more sort columns and the first sort column must be on a primary key column ascending.
2. The report must not contain custom scripts, comparison periods or date grouping.

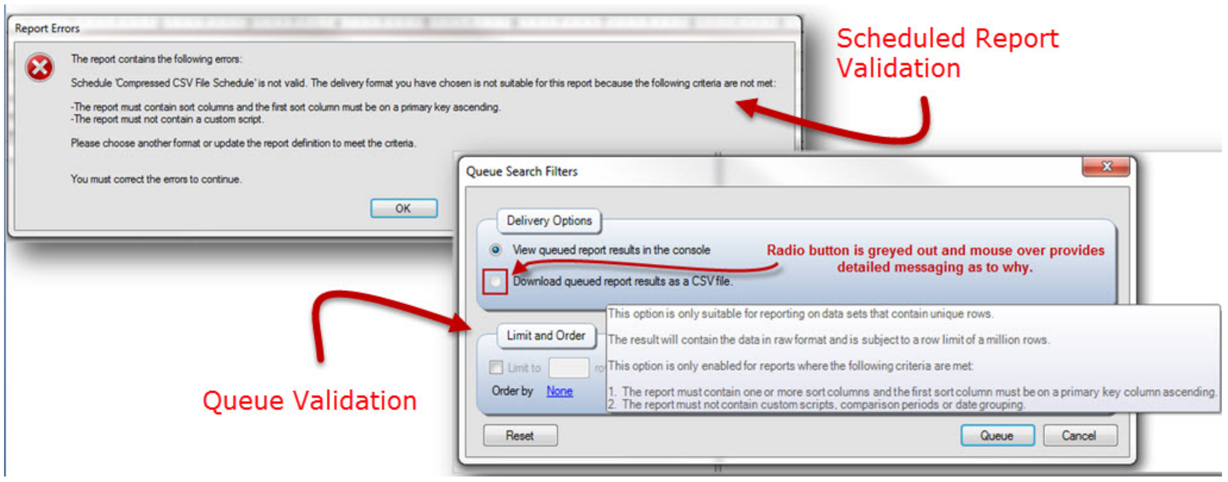
Scheduled reports
This option is described in the section "Publish or Schedule Large Reports" and allows you to have a report output delivered on a user specified schedule using the scheduled report editor. The schedule editor contains two additional format options—"Compressed CSV File" and "Send Report to Report Queue for Delivery as a CSV File".

CSV Export Requirements
There are a few required items for the CSV formatting to be available to the user. Each of these requirements must be met or the user will not be able to save a schedule in the report scheduler and/or queue the report itself from the reports explorer:

- The report **must** contain sorted columns
- The first column that is sorted **must** be a primary key (e.g. incidents.i_id or contacts.c_id) that is sorted **ascending**.
- The report **cannot** contain any custom scripts.
- The report **cannot** contain comparison columns.

- The report **cannot** contain date grouping columns.
- The CSV delivery options are only suitable for reports that return unique rows.

There is validation for the above items in the system. If the report fails any of these requirement checks the user is prevented from being able to use the CSV format options. The following are the validation windows that appear in if these checks fail:



The report contains the following errors:

Schedule 'Compressed CSV File Schedule' is not valid. The delivery format you have chosen is not suitable for this report because the following criteria are not met:

-The report must contain sort columns and the first sort column must be on a primary key ascending.
-The report must not contain a custom script.

Please choose another format or update the report definition to meet the criteria.

You must correct the errors to continue.

[● View queued report results in the console]
Download queued report results as a CSV file.

This option is only suitable for reporting on data sets that contain unique rows.

The result will contain the data in raw format and is subject to a row limit of a million rows.

This option is only enabled for reports where the following criteria are met:

1. The report must contain one or more sort columns and the first sort column must be on a primary key column ascending.
2. The report must not contain custom scripts, comparison periods or date grouping.

Please note that the required elements above will ensure that you can use this CSV export functionality.

Other Important Reporting Best Practices

This section lists other best practices that can help improve the performance and scalability of Oracle Service Cloud Analytics reports.

1. Include Only Pertinent Functionality within Your Report

   Eliminate any redundant or unnecessary fields, data, columns, filters, table instances, joins, expressions, etc. from your report. Including superfluous elements in a report can make the report run longer.

2. Ensure Design Integrity is Intact

   Proper use of filters, table joins (inner vs. outer), etc. is critical in determining what results your report will show and how it will perform. The more familiar a report writer is with SQL and relational databases the more intuitive report integrity can be. There are many resources available on the Internet to learn SQL and relational database fundamentals.

3. Display a Summarized Analysis First, Then Drill To Details

   Make reports more meaningful and performant by first showing high-level trends or summarizations, and then encouraging the user to drill to the details of interest via report linking. Starting with summarized information improves performance because less data is initially processed; drilling to details of interest also improves performance because only the data drilled to is processed when needed. Avoid reports that provide large volumes of transactional detail when possible. They typically are less meaningful to end-users, are more difficult to interpret, and can be less performant.

4. Break Large Reports into Smaller Reports

   Rather than having one large complex report, break it up into a series of smaller reports, and display them in a dashboard.

5. Design Reports with the Audience In Mind

   Remember that reporting audiences have different roles in an organization, and therefore different reporting needs. Keep this in mind when creating reports for different audiences, and design the report accordingly. For example, executives may be looking more for high-level trending information, with an option to drill into details, whereas a manager may need a more detailed analysis for his or her team immediately.

6. Train Your Staff to Use Filters Effectively

   Ensure that staff members understand the importance of filtering what they output in a report. A report can perform slowly if a staff member performs a minimally-filtered search that returns a large amount of data. For example, if the staff member sets the run-time filters to search all available options and doesn't search by specific search text, the query will still be evaluating potentially thousands or millions of rows that are not necessarily of interest. In some cases, access to reports should be restricted if the reports have potential to perform poorly unless used and filtered properly.

7. Use Standard Reports as a Starting Template for Custom Reports

   When possible, build a custom report starting with a similar standard report. Not only will this save time, but standard reports have typically been tuned for performance already. Be sure however to eliminate any reporting elements contained in the original report that the newer report doesn't need.

8. Tune Reports with Live Data

   To best ensure a report will perform well in a live production environment, use live data if possible in the report tuning activity. Live data from your operational database will provide the data shape, data volume and other data characteristics needed to ensure your report performs well under live, production circumstances.

9. Incorporate Report Tuning Into the Project Development Lifecycle

   A report should be tuned before it goes live. Factor report tuning into the development phase of your report deployment. As mentioned above, when possible, use live operational data to test and tune the report, or, use a test data set mirroring what is anticipated to be the live data shape and volume.

10. Examine Custom Scripts for Performance Impact

    Custom scripting is a powerful feature in Oracle Service Cloud Analytics. However, the flexibility custom scripting provides can also introduce processing that Oracle Service Cloud can't foresee. When using custom scripting, consider the impact the script will have on reporting performance.

11. Become Familiar with Relational Database Concepts and the Oracle Service Cloud Schema

    You will have a significant advantage in understanding Oracle Service Cloud Analytics and how the data is structured if you are familiar with relational database concepts. Please refer to the Oracle Service Cloud Analytics documentation, or Answer ID# 1350 "Accessing Oracle Service Cloud manuals, release notes, and upgrade documentation" on the Oracle Service Cloud Customer Support site for more information about this.

Conclusions / Summary

This Oracle Service Cloud Guide outlined best practices to help your Oracle Service Cloud Analytics reports run faster and scale as your Oracle Service Cloud deployment scales. These best practices can be summarized as follows:

DO:

- Report from the Report Database
- Permit report queuing
- Understand reporting thresholds and how they impact report execution
- Filter reports to narrow analyses as much as possible
- Configure filters properly to optimize performance
- Configure table joins to return the smallest data set possible
- Use the Report Analyzer to tune reports
- Use report linking
- Use drill-down level caching
- Leveraged cached data for historical insights
- Use report publishing and report scheduling
- Archive old data from your production database (but know how you will report on it first)

AVOID:

- Filtering on fields that aren't indexed
- Filtering on functions
- Using Oracle Service Cloud Analytics for data dumps; use export tools instead
- Using inefficient functions such as sum_distinct
- Using drill-downs when report linking will provide a better alternative
- Increasing threshold governors unless absolutely necessary. Consider the impact of doing so on your overall system, including on your live production system.
- Using initial search and auto refresh functions unless small data volumes will be returned
- Using unnecessary objects or functionality in reports

For More Help

If you still experience reporting performance or scalability challenges after applying the best practices outlined in this Oracle Service Cloud Guide, or if you need additional assistance with these best practices, contact Oracle Service Cloud Customer Care for recommendations on potential next steps.

Additionally, you may find other resources Oracle Service Cloud offers about its business intelligences solutions helpful. These resources are outlined in the following section.

Other Resources

The following additional resources are available for help in developing, tuning and using analyses and reports within Oracle Service Cloud:

- **Oracle Service Cloud Analytics documentation**
  - Report Management (Nov 2013 release and later) > Definitions > View Descriptions

- **Oracle Service Cloud Analytics Product Online Help** (see Help link in Service Cloud console)

- **Reporting & Analysis Forum on the Oracle Service Cloud Customer Community**

- ***Analytics Best Practices* document**
  - See Answer ID #2053, "Best Practices for Analytics", on Oracle Service Cloud Customer Support site

- **User Groups on Oracle Service Cloud Customer Community**

- **Answers on Oracle Service Cloud Customer Support site**. Particularly relevant are:
  - Answer ID #2380 "Improving performance of reports and console view"
  - Answer ID #2817 "Reporting data seems incorrect or behind"
  - Answer ID #2776 "Common questions regarding queued reports"
  - Answer ID #2149 "Receiving a message "Query processes too much data"
  - Answer ID #2223 "Analytics Error: Data Set has exceeded maximum size"
  - Answer ID #1266 "Impact of indexing a custom field"
  - Answer ID #1839 "Types of table joins used with reports and views"
  - Answer ID #2844 "Using the report analyzer with custom reports"
  - Answer ID #2789 "Report linking in Analytics"
  - Answer ID #1917 "Cached data used in reports"
  - Answer ID #1234 "Archiving older incidents without deleting them from our site"
  - Answer ID #265 "Specifying how long solved incidents remain in the system"
  - Answer ID # 781 "Purging the database of old incidents"
  - Answer ID #1936 "Implementing ODBC with our site"

- **Virtual Ask-the-Experts Webcasts for Oracle Service Cloud**
  - See http://bit.ly/OSVCexperts section of RightNow Customer Community

- **Online Training** (see Education & Services section of RightNow Customer Community)
  - Product Tutorials

- **In-Person Training** (via Oracle University)
  - Regional and Virtual Oracle Service Cloud Analytics training classes
  - **Oracle Consulting Services**
    - Custom training/Custom engagements

ABOUT THE AUTHORS

Rob Nash, Professional Services Consultant/Quality Assurance Engineer

Rob joined Oracle Service Cloud (formerly RightNow Technologies) in July 2005. From that time through 2010, he spent his time as a Professional Services Consultant primarily focusing on Analytics. He has worked directly with over 100 customers across all industries delivering custom Analytics reports and custom training. He is experienced in all facets of the full RightNow product suite with special emphasis on Analytics. In 2010, Rob migrated to the Development organization to employ his skills and knowledge of the product as a Quality Assurance Engineer. He is currently part of the Analytics scrum team, where he functions as Scrum Master in addition to his duties providing quality assurance for all new feature development from this team. Prior to RightNow, Rob spent six years in as a Consultant with Harbor Technology Group. He specialized in implementing and customizing Human Resources software. Rob was a communications officer in the USAF for six years prior to that. Rob graduated from Texas Tech University with a BS degree in Computer Science, and Oklahoma City University with an MBA.

Greg Rice, Director of Product Management, Oracle Business Intelligence Applications for CRM

Greg worked in RightNow's Engage Center of Excellence from 2008 - 2011 as Director of Product Management, holding responsibilities for product management, product marketing, strategy and the overall success of RightNow's analytics and reporting solutions. Greg joined RightNow in 2006 and lead other product management and product marketing initiatives during this time. Prior to RightNow, Greg held product leadership roles for Siebel System's analytics product line, and conducted CRM business strategy consulting engagements for key Siebel customers. Greg is currently responsible for Oracle business intelligence applications for CRM solutions. Greg holds a BS in Business Administration from California State University, Sacramento, and an MBA from The Wharton School at University of Pennsylvania.

Kenny Tietz, Principal Product Manager, Oracle Service Cloud Analytics

Kenny joined Oracle Service Cloud (formerly RightNow Technologies) in 2006 and is currently responsible for product management, strategy, and overall success of the Oracle Service Cloud Analytics platform. Kenny also is the product manager of the Management and Configuration pages along with Outlook Integration. Before transition into Product Management in Oct of 2013, Mr Tietz spent 8 years in the Customer Care support organization in deep technical support, leadership, and operations roles. This helped to build a deep technical knowledge of many areas of the Oracle Service Cloud product and the way our customers utilize the solution. Kenny holds a BS in Chemical Engineering from Colorado State University (Go Rams!). When away from work Kenny enjoys hiking, biking, skiing, fishing, and most of all spending quality time with his family.

White Paper Title
January 2015
Author: Nick Lenzmeier, Greg Rice, Rob Nash, Kenny Tietz
Contributing Authors: Melinda Uhland