Performance Tuning Guidelines for Siebel CRM Applications on Oracle Database

An Oracle Technical White Paper

December 2010

Authors: James Qiu, Paul Blokhin, Mehdi Gerami

Overview

The purpose of this document is to clarify the formal Oracle Database configuration settings for Siebel CRM applications. Its primary goal is to provide the required settings relevant to the Cost-Based Optimizer in Oracle Database 10g and 11g where it is the database platform for Siebel CRM applications.

The guidelines in this document take precedence over all other documents on this subject.

Siebel CRM performance tuning is a complex topic. Although there is no "one size fits all" solution, following the guidelines in this document will establish a solid foundation and a healthy baseline for further performance tuning.

The required settings in this document provide these guidelines for the key area of SQL optimization, which affects performance of Siebel CRM application deployed on the Oracle 10g and 11g database platforms.

The guidelines are divided into two major sections:

- Collecting Cost-Based Optimizer (CBO) Statistics
- Setting CBO-related Oracle Parameters

NOTE: Siebel CRM customers are strongly recommended to follow the guidelines in this document. Failure to do so may limit Oracle's ability to support your Siebel implementation.

This document should be used in conjunction with the *Siebel Installation Guide* for your operating system, located in the Siebel Bookshelf. The "Guidelines for Configuring an Oracle Database" section provides other important required settings for Oracle 10g and 11g for Siebel CRM applications.

Additionally, a well-tuned out-of-the-box Siebel CRM configuration may need further tuning for most implementations, given the different data volume, the data shape, the transaction mix, customizations, and so on. We highly recommend following the guidelines in this document, the ones in "Tuning Customer Configurations for Performance," in the *Siebel Performance Tuning Guide* in the Siebel Bookshelf and "Oracle Database Performance Tuning Guide" in Oracle Database Reference Documentation

Collecting Cost-Based Optimizer (CBO) Statistics

Oracle CBO bases the decisions it makes on data statistics. Proper collection of optimizer statistics for Siebel CRM data tables, including indexed columns, indexes, and histograms for all indexed columns, is key in generating optimal query execution plans.

It is recommended to use PL/SQL package DBMS_STATS for all operations with statistics in Oracle Database 10g and 11g. Alternative approaches, including the ANALYZE command, are not advised.

A SQL script is provided as an attachment to this document on My Oracle Support (Article ID (Doc ID) 781927.1), which contains our recommendations for gathering statistics (see Recommended SQL Script to Collect Optimizer Statistics section).

Steps to Collect Statistics

1. Deleting Optimizer Statistics
2. Locking Optimizer Statistics
3. Collecting Optimizer Statistics

Deleting Optimizer Statistics

It is imperative to delete statistics for all Siebel data tables, if such statistics were collected with deviations from our recommendations. It is recommended to delete such statistics on a per-table basis, immediately before collecting the new statistics. It is not recommended to do a massive delete of statistics followed by a massive collection of statistics. The reason for this is that, in cases when a maintenance window is not large enough, some tables may not be processed, and thus would have no statistics at all.

The following example illustrates how to delete optimizer statistics.

```
EXECUTE DBMS_STATS.DELETE_TABLE_STATS (ownname=> '<table owner>', tabname=> '<table name>');
```

Locking Optimizer Statistics

It is mandatory to drop and lock statistics for Siebel tables with 15 rows or less (automated by the script).

Beginning with Oracle 10gR2, PL/SQL package DBMS_STATS provides functionality to lock statistics on a table or schema. Once statistics are locked, no modifications can be made to those statistics until the statistics have been unlocked.

Thus, the DBA will not have to explicitly drop statistics for such tables every time they do statistics collection. (Note that, on Oracle 9i, the statistics for tables that have only 15 rows or less must be dropped explicitly every time.)

Collecting Optimizer Statistics

Statistics must be collected for all Siebel data tables having more than 15 rows (including indexed columns, indexes, and histograms for all indexed columns). It is important to collect statistics frequently, using the right method and appropriate sample sizes. The following example illustrates the syntax of collecting statistics using DBMS_STATS package:

```
EXECUTE DBMS_STATS.GATHER_TABLE_STATS
( ownname=> '<table owner>'
, tabname=> '<table name>'
, method_opt=> '<recommended method>'
, estimate_percent=> <number>
, granularity=> 'ALL'
, cascade=> TRUE
, DEGREE=> DBMS_STATS.DEFAULT_DEGREE
);
```

In this example, the important options that must be set in accordance with our recommendations have been highlighted.

method_opt option

For both Oracle 10g and 11g, the recommended method is:
'FOR ALL INDEXED COLUMNS SIZE 254'

However, there are a few exceptions to this rule. It has been determined through practical experience that, for a few tables in Siebel CRM schema with skewed data, collection of all column statistics histograms may lead to generation of better query plans. As of this writing, we have identified three such tables:

S_POSTN_CON
S_ORG_BU
S_ORG_GROUP

For these tables, in both Oracle 10g and Oracle 11g the recommended method is:
'FOR ALL COLUMNS SIZE 254'

estimate_percent option

In Oracle 11g, it is recommended to use automatic statistics sampling. For that purpose, the estimate_percent option for all Siebel tables must be set to **DBMS_STATS.AUTO_SAMPLE_SIZE.**

In Oracle 10g, the estimate percent of statistic sampling should be determined based on the number of rows in the table. It is recommended to use large sample sizes, if the maintenance window permits using 100 percent. Otherwise, use the following guidelines:

- For tables with less than 1 million records, use 100 percent
- For tables with more than 1 million records, use no less than 30 percent
- For tables with more than 10 million records, use no less than 10 percent
- For very large tables, with more than 100 million records, estimate_percent must be determined on a case-by-case basis

Frequency of updating Optimizer Statistics

When more than 10% of rows in a table have been inserted, updated, or deleted, it is recommended to re-collect statistics for such table. In addition, statistics must be re-collected when they become outdated. As a guideline, outdated statistics should be re-collected at least once a month. However, if the maintenance window permits, this could be done more often. If there are signs of Siebel CRM application performance degradation, outdated statistics must be collected as soon as possible.

Recommended SQL Script to Collect Optimizer Statistics

A SQL script is provided as an attachment to this document on My Oracle Support (Article ID (Doc ID) 781927.1).

To facilitate the task of collecting and maintaining the optimizer statistics for Siebel CRM, Oracle Database 10g and 11g experts have prepared a PL/SQL script that follows all of the above listed recommendations (see attachment).

At the same time, you need to disable automatic gathering of CBO statistics:

- On Oracle 10g, connect as SYS and execute:

  ```
  EXEC DBMS_SCHEDULER.DISABLE('GATHER_STATS_JOB');
  ```

- On Oracle 11g, connect as SYS and execute:

  ```
  EXEC DBMS_AUTO_TASK_ADMIN.DISABLE('auto optimizer stats collection', NULL, NULL);
  ```

Execute this script connected as SYSDBA

The **coe_siebel_stats.sql** script accepts three parameters:

- **Process Type**
  Valid values for this parameter are “B” (Baseline) or “N” (Normal).
  - "Baseline" process should be used for the first time only, to collect statistics baseline for all Siebel schema objects.

- "Normal" process should be used for all subsequent executions. This process gathers schema object statistics for some tables according to a predefined scheduled based on size.

- **Auto Execute**
  Valid values are Y ("Yes") or N ("No)
  - If Y is passed, the gathering of statistics starts automatically right after coe_siebel_stats.sql completes.
  - If N is passed, the script generates the coe_gather_statistics.sql file, but does not execute it automatically. This provides an opportunity for the DBA to review the output file, and see what tables will have their statistics refreshed.

- **Schema Owner**
  This is an optional parameter with the default value of SIEBEL. If your Siebel table owner is different from the default, you must specify it here. Otherwise, press the Enter key to launch the script.

For possible errors, see **coe_siebel_stats.log.**

You can schedule daily execution of coe_siebel_stats.sql during a low-activity window (maintenance window) after the first run, which collects the statistics based on the aforementioned recommendations and generate a baseline.
There are two options to schedule the execution of this script, as a SQL*Plus script or as a job executing a PL/SQL library:

1. START coe_siebel_stats.sql N Y SIEBEL
2. siebel_stats.gather_siebel_stats('SIEBEL', 'N');

Only those tables that require their stats to be refreshed will be selected.

Siebel Profiling: Detection of columns requiring index

This script result provides a list of columns that are not indexed but potentially are good candidates for indexing according to their usage by the Optimizer. The analysis done by this script is based on historical column usage collected by the Oracle RDBMS. This historical data is gathered through the life of any column in the database and is incremented upon each usage by the optimizer.

The SQL script (coe_siebel_profile.sql) is provided as an attachment to this document on My Oracle Support (Article ID (Doc ID) 781927.1).

The following example shows certain columns to be considered for indexing, which could improve the response time significantly due to a different optimizer's execution plan.

Output Sample

```
Top 25 columns in need of an index
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Partial list of non-indexed columns referenced by at least one SQL
predicate.

PREDICATES TABLE_NAME                     COLUMN_NAME
---------- ------------------------------ ------------------------------
       104 S_ORDER                        X_AGREE_LINE_ID
        84 S_ORG_EXT                      X_IN_COMPANY_SM_ID
        80 S_ORDER                        PR_PAYMENT_ID
        80 S_ORDER                        PR_POSTN_ID
        80 S_ORDER                        X_PROD_ID
        80 S_ORDER                        X_SVC_DLVRY_REVIEWER
        ..... .....
```

Pros:

- Helps the optimizer to generate a better execution plan

Cons:

- Might impact other queries negatively
- Adds overhead for insert, update, and delete operations
- Requires extra storage space

These concerns must be closely examined by testing to decide whether addition of indexes listed in the Siebel profile's output is justified or not. The index definitions need to be added to the Siebel Repository. Otherwise, these indexes are not carried over during upgrades.

Running Siebel Profile

To run the coe_siebel_profile.sql script, you must use SQL*Plus and connect as SYSDBA or a user with DBA privilege. To run this script, see the attachments.

For possible errors see **coe_siebel_profile.log.**

Setting CBO-related Oracle Parameters

More than 60 Oracle database parameters affect the generation of query plans and execution of SQL statements. In this document, we provide recommendations for those parameters that have direct impact on Siebel CRM application performance.

Parameters that are not listed in this document, especially undocumented Oracle hidden parameters, should not be modified from their default values unless you are instructed to do so by Oracle's Application Expert Services or Engineering.

Recommended Database Parameters for Oracle 10g and 11g

Oracle Parameter	10g Recommended Value	11g Recommended Value
OPTIMIZER_FEATURES_ENABLE	10.2.x*	11.x*
OPTIMIZER_INDEX_CACHING	0	0
OPTIMIZER_MODE	ALL_ROWS	ALL_ROWS
QUERY_REWRITE_INTEGRITY	Enforced	Enforced
STAR_TRANSFORMATION_ENABLED	False	False
OPTIMIZER_INDEX_COST_ADJ	1**	1**
OPTIMIZER_DYNAMIC_SAMPLING	1***	1***
QUERY_REWRITE_ENABLED	FALSE	FALSE
SORT_AREA_SIZE	Set PGA_AGGREGATE_TARGET instead	
SORT_AREA_RETAINED_SIZE		
HASH_AREA_SIZE		
WORKAREA_SIZE_POLICY	If PGA_AGGREGATE_TARGET is set, then AUTO	
PGA_AGGREGATE_TARGET	1 GB or greater	
STATISTICS_LEVEL	TYPICAL****	TYPICAL****
MEMORY_TARGET	N/A	60% of total physical memory on DB server *****
MEMORY_MAX_TARGET		
_always_semi_join	OFF	OFF
_b_tree_bitmap_plans	FALSE	FALSE
_partition_view_enabled	FALSE	FALSE
_gc_defer_time	0	0
_no_or_expansion	FALSE	FALSE
_optimizer_max_permutations	100	100

* It is always a good idea to set the parameter **OPTIMIZER_FEATURES_ENABLE** to a value corresponding to the Oracle Database patch version being used, in order to get the full benefit of the optimizer features supported by the Oracle Database version. Hence, if your Oracle Database version is already 10.2.5, then you can set the OPTIMIZER_FEATURES_ENABLE = 10.2.0.5.

** The parameter **OPTIMIZER_INDEX_COST_ADJ** is of critical importance for Siebel CRM application performance. Setting this parameter incorrectly may result in severe performance degradation.

For Oracle 10g or 11g, the CBO setting **OPTIMIZER_INDEX_COST_ADJ** = 1 is strongly recommended because in-house tuning of Siebel CRM application was performed with OPTIMIZER_INDEX_COST_ADJ = 1 setting. However, Oracle default setting of 100 has also shown good results in certain cases. Customers who want to implement OPTIMIZER_INDEX_COST_ADJ = 100 on Oracle 10g or 11g may need to allocate extra development time for

additional tuning. Such tuning is not considered to be an optimizer defect or a regression issue. Oracle Engineering will only review cases when it is confirmed that performance for a specific query is equally unacceptable both under OPTIMIZER_INDEX_COST_ADJ = 1 and OPTIMIZER_INDEX_COST_ADJ = 100. In all other cases, additional tuning must be done by the customer or by Oracle's Application Expert Services, or else OPTIMIZER_INDEX_COST_ADJ setting should be reverted to 1.

*** **OPTIMIZER_DYNAMIC_SAMPLING** parameter was introduced in Oracle 9i with the default settings of 1. Beginning with Oracle 10g, the default setting was increased to 2. Increasing the value of this parameter allocates more resources to dynamic sampling, in terms of both the type of tables being sampled by the optimizer (analyzed or unanalyzed), and the amount of I/O spent on sampling. Our recommendation is to set OPTIMIZER_DYNAMIC_SAMPLING = 1, and follow all our instructions pertaining to the collection of Optimizer statistics for Siebel CRM data tables. From practical experience, in Siebel CRM environments such combination is known to produce good and reasonably stable query plans.

**** **STATISTICS_LEVEL** specifies the level of collection for database and operating system statistics. In Siebel CRM production environments, we strongly recommend to set this parameter to TYPICAL level. This is the default level for Oracle 10g and 11g.

***** Oracle 11g introduces Automatic Memory Management (AMM) feature, which allows managing both the SGA memory and the instance PGA memory automatically.

Our recommendation to Siebel CRM customers on Oracle 11g is to switch to the AMM. However, this setting is not optimal in all cases, such as Oracle RAC implementations.

To implement AMM, two new initialization parameters were introduced in Oracle 11g: **MEMORY_TARGET** and **MEMORY_MAX_TARGET**.

It is recommended to set MEMORY_TARGET to a value equaling 60% of the total physical memory on the database server.

It is not recommended to explicitly sett MEMORY_MAX_TARGET. This parameter is optional, and if you do not set it explicitly, it is automatically set to the value of MEMORY_TARGET.

Hard-coded Session Parameters for Siebel OM Sessions

Beginning with Siebel 7.7 and 7.8 on Oracle Database 9i, the Siebel database connector was modified to make a few ALTER SESSION statements for OLTP operations. The same functionality has been built into the Siebel database connectors for Oracle 10g and 11g. Thus, in all current Siebel CRM versions (7.7 and onward), each Siebel Application Object Manager session automatically sets the following session parameters:

- ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_10*
  When optimizer_mode is set to first_rows_n, the optimizer in all cases uses cost-based optimization, and sets the optimizer goal for best response time (instead of best throughput).
- ALTER SESSION SET HASH_JOIN_ENABLED = FALSE
  In Oracle 10g and 11g, this parameter is deprecated; Siebel CRM uses _hash_join_enabled instead and sets it to FALSE.
- ALTER SESSION SET _OPTIMIZER_SORTMERGE_JOIN_ENABLED = FALSE
- ALTER SESSION SET _OPTIMIZER_JOIN_SEL_SANITY_CHECK = TRUE
  (These parameter settings enable sanity checks for a join using two columns.)

* There have been multiple discussions regarding the FIRST_ROWS_10 setting. This is a very important question which relates to the overall Siebel CRM architecture and design philosophy. Siebel CRM was designed with scalability as a top priority. The goal is for Siebel CRM to support very large numbers of users without exhausting system resources. Thus, we provide some mechanisms to restrict user access to such resources.

FIRST_ROWS_10 has been one of the key elements to support this goal. Oracle internal testing confirms that FIRST_ROWS_10 setting produces good results when Siebel CRM application is used and configured properly. For instance, customers are advised that Siebel CRM users should not perform random sorts on huge result sets, or execute queries without appropriate search specifications. Best practice to address such issues would be to create meaningful predefined queries, and tune them. Follow the guidelines in "Tuning Customer Configuration for Performance", Chapter 12 of the *Siebel Performance Tuning Guide* in Siebel Bookshelf.

The setting OPTIMIZER_MODE = FIRST_ROWS_10 must not be altered, the same as for all other hard-coded session parameters listed above. The ALTER SESSION commands are issued by the Siebel Database connector module. No mechanisms are provided for altering this behavior. No attempts to devise an override for these session parameter settings should be done by any means.

Peeking of User Binds in Siebel CRM

In the Siebel CRM application, the Siebel Application Object Manager (AOM) generates SQL statements with the syntax :n in the WHERE clause, for example:

```
select LAST_NAME from S_CONTACT where TIMEZONE_ID=:n;
```

The :n is a placeholder for values to be replaced at run-time. At run-time, the Siebel AOM substitutes the real values for the placeholder(s) to seek a better query plan. This process is called "peeking of user binds" or simply "bind peek".

On a hard parse, Oracle will peek the value provided for :n, and optimize the query as if the same query was submitted with this particular literal value. This allows the optimizer to leverage optimizer statistics and generate the most efficient query plan for given value.

The implementation of bind peeking in Siebel CRM may vary depending on the base Siebel CRM version and the Fix Pack or Quick Fix version. For example, in Siebel CRM 7.7 and 7.8, bind peeking was implemented through a hard-coded call to Oracle Call Interface (OCI), issued by the Siebel AOM.

The most recent implementation of bind peeking available (please see the next section, **Major Change in Peeking of User Binds in Siebel CRM**) in Siebel CRM relies on the operating system environment variable SIEBEL_ORA_BIND_PEEK. The default value for SIEBEL_ORA_BIND_PEEK is FALSE, which means the Siebel AOM sends SQL statements using a bind variable. The Oracle DBMS will peek the bind to optimize the query using statistics that include histograms. However, when SIEBEL_ORA_BIND_PEEK is set to TRUE, the Oracle DBMS will peek bind a second time if the value of the bind variable is different.

It is recommended that Siebel CRM customers deploy a Fix Pack or Quick Fix for their base application version, and leverage the environment variable SIEBEL_ORA_BIND_PEEK to enable or disable bind peeking based on their requirements. The recommended setting is FALSE.

Attempts to override the Siebel CRM implementation of bind peeking by means of Oracle database server parameter settings are also discouraged. It has been reported that some Siebel CRM customers have experimented with the hidden parameter _optim_peek_user_binds. However, this is an internal Oracle parameter only; it is not documented, and must not be changed except according to instructions from Oracle's Application Expert Services or Engineering.

The following table contains the list of Siebel CRM Fix Pack and Quick Fix versions that deliver SIEBEL_ORA_BIND_PEEK functionality:

Base Siebel CRM Version	Siebel CRM Versions with SIEBEL_ORA_BIND_PEEK support
Siebel 7.7	7.7.2.12 Fix Pack Build2 [18391]
Siebel 7.8	7.8.2.10[19241]QF0A52 7.8.2.11[19244]QF0B09 7.8.2.13 Fix Pack
Siebel 8.0	8.0.0.6 [20423]QF0637 8.0.0.6 Quick Fix 26xx Branch 8.0.0.7 [20426]QF0707 8.0.0.7 [20426]QF1703 8.0.0.7 Quick Fix 17xx Branch 8.0.0.8 Fix Pack Build2[20428] 8.0.0.9 Fix Pack
Siebel 8.1	8.1.1 [21112] QF0023 8.1.1.2 Fix Pack

Major Change in Peeking of User Binds in Siebel CRM

Oracle is introducing a change specifically for the Siebel application that will eliminate the requirement for the SIEBEL_ORA_BIND_PEEK variable setting. This change will reduce hard-parsing to only a single one when the SQL statement is first processed. Subsequent executions of the same statement (with the same or different bind values) will not generate more hard-parsing. Other improvements are included to further enhance SQL performance, which will be explained in detail when the fix becomes available.

This fix will be included in the Fix Packs targeted for mid-2011 and Siebel CRM subsequent major releases for Oracle 10g and 11g on all supported platforms for Siebel CRM. The fix availability will be communicated through the “Bind peeking performance degradation future enhancement” Alert (Doc ID 1273535.1) on My Oracle Support.

Monitoring SQL Performance

Siebel CRM, like all applications, must be monitored on a regular basis. Siebel CRM running on an Oracle database can use many tools and utilities for performance monitoring, such as Automatic Workload Repository (AWR), Oracle SQL Trace and Tkprof, Siebel log files, Siebel SQL tagging (8.1.1.2 and after). The major task here is to identify the SQL statements that perform poorly. AWR (SQL Statistics section) is probably the most suited for a production environment, due to insignificant performance impact.

Siebel Log Files - Siebel Event Log level 4

Siebel event log levels are explained in *Siebel System Monitoring and Diagnostics Guide* in Siebel Bookshelf. To get full SQL information, including the bind variables, a command like the following could be used for all Siebel Servers.

*change **evtloglvl** %**SQL**%=4 for component sccobjmgr_enu server <**siebel** server name>*

In a production environment, the above command will produce large log files, which are hard to visualize, parse, and analyze. Also, setting this parameter to values of 4 or 5 has a performance impact. So, if evtloglvl is set to any value other than 1, it is important to use those settings cautiously, for very limited duration, and in a controlled manner.

Siebel SQL Tagging (8.1.1.2 and after)

SQL tagging is a new feature for version 8.1.1.2 that, once set, provides the following information in the Siebel Application Object Manager log files:

- *componentname* is the alias of the component, for example, SCCObjMgr_enu.
- *servername* is the name of the Siebel Server on which the component or task is running.
- *taskid* is the task ID of the user that generated the query.
- *userid* is the login name of the user who generated the query.
- *flowid* is the flow ID of the component or task.
- *sarmid* is the SARM ID of the component or task.
- *busobjname* is the business object name.
- *buscompname* is the business component name.
- *viewname* is the view name (only in UI mode).

SQL Tagging Format

SQL tagging information is formatted as a comma-separated list of values using the following syntax:
<componentname>,<servername>,<taskid>,<userid>,<flowid:sarmid>,<busobjname>,
<buscompname>,<viewname>

NOTE: Any optional elements of a tag that are irrelevant or missing for the query are replaced with an empty string.

For more information about SQL tagging, see *Siebel System Monitoring and Diagnostics Guide* (8.1 Revision A – August 2009) in Siebel Bookshelf.

Sample SQL-Tagged Code

The following is a sample of how a tagged SQL statement might appear in a log file when the SQL statement was generated from a Siebel Call Center Object Manager component (for example, SCCObjMgr_enu, for U.S. English) and an Oracle database. The changes made by the SQL tagging feature appear in italics.

```
SELECT
FIRST_NAME,
LAST_NAME,
…,
:1
FROM
TBO.S_CONTACT
…
WHERE
LAST_NAME LIKE:2
ORDER BY
…
Bind variable 1:
SCCObjMgr_enu,sdchs20i046,10485776,SADMIN,00000089489108a8:50557,Account,Account,
Account List View
Bind variable 2: Foo*
```

The above information provides the source of SQL in the Siebel CRM application, and thus facilitates the analysis and implementation of possible changes that could be made to address the performance issues.
Note:Due to addition of ":1" in the query, the Oracle RDBMS considers the SQL SELECT statement to be a new and different statement compared to the base version when this feature is not ON. Therefore, Stored Outlines and SPM Plans that were captured without ':1' are not used.

Diagnostics Using SQL Query Execution Plan

In production, the fastest way to track down the performance issues including SQL is to start from HW/OS configuration to Oracle instance & database settings. Hence, a sanity check of these areas is a must.
Analyzing CBO Query Plan is the best method to determine the cause of a given poor-performing SQL query when other factors like SGA memory size and IO throughput are carefully examined and discarded. A good starting point to select the SQL statement with poor performance is AWR.

SQL Statements and Their Bind Variables

In order to select the SQL statement for analysis, you need the query, including its bind variables. AWR or Tkprof provide a consolidated result of SQL statements, which includes the number of parses, executions, fetches, and so on. However, these tools do not provide bind variables of a single query execution. SQL statements and their binds can be retrieved using Siebel Event Log Level 4 (see "Siebel log files - Siebel Event Log level 4") or by adding "/s Siebel_trace_file_name" to the Siebel Developer Web Client command line or to its short cut (see "Specifying SQL Spooling in Siebel Developer Web Client" in the *Siebel Performance Tuning Guide*).
Generating, parsing, and executing the SQL statement in order to produce its CBO query plan is best when it is done through Siebel CRM, because, among many benefits, the Siebel AOM will manage the fetches through the database cursor that it starts and manages. For example, if the result set contains 2,000 records, then the Siebel AOM (such as Siebel Call Center Object Manager) always fetches 50 records, unless the users drills down; in that case the next 50 records are fetched from the cursor. Hence, if the above execution of SQL statement was simulated through SQL*Plus, there would have been 2,000 records fetched, significantly adding to the total time.

SQL Trace and Tkprof can be used to obtain the CBO query plan when Siebel CRM is used.

Important Considerations When Using SQL*Plus

With the above caveat in mind, sometimes for control, simplicity and also having rapid access to CBO query plan details through the use of DBMS_XPLAN, SQL*Plus can be used. However, the following setting must be done before.

1. The Oracle database parameters that the Siebel AOM sets using ALTER SESSION:

    - ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_10
    - ALTER SESSION SET HASH_JOIN_ENABLED = FALSE
    - ALTER SESSION SET _OPTIMIZER_SORTMERGE_JOIN_ENABLED = FALSE
    - ALTER SESSION SET _OPTIMIZER_JOIN_SEL_SANITY_CHECK = TRUE

2. **Passing values through bind**

    Examples:

    ```
    -- declare bind variables and set them
    variable b1 number;
    variable b2 varchar2(100);
    variable b3 char(1);
    begin
    :b1 :=1;
    :b2 :='1-8CSJQG';
    :b3 :='Y';
    end;
    /
    ```

    Examples 2: DATE (special case)

    Example 2a:

    ```
    var b4 varchar2(100);
    exec :b4 := '01-jan-2010';
    select appt_start_dt, ACTIVITY_UID from Siebel.S_EVT_ACT where appt_start_dt
            > TO_DATE(:b4, 'DD-MON-YYYY');
    ```

    Example 2b:

    ```
    var b4 varchar2(100)
    exec :b4 := '01-jan-2010 17:25:15';
    select appt_start_dt, ACTIVITY_UID from Siebel.S_EVT_ACT where appt_start_dt
            > TO_DATE(:b4, 'DD-MON-YYYY HH24:MI:SS');
    ```

    Remark: The value assigned to :b4 and mask should match. If the mask is not included, Oracle will use nls_date_format, which could be same or not.

Note: It is important to pass values through the bind. If the binds are directly replaced by their values in the SQL statement, CBO might generate a totally different execution plan.

For generating an explain plan, DBMS_XPLAN, or using SQL Trace and Tkprof, see *Oracle Database Performance Tuning Guide* in Oracle Database Reference Documentation.

SQLT

Oracle Global Customer Support provides a tool called SQLT. SQLT XTRACT method takes SQL_ID as argument and produces the execution plan used by the corresponding query, together with binds values and a script that can reproduce the execution of the query in a stand-alone mode from SQL*Plus. For full description, kit download and examples, please see Article ID (Doc ID) 215187.1 on My Oracle Support.

Siebel Configuration Guidelines for Performance

This section describes some of the most important SQL tuning recommendations. For their explanation and details of other major best practices, see "Tuning Customer Configurations for Performance" in *Siebel Performance Tuning Guide* in Siebel Bookshelf.

- Always evaluate and add index on columns used in joins, sort and search specifications (see Siebel Profiling: Detection of columns requiring index)
- Avoid using sort specifications on non-indexed columns or joined columns
- Limit the use of case insensitivity for queries
- Avoid overly complex user interface configuration
- Limit the number of business components in a view
- Limit the number of virtual business components in a view
- Limit the number of records returned
- Limit the number of joins, extension tables, and primary ID fields in a business component
- Use inner joins rather than outer joins
- Remove unneeded sort buttons
- Provide tuned PDQs
- Avoid calculated fields that do Counts and Sums

Siebel SQL Statements

The Siebel Repository File is the base for generation of SQL statements by the Siebel AOM. This file contains metadata definitions of all the application objects. This is where all the UI elements are defined, as well as the business logic.

Modifying the SRF has a direct impact on SQL generation, in that new relationships can be established (joins or links) or new fields. These changes will result in entirely new SQL statements.

Changes in Generated SQL Statement

Compiling a new SRF could potentially produce a SRF where the new generated SQL statement is different from the original SQL statement. The SQL statement is logically the same, but the order of some of the columns in the SQL statements select list is different.

The following is a partial list of SRF changes that could result in SQL statement changes:

Configuration:

- Calculated fields calling business services using InvokeSvcMethod …
- User pre-defined queries or ad hoc queries
- Runtime events that may have added something to the action set
- Use of Data Validation Manager
- Adding toggle applets to views.
- If sequence of column order displayed is not defined correctly in Siebel Tools, the SQL generated with each compile could be different
- The order and the displayed columns in any list applet, when changed, can result in a different SQL statement. To ensure that tests across environments are accurate, it is advised to make sure that the list applet columns are the same. Reset by going to Columns Displayed > Reset Defaults

Scripting:

- In case of scripting, the data can be the culprit without ever changing any configuration. This is because some piece of data makes the execution of the script take some different path than usual

Siebel Server/UI:

- Certain system settings which dictate how queries are created can influence the resulting SQL. Examples:
  - Default MVG Exists Query
  - AutomaticTrailingWildcards

Diccache.dat file:

- In rare instances, a corrupt or invalid diccache.dat file may result in modified SQL. In cases where this is suspected, it is advised to create a backup of the current file and delete the original. A fresh copy will be created when the Siebel Servers are restarted.

Change of SQL Statement when SQL Tagging is used

See Siebel SQL Tagging (8.1.1.2 and after)

Recommendations: No magic formula

- Change Control
- Improving overall database's query performance
  - Gathering Statistics (including histograms)
  - Reviewing indexes
  - Database configuration and setup (including certain parameters)
- Terminating long running queries

Stored Outlines

A Stored Outline allows preserving and maintaining an execution plan (captured with certain conditions or hints) for a query performing slower than expected without such an outline. At run-time, CBO is forced to use this execution plan[1], regardless of changes in the environment configuration or in associated statistics.

The detailed discussion on usage of a Stored Outline is beyond the scope of this document. The goal of this topic is to provide guidance on when to use a Stored Outline while tuning Siebel CRM application performance.

The main point may be expressed in this way: "Fix the root cause, not the symptom!"

As a general rule, Stored Outlines are not a viable solution for Siebel CRM application performance issues. Stored Outlines must not be used during Siebel CRM application development and performance testing or tuning. If such Stored Outlines exist, they should be dropped in order to expose all performance problems that need to be properly fixed.

NOTE: In rare cases, when a product defect is confirmed, or when it is not feasible to implement a configuration fix to address a performance problem in a production system, an Oracle support or engineering representative may instruct you to use a Stored Outline. However, such a Stored Outline is temporary and must be dropped as soon as a permanent fix is available.

The major issues with using Stored Outlines relate to Siebel CRM or Oracle Database upgrades, and customer functional releases of their Siebel CRM applications. Even a full recompilation of the SRF in certain cases could generate new SQL statements even when no change has been made to the underlying entity, such as a Siebel view or applet. Even though the new SQL statement is logically identical, the column order in the select list could be different. In such event, CBO considers the new statements different and will not use the Stored Outlines. This in turn could produce significant performance degradation, because the previous SQL statement was tuned through the use of Stored Outline (see Changes in generated SQL statement for more details).

Stored Outlines may cause performance degradations after the Oracle database upgrade, because each database version may retire a set of database parameters that affect the query plan preserved by the Stored Outlines.

Dropping Stored Outlines

To comply with the above recommendations, it may be necessary to identify the Stored Outlines that exist in the environment, check if the outlines have been used, and understand their intended usage.
[USER|ALL|DBA]_OUTLINES view should be used to display information about existing outlines:

```
SELECT name, category, used FROM user_outlines;
```

Stored Outlines that were not created upon instruction from Oracle's Application Expert Services or Engineering must be removed through meticulous implementation of the guidelines in this document. However, there are complex cases that might need assistance from Application Expert Services for more advanced CBO tuning.

The Stored Outlines can be dropped by using a PL/SQL package DBMS_OUTLN.

[1] In certain cases, CBO ignores the Outline if it cannot use it. For example, if an index that the Outline is supposed to use is dropped the CBO will ignore the Outline and generates a new query plan.

For example:

```
BEGIN
 DBMS_OUTLN.drop_by_cat (cat => '< category >');
END
/
```

SQL Plan Management

When SQL plans are required, it is strongly recommended to use SQL Plan Management (SPM) to record and evaluate the execution plans of SQL statements over time. SPM provides the ability to build baselines that are known to be efficient and that are composed of existing plans or plans imported either from other instances or previous versions. SQL plan baselines are used to maintain performance of the matching SQL statements. Over time, new plans are evaluated and preserved when their performance improvements are significant. Subsequently, these plans can be evolved to "accepted" if their performance is superior to the baseline's SQL plan.

For more information about SQL Plan Management please refer to *Oracle Database Performance Tuning Guide* in Oracle Database Reference Documentation.

Transitioning from RBO to CBO

When you transition your deployment from RBO to CBO, a few problems must be addressed to avoid Siebel CRM application performance degradation. This primarily pertains to specific tuning due to customizations implemented in addition to Siebel out-of-the-box configuration and tuning:

- RBO-specific tuning must be withdrawn. The list below shows some examples of such RBO-specific tuning:
    - Customized single-column indices that worked best for the RBO need to be dropped for CBO to avoid generating suboptimal query plans in the CBO context.
    - Hard-coded predicates embedded into the SQL for the purpose of forcing RBO to use a certain index need to be removed.
- The environment needs to be reconfigured with the parameters and settings recommended for CBO. (More information about recommendations on environment configuration and methods of statistics collection can be found later in this document.)
- CBO-specific tuning may need to be put into place as necessary. The primary focus should be on the customer's extensions and a limited number of cases when the Siebel CRM default configuration requires additional tuning with respect to customer environment specifics.

Transition from RBO to CBO is a one-time effort that may take from a few days to a few weeks, depending on the level of application customization, data volume, data shape, etc.

Transitioning from Oracle CBO 9i to Oracle CBO 10g/11g

Under normal circumstances, migrating from Oracle CBO version 9i to later versions of CBO (such as 10g or 11g) is not expected to cause Siebel CRM application performance degradation, if the recommendations in this document are followed. There is a possibility of encountering some isolated cases of performance regression, but all such occurrences must be studied and resolved on a case-by-case basis.
Where recommendations are not followed, almost any major degradation caused by migrating from Oracle 9i CBO to later versions of CBO falls into one or more of the following categories:

- The Siebel CRM application was not tuned to work well with Oracle 9i CBO in the first place.
- The level of Siebel CRM application customization is very high, and additional tuning may be required to address customer-specific configurations.

In general, transitioning from Oracle CBO version 9i to later versions of Oracle CBO is a one-time effort of much lesser scope than the migration from Oracle RBO to the CBO. However, customers who upgrade from Oracle 9i CBO to a later version of Oracle CBO in parallel with deployment of a customer application release may need to allocate some additional development time for performance tuning.

Oracle 9i recommended parameters settings

Oracle Parameter	Recommended Value for 9i
HASH_JOIN_ENABLED	TRUE
OPTIMIZER_FEATURES_ENABLE	9.2.0*
OPTIMIZER_INDEX_CACHING	0
OPTIMIZER_MODE	CHOOSE
QUERY_REWRITE_INTEGRITY	Enforced
PARTITION_VIEW_ENABLED	FALSE
STAR_TRANSFORMATION_ENABLED	FALSE
OPTIMIZER_DYNAMIC_SAMPLING	1
OPTIMIZER_INDEX_COST_ADJ	1
OPTIMIZER_MAX_PERMUTATIONS	100
QUERY_REWRITE_ENABLED	FALSE
SORT_AREA_SIZE	Set PGA_AGGREGATE_TARGET instead
SORT_AREA_RETAINED_SIZE	Set PGA_AGGREGATE_TARGET instead
PGA_AGGREGATE_TARGET	1 GB or greater
STATISTICS_LEVEL	TYPICAL in production
HASH_AREA_SIZE	Set PGA_AGGREGATE_TARGET instead
WORKAREA_SIZE_POLICY	If PGA_AGGREGATE_TARGET is set, then AUTO
_always_semi_join	OFF
_b_tree_bitmap_plans	FALSE
_partition_view_enabled	FALSE
_gc_defer_time	0
_no_or_expansion	FALSE

- It is always a good idea to set the OPTIMIZER_FEATURES_ENABLE = <Oracle DB patch version being used> to get the full benefit of a series of optimizer feature supported by the Oracle Database version. Hence, if your Oracle Database version is already 9.2.x, then you can set the OPTIMIZER_FEATURES_ENABLE=9.2.x.

Parameter OPTIMIZER_INDEX_COST_ADJ

This parameter is of critical importance for Siebel CRM application performance. Incorrect setting may result in severe performance degradation of the Siebel CRM applications.

- For Oracle 9i, the CBO setting **OPTIMIZER_INDEX_COST_ADJ = 1** is imperative. This parameter sets the optimizer goal for best response time (versus best throughput). Incorrect setting may cause the optimizer to favor full-table scans instead of index access.

Siebel CRM Versions and Supported Oracle Database Versions

The table below represents supported combinations of Siebel CRM and Oracle RDBMS versions.

Siebel CRM Version	Oracle RDBMS Product	Oracle RDBMS Version	Optimizer Type	INIT.ORA parameter OPTIMIZER_MODE
Siebel 7.7	Oracle 8i Enterprise Server	8.1.7.4 + p1744093 or above for Unix 8.1.7.4.6 or above for Windows	RBO	RULE
Siebel 7.7	Oracle 9i Enterprise Server	9.2.0.4 or above and within the 9.2.0.x series	CBO	CHOOSE
Siebel 7.7	Oracle 10g Enterprise/Standard Server	10.1.0.3 or above, Including 10gR2	CBO	ALL_ROWS
Siebel 7.7	Oracle 11g Enterprise/Standard Server	10.1.0.3 or above, Including 10gR2	CBO	ALL_ROWS
Siebel 7.8	Oracle 9i Enterprise Server	9.2.0.4 or above and within the 9.2.0.x series	CBO	CHOOSE
Siebel 7.8	Oracle 10g Enterprise/Standard Server	10.1.0.3 or above, Including 10gR2	CBO	ALL_ROWS
Siebel 7.8	Oracle 11g Enterprise/Standard Server	11.1.0.6 or above	CBO	ALL_ROWS
Siebel 8.0	Oracle 10g Enterprise/Standard Server	10.2.0.2 or above	CBO	ALL_ROWS
Siebel 8.0	Oracle 11g Enterprise/Standard Server	11.1.0.6 or above	CBO	ALL_ROWS
Siebel 8.1	Oracle 10g Enterprise/Standard Server	10.2.0.2 or above (with Oracle 11g client)	CBO	ALL_ROWS
Siebel 8.1	Oracle 11g Enterprise/Standard Server	11.1.0.6 or above	CBO	ALL_ROWS

With the exception of Siebel CRM version 7.7 on Oracle 8i Enterprise Server, all the above listed Siebel CRM versions can only be configured to use the Cost Based Optimizer (CBO). This differs from the recommended configuration for pre-Siebel 7.7 CRM versions, which utilized the Rule Based Optimizer (RBO) on Oracle 8i Enterprise Server.

In the vast majority of cases, performance of the Siebel CRM applications with properly implemented and configured CBO will meet or exceed RBO performance, all things being equal (that is, under similar architecture/network topologies, hardware configurations, data volumes and distributions, storage characteristics, etc.). Detailed discussion of the Rule-Based Optimizer (RBO) is beyond the scope of this document. The documents in the following list provide general guidelines for RBO to CBO migration.

Bibliography and Useful References

This document is partially based on Siebel/Oracle bulletins and white papers published in the past, with our special thanks to the authors and contributors:

- Oracle CBO and Siebel Business Applications (Article ID (Doc ID) 478028.1) by
  Selma Takara, Principal Technical Support Engineer, Oracle Corp
- Siebel CRM with Oracle® Cost-Based Optimizer (CBO) by
  Dillip Kumar Praharaj, CMTS, Oracle Corp
  Anna Leyderman, Development Manager, Oracle Corp
  Kavitha Raghunathan, SMTS, Oracle Corp
- Tuning Oracle Cost Based Optimizer (CBO) for Siebel OLTP CRM Applications by Dominic Stewart
  Principal Sales Consultant, Oracle Corp
- Troubleshooting Guide for Siebel Performance after Upgrade to Oracle 10G, by Caroline Wanyonyi,
  Principal Architecture Specialist, Oracle Corp

However, it is important to mention that the current document should be considered as the ultimate reference. The recommendations in this document take precedence over all other documents on this subject if any other document is advising otherwise.

Oracle Corporation Documentation

- Oracle 9i Database Performance Tuning Guide and Reference, Release 2 (9.2).
- Oracle9i Supplied PL/SQL Packages and Types Reference.
- Oracle 10g Database Performance Tuning Guide and Reference, Release 2 (10.2)
- Oracle10g Supplied PL/SQL Packages and Types Reference

Siebel Product Defects Affecting Performance on Oracle RDBMS

Siebel CR	Description	Resolution/FR Targets
12-R5WXCB	Siebel 8.0 client sets session parameter HASH_JOIN_ENABLED, which is deprecated in Oracle 11g. **Note:** This problem was corrected for the Siebel 8.0 client on the Oracle 10g platform. **Note:** This problem does not exist for the Siebel 8.1.1 client on Oracle 11g platform.	Fixed in: 8008 FixPack 8005 QuickFix
12-1TK6KQ9	There is a bug in the DBConn, which does not recognize Oracle 11g banner and will cause performance problem.	12-1TK6KQU (Aspen) 12-1TK9A1L (8.0.0.8 FP) 12-1UI7Y8T (7.7.2.10 QF) 12-1UIA88O (7.7.2.12 FP) 12-1UIA89R (7.8.2.x FP) 12-1UIA8AQ (8.1.1.2 FP) 12-1V14HVD (7.7.2.8 QF)

Oracle Product Defects Affecting Siebel

Oracle 9i Product Defects

Oracle Bug/Note	Description	Resolution
Bug 3756797	High CPU may be used during parse of certain query forms	
Bug 3239873	High CPU during parse of statements with many identical predicates	
Bug 3737955	Long parse times for long inlists / many AND/OR terms.	Fixed in 9.2.0.7 (or patch)
Bug 3406977	HIGH VERSION COUNT DUE TO BINDS MARKED AS NONDATA WITH CURSOR_SHARING=FORCE.	Fixed in 9.2.0.6
Bug 3321724	EXCESSIVE HIGH VERSION COUNT WHEN CURSOR_SHARING=SIMILAR.	Not fixed
Bug 3898338	HIGH PARSE TIME WITH USE_CONCAT AND INLISTS AT EMPTY TABLE.	Fixed in 10.2
Bug 3075139	CBO may produce a suboptimal plan in FIRST_ROWS_N mode due to incorrect cardinalities being used.	Fixed in 9.2.0.5
Bug 2991526	Incorrect cardinality estimates may occur for LIKE predicates resulting in poor execution plans.	Fixed in 9.2.0.5
Bug 4275247	Statistics on empty tables (probably in conjunction with FIRST_ROWS_N). Source of Alert 1162.	Internal only.
Bug 3380026	Suboptimal plan possible for FIRST_ROWS for SQL with ORDER BY and indexes.	Fixed in 9.2.0.6
Bug 3220073	ORDER BY sort may not be eliminated in First_rows_NNN.	Fixed in 9.2.0.5
Bug 3075139	Suboptimal plan possible in FIRST_ROWS_N mode.	Fixed in 9.2.0.5
Bug 3018146	Long parse times for many ORed predicates with FIRST_ROW_XX optimization.	Fixed in 9.2.0.5
Bug 3566843	Optimizer may not choose best index.	Fixed in 9.2.0.7
Bug 4335559	Ora-00600 [Kkojcio] on a query	Fixed in 9.2.0.6

Oracle 10g Product Defects

Oracle Bug/Note	Description	Resolution
4878299	Bad plan from join with FIRST_ROWS_N or a ROWNUM predicate ORDER BY elimination may not find best plan	Patch in 10.2.0.4 (Server Patch) Fixed in 11.1.0.6 (Base Release)
7430474	FIRST_ROWS_K recost for ORDER BY elimination may not find best plan	Fixed in 11.1.0.8 Related to Bug 5288623
6990305	Suboptimal plan possible as bind peeking not used during CBQT	Fixed in 11.1.0.8
7236148	CBO does not prorates chained_cnt under first_rows_k	Fixed in 11.1.0.8
Note 406966.1	Intermittend wrong results on 9iR2,10gR1 and 10gR2	Fixed in 10.2.0.4
Note 4724074.8	Increased Parse Time	Fixed in 10.2.0.4
Bug 6455161	Higher "cache buffer chains" latch gets / higher "consistent gets" after truncate/Rebuild	Fixed in 10.2.0.4
Bug 5240607	FIRST_K_ROWS may choose inefficient NESTED LOOPS join	Fixed in 10.2.0.4
Note 4878299.8	Bad plan from join with FIRST_ROWS_N or a ROWNUM predicate	Fixed in 10.2.0.4
Bug 7891471	Query with ORDER BY has a bad plan with FIRST_ROWS_10 optimization in 10.2.0.4	Fixed in 10.2.0.5
Bug 4742607	"cache buffer chains" latch contention from concurrent index range scans in extreme situations, excessive CPU consumption without any user-SQL activity.	Fixed in 10.2.0.3

Oracle 11g Product Defects

Oracle Bug/Note	Description	Resolution
8839301	On Oracle 11.2.0.0.2 only: ALTER TABLE command to add a CHAR (1 char) column creates such column with length 4.	To be fixed in the next distributed version

Performance Tuning Guidelines for Siebel CRM Application on Oracle
Article ID – Doc ID 781927.1
December 2010
Authors: James Qiu, Paul Blokhin, Mehdi Gerami
Major Contributors: Carlos Sierra, Abel Macias, Dave Henriquez

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com