Autodesk User Group

Oracle Performance Tuning

Daniel von Känel

Project Manager PMP

AUTODESK.

Solution Tuning (1)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure infrastructure (Oracle instance parameters impacting CBO)	**Issue**: It has been observed that CBO may select a "slow" plan (including FULL TABLE scans) for SQL generated by Map 3D client application while there are indexes available (as configured by the product) **Analysis**: Customer disabled CBO adaptive capabilities as per recommendations from Oracle: • *optimizer_adaptive_features FALSE* • *optimizer_adaptive_reporting_only TRUE* Evaluated CBO settings influencing the use of indexes in Plans. *alter session set optimizer_index_caching=10; (default 0)* *alter session set optimizer_index_cost_adj=30; (default 100)* **Result**: After setting optimizer parameter at the session's level, the Team observed changes to the SQL execution plans (improved plans). Thus confirming that CBO can be optimized and the proposed methodology should be followed **Recommendation**: It is not recommend updating the optimizer's parameters at the instance level as such changes to Oracle instance would be impacting all processes, instead Customer should consider applying other fixes targeting the COB at the objects level	All groups

Solution Tuning (2)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure Map 3D product (Applying hints in the Map 3D IM core code)	**Issue**: It has been observed that CBO may select a “slow” plan (including FULL TABLE scans of Map 3D’s system tables (like TB_JOB_VERSION) while there are indexes available (as configured by the product) **Analysis**: When Map 3D generates SQL and requests data, the Oracle server 1. Invokes the Job3.GetPolicy VPD policy function (defined in Map 3D product, VPD implementation) 2. The policy function returns a predicate, based on session attributes and database contents 3. The Oracle server dynamically rewrites the submitted query by appending the returned predicate to the WHERE clause. 4. Oracle CBO selects execution plan based on number of parameters (including data distribution / histogram). Map 3D was used to update statistics. Unfortunately, the CBO were still selecting not-optimal plans (as confirmed by SQL plan analysis). **Recommendation**: Add SQL hints to the Job3.GetPolicy VPD policy function in order to influence generation / selection of the execution plans. This is considered to be a low risk change to the product as it does not change Map 3D functionality, just provides additional instructions to the Oracle CBO. **Result**: Applied changed to Job3.GetPolicy function (see notes below). The Team observed changes to the SQL execution plans (no full table scans of TB_JOB_VERSION). After the change, all joins between the feature class tables and TB_JOB_VERSION table are using the IM indexes as intended by the product).	All groups

Solution Tuning (3)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure Map 3D product (Applying hints in the Map 3D IM core code)	**Issue**: It has been observed that CBO may select a “slow” plan (including FULL TABLE scans of Map 3D’s system tables (like TB_JOB_VERSION) while there are indexes available (as configured by the product) **Analysis**: When Map 3D generates SQL and requests data, the Oracle server 1. Invokes the Job3.GetPolicy VPD policy function (defined in Map 3D product, VPD implementation) 2. The policy function returns a predicate, based on session attributes and database contents 3. The Oracle server dynamically rewrites the submitted query by appending the returned predicate to the WHERE clause. 4. Oracle CBO selects execution plan based on number of parameters (including data distribution / histogram). Map 3D was used to update statistics. Unfortunately, the CBO were still selecting not-optimal plans (as confirmed by SQL plan analysis). **Recommendation**: Add SQL hints to the Job3.GetPolicy VPD policy function in order to influence generation / selection of the execution plans. This is considered to be a low risk change to the product as it does not change Map 3D functionality, just provides additional instructions to the Oracle CBO. **Result**: Applied changed to Job3.GetPolicy function (see notes below). The Team observed changes to the SQL execution plans (no full table scans of TB_JOB_VERSION). After the change, all joins between the feature class tables and TB_JOB_VERSION table are using the IM indexes as intended by the product).	All groups

Solution Tuning (4)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure Solution (display model)	**Issue**: It has been observed that after editing of features in Map 3D, number of layers are being refreshed thus resulting in multiple “flashing windows” each showing a progress bar per layer. This is impacting users’ experience in data editing **Analysis**: Analysis of the Solution display model indicated that number of layers reference the same source feature class table. For example, various Switch related layers (total 13 layers) were created with corresponding Oracle views (one for each FEAT_CODE from the migrated AGDS data). **Recommendation**: Reduce the number of layers in the Solution display model. This can be done by combining layers referencing the same source feature class view/table in one. Map 3D thematic can be configured in a combined layer to symbolize the required feature codes. Note that this change will impact ability to turn on/off layers on the map. It is therefore also recommended to create separate display models for “data editing” (with reduced number of layers) and data analysis / plotting (with ability to turn on / off individual feature classes). **Result**: The Team merged layers for EL_SWITCH feature class, observed improvement in the graphics generation and reduced time of map refresh after features editing operations.	Slow graphics generation

Solution Tuning (5)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure Solution (Applying hints in Oracle views driving the graphics generation)	**Issue**: It has been observed that generation of graphics is slow for feature classes with large number of records (>1 mln). Layers in the solution's display model reference corresponding Oracle views. **Analysis**: The team captured SQL that is executed by slow generation layers. Analysis of respective execution plans demonstrated that full table scans are selected by CBO for feature class tables thus resulting in a slow performance of graphics generation. Performance of layers with smaller number of records (<500K) was considered to be good **Recommendation**: Add SQL hints (see notes for an example) to the Solution Oracle views that are referenced by slow rendering layers **Result**: Applied changed to view definitions. Performance of graphic generation was improved.	Slow graphics generation

Solution Tuning (6)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure Solution (forms)	**Issue**: Opening feature class forms from Model Explorer takes very long time (~10 min). **Analysis**: Analysis of the Solution configuration indicated that feature class forms were configured to present 100 first rows (generating SQL to fetch data with “rownum<100”). Oracle would use full table scans when SQL contains a pseudo columns. **Recommendation**: Configure Start Mode of feature class forms to ‘View Empty’ (see notes for potential SQL statement to fix the issue for all forms). **Result**: Fix was applied for customer solution. Performance of opening a feature class form from the Model Explorer is very fast.	Slow performance

Solution Tuning (7)

Step in tuning methodology	Analysis and Recommendations	Impact on issues
Configure Solution (display model)	**Issue**: It has been observed that an “empty window” is displayed for 22s-25s when graphics is generated in Map 3D client for the first time. **Analysis**: The team set up environment to troubleshoot (client side: monitoring performance, server side – collecting AWR snapshots and monitoring generated by the Map 3D session’s SQL) all within the 25s window. Tests have been repeated multiple times within 30min interval. Analysis of the ADDM reports highlighted a potential SQL that should be further analyzed / assessed for impacting graphics generation (see notes) **Recommendation**: Information collected during the workshop is not sufficient to produce a reliable assessment. The Team should continue researching this issue. **Result**: This issue is not observed on 2 control instances of Oracle in Autodesk data center (both using customer database), so the issue is considered to be related to the Customer’s Oracle instance. It has been noted that customer Oracle was built using the Oracle’s new multitenant architecture as a multitenant container database (CDB).	Slow generate graphics

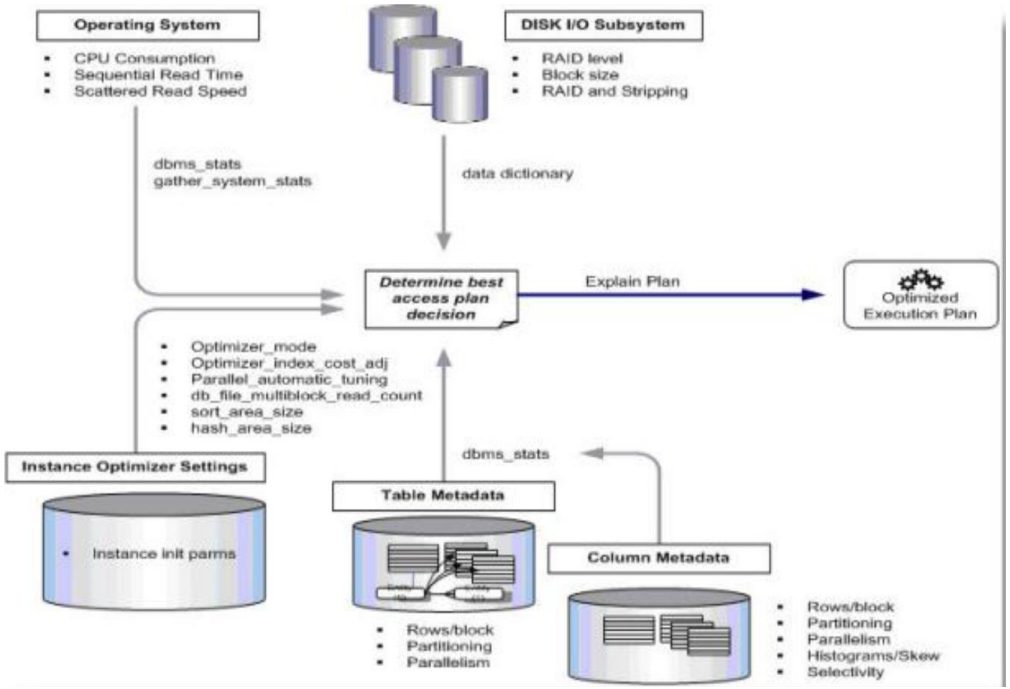
Additional material

Oracle VPD

- VPD implementation defines security or access policies for SQL operations on tables.
- When a user/SQL accesses a table (or view or synonym) which is protected by a VPD policy (function):
  1. The Oracle server invokes the policy function.
  2. The policy function returns a predicate, based on session attributes or database contents.
  3. The server dynamically rewrites the submitted query by appending the returned predicate to the WHERE clause (see example below).

     SELECT "FID","F_CLASS_ID","JOB_VERSION"
     FROM "TB_FEATURE_GROUP_FEATURE" "TB_FEATURE_GROUP_FEATURE" WHERE (
     EXISTS (SELECT 1 FROM TB_JOB_VERSION B WHERE B.JOB_VERSION=TB_FEATURE_GROUP_FEATURE.JOB_VERSION
     AND (B.State <= 1) and b.JOB_OPERATION_ID <> 3 AND NOT EXISTS (SELECT 1 from TB_JOB_VERSION A WHERE B.FID = A.FID and (A.State <= 1) AND A.JOB_OLD_VERSION = B.JOB_VERSION)))
  4. The modified SQL query is executed.
- Data statistics and distribution are critical (i.e. impact the Oracle's CBO execution plans). For example, a typical TB_JOB_VERSION table would have a large number of jobs, but a small number of features associated with each job.

Oracle Optimizer



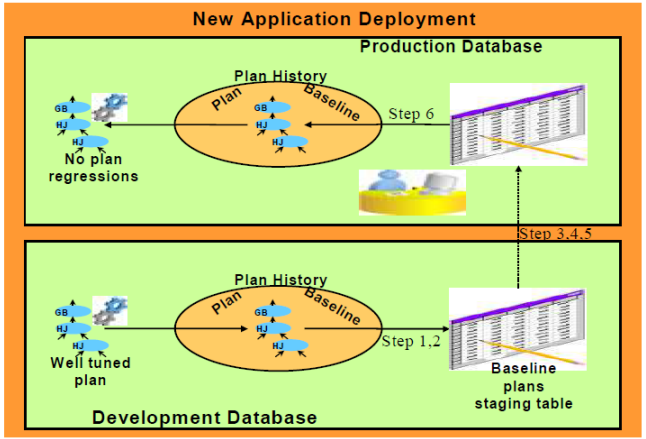
https://jparnitzke.wordpress.com/2009/02/12/oracle-tuning-when-the-sql-can%E2%80%99t-be-modified/

SQL Plan Management in Oracle Database 11g

Introduction

The performance of any database application heavily relies on query execution. While the Oracle optimizer is perfectly suited to evaluate the best possible plan without any user intervention, a SQL statement's execution plan can change unexpectedly, for a variety of reasons including: re-gathering optimizer statistics, changing optimizer parameters or schema/metadata definitions. Not being able to guarantee a plan will change always for the better has lead some customers to freeze their execution plans (Stored Outlines) or lock their optimizer statistics. However, doing so prevents such environments from ever taking advantage of new optimizer functionality or access paths, which would improve the SQL statements performance. Being able to preserve the current execution plan amidst environment changes and allowing changes only for the better would be the ultimate solution.

Oracle Database 11*g* is the first database on the market capable of solving this challenge. SQL Plan Management (SPM) provides a framework for completely transparent controlled execution plan evolution. With SPM the optimizer automatically manages execution plans and ensures only known or verified plans are used. When a new plan is found for a SQL statement it will not be used until it has been verified by the database to have comparable or better performance than the current plan.



Import SQL plan baselines from test when implementing a new application

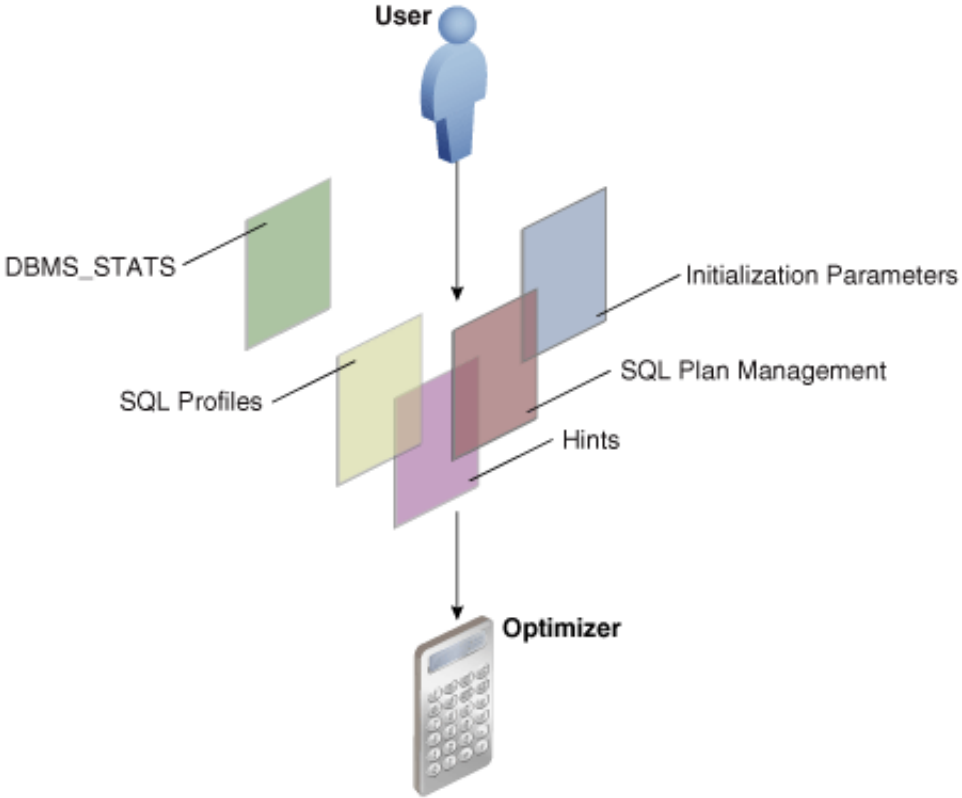
Oracle Performance Tuning Features

- Oracle has designed a performance tuning methodology
- The Oracle Database automatic performance tuning features include:
  - Automatic Workload Repository (AWR) collects, processes, and maintains performance statistics for problem detection and self-tuning purposes.
  - Automatic Database Diagnostic Monitor (ADDM) analyzes the information collected by AWR for possible performance problems with the Oracle database
  - SQL Tuning Advisor allows a quick and efficient technique for optimizing SQL statements without modifying any statements.
  - SQL Access Advisor provides advice on materialized views, indexes, and materialized view logs.
  - End to End Application tracing identifies excessive workloads on the system by specific user, service, or application component.
  - Server-generated alerts automatically provide notifications when impending problems are detected

Techniques for Influencing the Optimizer

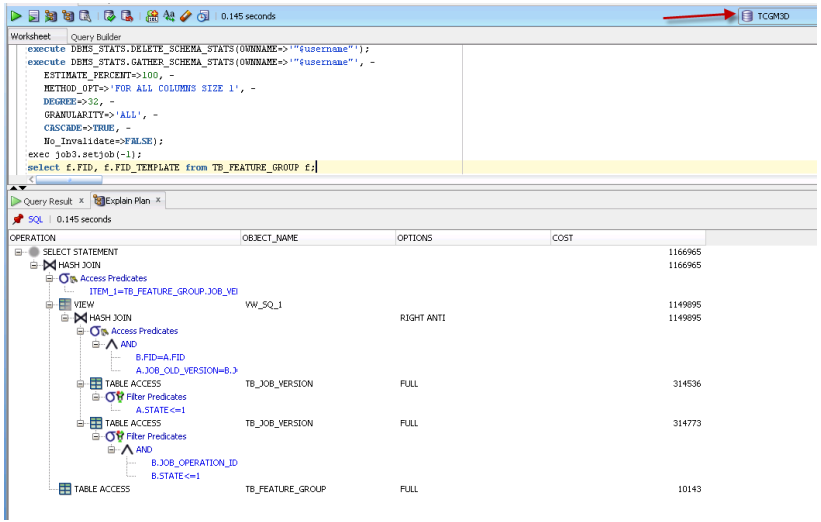
Oracle documentation lists the following:

- Initialization parameters influence many types of optimizer behavior at the database instance and session level
- A **hint** is a commented instruction in a SQL statement.
- DBMS_STATS package updates and manages optimizer statistics
- A **SQL profile** is a database object that contains auxiliary statistics specific to a SQL statement. Conceptually, a SQL profile is to a SQL statement what a set of object-level statistics is to a table or index. A SQL profile can correct suboptimal optimizer estimates discovered during SQL tuning
- **SQL plan management** is a preventative mechanism that enables the optimizer to automatically manage execution plans, ensuring that the database uses only known or verified plans.

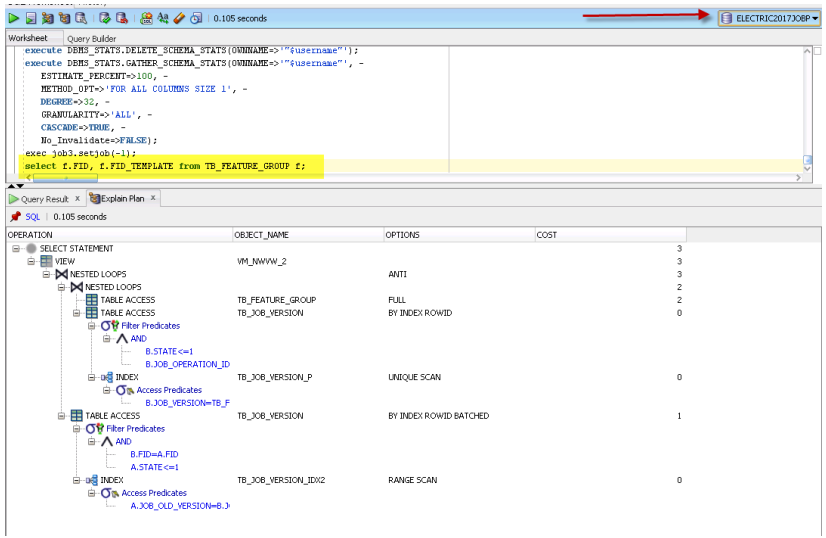


Impact of CBO and execution plans (example)

Customer implementation



M3D COTS Sample



Optimizer Initialization Parameters

(IM important)

OPTIMIZER_INDEX_CACHING *(IM recommendation for 11.2.0.3 = 10)*	Controls the cost analysis of an index probe with a nested loop. The range of values 0 to 100 indicates percentage of index blocks in the buffer cache, which modifies optimizer assumptions about index caching for nested loops and IN-list iterators. A value of 100 infers that 100% of the index blocks are likely to be found in the buffer cache, so the optimizer adjusts the cost of an index probe or nested loop accordingly. Use caution when setting this parameter because execution plans can change in favor of index caching.
OPTIMIZER_INDEX_COST_ADJ *(IM recommendation for 11.2.0.3 = 30)*	Adjusts the cost of index probes. The range of values is 1 to 10000. The default value is 100, which means that the optimizer evaluates indexes as an access path based on the normal cost model. A value of 10 means that the cost of an index access path is one-tenth the normal cost of an index access path. For example, a setting of 50 makes the index access path look half as expensive as normal.
OPTIMIZER_MODE	Sets the optimizer mode at database instance startup. Possible values are ALL_ROWS, FIRST_ROWS_n, and FIRST_ROWS.
OPTIMIZER_INMEMORY_AWARE	This parameter enables (TRUE) or disables (FALSE) all of the in-memory optimizer features, including the cost model for in-memory, table expansion, bloom filters, and so on. Setting the parameter to FALSE causes the optimizer to ignore the in-memory property of tables during the optimization of SQL statements.
OPTIMIZER_USE_INVISIBLE_INDEXES	Enables or disables the use of invisible indexes.

AUTODESK®

Make anything.