The Power of 11g Automatic SQL Tuning

Julian Dontcheff, Nokia, OCM
What is Automatic SQL Tuning?

- Oracle automatically runs the SQL Tuning Advisor on selected high-load SQL statements from the Automatic Workload Repository (AWR) that qualify as tuning candidates.
- This task, called Automatic SQL Tuning, runs in the default maintenance windows on a nightly basis.
- You can customize attributes of the maintenance windows, including start and end time, frequency, and days of the week.
- Automatic SQL tuning runs by default for at most one hour during a maintenance window.
- Automatic SQL Tuning needs one-time configuration for accepting automatically SQL profiles on the high-load SQL.
Why do we need Automatic SQL Tuning?

Suboptimal execution plans are often generated by Oracle:

• One reason is the fact that the cost formula is based on a mathematical model which is nothing more than a model

• Another reason is the lack of correlation information between columns or set of columns (in 11g extended statistics were introduced)
  • Two columns in a table are said to be correlated if the values in the columns do not vary independently
  • Job title and salary are related (the DBA of a company is likely to earn a lot more than the developer does?)
  • Car make and price (Lexus is likely to be a lot more expensive than a Toyota)

• The optimizer needs to make decisions about execution plans in a very short time: not all execution plans can be scanned

• Not possible to create a single histogram on multiple columns

• Oracle bugs
**SQL profiles**

- Oracle 10g allows the optimizer to run in tuning mode where it can gather additional information and make recommendations about how specific statements can be tuned further.
- This process may take several minutes for a single statement so it is intended to be used on high-load resource-intensive statements.
- In comprehensive scope the SQL Tuning Advisor does complete analysis including SQL profiles.
- Profiles can be accepted/implemented only if suggested.
- There is also undocumented way to create profiles using `dbms_sqltune.import_sql_profile`.
- Funny thing is that we rely on the same component to improve the SQL statement that could not generate an optimal plan in the first place 😊.
SQL profiles

• The optimizer may be able to improve performance by gathering additional statistics (data distribution, relations between the columns and joined tables and more useful optimizer information) and altering session specific parameters such as the OPTIMIZER_MODE
• The information is stored in a SQL profile
• If accepted, this information can then be used by the optimizer when running in normal mode
• Unlike a stored outline which fixes the execution plan, a SQL profile may still be of benefit when the contents of the table alter drastically
• Even so, it’s sensible to update profiles periodically (when?)
SQL profiles

• Typically, an accepted SQL profile is associated with the SQL statement through a special SQL signature that is generated using a hash function.

• This hash function normalizes the SQL statement for case sensitivity (changes the entire SQL statement to upper case) and white spaces (removes all extra white spaces) before generating the signature.

• The same SQL profile thus will work for all SQL statements that are essentially the same, where the only difference is in case usage and white spaces.

• By setting force_match to true, the SQL profile will additionally target all SQL statements that have the same text after normalizing literal values to bind variables.
SQL Profiles

Submit the SQL Tuning Advisor task and view the recommendations

Recommendations for SQL ID:0jfuwvtgnkqs5

Only one recommendation should be implemented.

**SQL Text**

```sql
select DF.MARKUP_DEFINITION_ID, DF.IS_PUBLIC, DF.IS_HIDDEN, DF.INTERSECTION_ID, DF.PAGE_LABEL, DF.WEBAPP_NAME, INS.PAGE_DEFINITION_ID, INS.INSTANCE_TITLE, INS.THEME_DEFINITION_ID, LO.LAYOUT_DEFINITION
```

**Select Recommendation**

(Original Explain Plan (Annotated))

<table>
<thead>
<tr>
<th>Select Type</th>
<th>Findings</th>
<th>Recommendations</th>
<th>Rationale</th>
<th>New Benefit (%)</th>
<th>New Explain Plan</th>
<th>Compare Explain Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Profile</td>
<td>A potentially better execution plan was found for this statement.</td>
<td>Consider accepting the recommended SQL profile.</td>
<td></td>
<td>34.93</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SQL Profiles

Tip 1: Accept the SQL profile with force_matching:

```sql
SQL> begin
  2    DBMS_SQLTUNE.ACCEPT_SQL_PROFILE(
  3    task_name => 'SQL_TUNING_1214984590140',
  4    category => 'DEFAULT',
  5    force_match => TRUE);
  6    end;
  7 /

PL/SQL procedure successfully completed.
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>CATEGORY</th>
<th>SIGNATURE</th>
<th>SQL_TEXT</th>
<th>CREATED</th>
<th>LAST_MODIFIED</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>STATUS</th>
<th>FORCE_MAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_SQLPROF_0146b3a6a39bc000</td>
<td>DEFAULT</td>
<td>8.3109E+17</td>
<td>select site_id, id, content_type, type, creator, create_date, logicalpath from v7</td>
<td>28-JUN-08</td>
<td>28-JUN-08</td>
<td></td>
<td>MANUAL</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SQLPROF_0146b8900a7c6c000</td>
<td>DEFAULT</td>
<td>1.7344E+19</td>
<td>select DF.MARKUP.DEFINITION_ID, DF.IS_PUBLIC, DF.IS_HIDDEN, DF.INTERSECTION_ID</td>
<td>02-JUL-08</td>
<td>02-JUL-08</td>
<td></td>
<td>MANUAL</td>
<td>ENABLED</td>
<td>YES</td>
</tr>
</tbody>
</table>
SQL Profiles

• Tip 2: Use the internal tables for viewing the profiles:
  • In 10g the profile data is stored in sqlprof$ and sqlprof$attr
  • In 11g the profile data is stored in sqlobj$ and sqlobj$data
  • In 11g hints are stored in XML format, thus a conversion is necessary to have a readable output

```sql
SQL> SELECT extractValue(value(h),'.') AS hint
  2   FROM sys.sqlobj$data od, sys.sqlobj$ so,
  3   table(xmlsequence(extract(xmltype(od.comp_data),'/outline_data/hint'))) h
  4   WHERE so.name = 'SYS_SQLPROF_011c676326b0006'
  5   AND so.signature = od.signature
  6   AND so.category = od.category
  7   AND so.obj_type = od.obj_type
  8   AND so.plan_id = od.plan_id;
```
### SQL Profiles

The internal hints look like this:

<table>
<thead>
<tr>
<th>HINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGNORE_OPTIM_EMBEDDED_HINTS</td>
</tr>
<tr>
<td>OPTIMIZER_FEATURES_ENABLE(default)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, INDEX_SKIP_SCAN, &quot;0&quot;@&quot;SEL$3&quot;, I_OBJ2, SCALE_ROWS=50.26111767)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, JOIN, (&quot;U&quot;@&quot;SEL$2&quot;, &quot;U&quot;@&quot;SEL$3&quot;, &quot;C&quot;@&quot;SEL$3&quot;), SCALE_ROWS=1.592796321)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, INDEX_SKIP_SCAN, &quot;0&quot;@&quot;SEL$3&quot;, I_OBJ5, SCALE_ROWS=50.26111767)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, JOIN, (&quot;U&quot;@&quot;SEL$2&quot;, &quot;U&quot;@&quot;SEL$3&quot;, &quot;C&quot;@&quot;SEL$3&quot;, &quot;OA&quot;@&quot;SEL$2&quot;), SCALE_ROWS=7.411645275)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, JOIN, (&quot;TPM&quot;@&quot;SEL$2&quot;, &quot;U&quot;@&quot;SEL$2&quot;, &quot;OA&quot;@&quot;SEL$2&quot;), SCALE_ROWS=2.429223569)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, JOIN, (&quot;U&quot;@&quot;SEL$2&quot;, &quot;0&quot;@&quot;SEL$3&quot;), SCALE_ROWS=7.433358831)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$5C160134&quot;, INDEX_SKIP_SCAN, &quot;TPM&quot;@&quot;SEL$2&quot;, I_TABLE_PRIVILEGE_MAP, SCALE_ROWS=26)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_SKIP_SCAN, &quot;02&quot;@&quot;SEL$4&quot;, I_OBJ5, SCALE_ROWS=15554.75196)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_SKIP_SCAN, &quot;02&quot;@&quot;SEL$4&quot;, I_OBJ2, SCALE_ROWS=15554.75196)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_FILTER, &quot;02&quot;@&quot;SEL$4&quot;, I_OBJ2, SCALE_ROWS=0.2806703709)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_SKIP_SCAN, &quot;U2&quot;@&quot;SEL$4&quot;, I_USER2, SCALE_ROWS=5.6e+10)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_FILTER, &quot;02&quot;@&quot;SEL$4&quot;, I_OBJ5, SCALE_ROWS=0.2806703709)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_SKIP_SCAN, &quot;02&quot;@&quot;SEL$4&quot;, I_OBJ1, SCALE_ROWS=15554.75196)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, INDEX_FILTER, &quot;02&quot;@&quot;SEL$4&quot;, I_OBJ1, SCALE_ROWS=0.2806703709)</td>
</tr>
<tr>
<td>OPT_ESTIMATE(&quot;SEL$4&quot;, TABLE, &quot;02&quot;@&quot;SEL$4&quot;, SCALE_ROWS=0.2806703709)</td>
</tr>
</tbody>
</table>
The dictionary tables storing the SQL profiles are restructured to accommodate the storage of SQL plan baselines, which are also SQL management objects.

Further, these dictionary tables are now defined in the SYSAUX tablespace.

Tip 3: Usage of the undocumented init.ora parameter _STN_TRACE allows tracing analysis done by the automatic tuning optimizer.

The output is written to a trace file.

For specific values the output is also written to the ORA_DEBUG_TABLE: it does not exist by default, you will have to create it.
SQL Profiles: ORA_DEBUG_TABLE

CREATE TABLE ora_debug_table (  
time DATE,  
txt0 VARCHAR2(4000), txt1 VARCHAR2(4000),  
txt2 VARCHAR2(4000), txt3 VARCHAR2(4000),  
txt4 VARCHAR2(4000),  
um0 NUMBER, num1 NUMBER, num2 NUMBER,  
um3 NUMBER, num4 NUMBER, num5 NUMBER,  
um6 NUMBER, num7 NUMBER, num8 NUMBER,  
um9 NUMBER  
);
SQL Profiles: _STN_TRACE

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No output has been observed</td>
</tr>
<tr>
<td>2</td>
<td>No output has been observed</td>
</tr>
<tr>
<td>4</td>
<td>SQL text, findings, original execution plan and, optionally, INIT.ORA parameters</td>
</tr>
<tr>
<td>8</td>
<td>Same as 4 and, in addition, adjusted execution plan</td>
</tr>
<tr>
<td>16</td>
<td>SQL text and, optionally, INIT.ORA parameters</td>
</tr>
<tr>
<td>32</td>
<td>Almost same as 16</td>
</tr>
<tr>
<td>64</td>
<td>Verification of statistics</td>
</tr>
<tr>
<td>128</td>
<td>Almost same as 64</td>
</tr>
<tr>
<td>256</td>
<td>No output has been observed</td>
</tr>
<tr>
<td>512</td>
<td>List of recursive queries with execution statistics</td>
</tr>
<tr>
<td>1024</td>
<td>Same as 512 but output written into ORA_DEBUG_TABLE</td>
</tr>
</tbody>
</table>

- Observations thanks to Christian Antognini

NOKIA
11g automatic SQL tuning

• Oracle 11g automatically runs the SQL Tuning Advisor against high impact SQL statements during maintenance windows.

• AWR statistics are used to compile an ordered list of the SQL statements with the greatest performance impact on the system, where the impact is the sum of the CPU and I/O times for the statement during the past week.

• The list excludes statements that are inherently less tunable, such as recently (within a month) tuned recursive statements, parallel queries, DML, DDL and SQL statements whose performance problems are caused by concurrency issues.

• The SQL tuning advisor is run against each statement in turn.

• The outcome may include both SQL profiles and other recommendations.
11g automatic SQL tuning

- STS
- create_tuning_task
- execute_tuning_task
- report_tuning_task
- Implement Recommendations
  - Gather optimizer statistics
  - Create SQL Profile
  - Create index
  - Restructure SQL
11g automatic SQL tuning

- Suggested SQL profiles are performance tested, and those that result in at least a threelfold improvement are accepted if the ACCEPT_SQL_PROFILES parameter is set to TRUE, or reported if it is set to FALSE.
- The accepted SQL profiles are optionally implemented.
- Several factors may prevent SQL profiles from being implemented automatically, including stale optimizer statistics of dependent objects.
- The TYPE column of the DBA_SQL_PROFILES view indicates if SQL profiles are created manually (MANUAL) or automatically (AUTO-TUNE): shown in the next slide.
## 11g automatic SQL tuning

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>SQL_TEXT</th>
<th>CREATED</th>
<th>TYPE</th>
<th>STATUS</th>
<th>FOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_SGPROF_011f20d902ad0007</td>
<td>7.0461E+18</td>
<td>/* OracleCEM */ select distinct grants.table_name, 'SYS_PUBLIC_PACKAGE'</td>
<td>31-JAN-09 06:01:19.000000 AM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011a634e34320002</td>
<td>1.9014E+18</td>
<td>SELECT LOG_ID, PID, SESSION_ID, PAGE_COUNT, QUERY_STRING, POST_PARAMETERS, APPLI</td>
<td>13-JUN-08 10:01:57.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011c676326b00006</td>
<td>5.3907E+18</td>
<td>select /*+ RULE */ TABLE_NAME from all_tab_prms where table_name like 'DBA_%' a</td>
<td>15-SEP-08 10:01:03.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_0121889b0ec10000</td>
<td>1.7258E+19</td>
<td>/* OracleCEM */ SELECT end_time, stat_us, session_key, session_recid, session_</td>
<td>28-MAY-09 10:04:20.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011c33e5cd0b0005</td>
<td>5.9401E+18</td>
<td>select tabs.table_name, 'ICO', tabs.cluster_name, p stitioned, iot_type</td>
<td>05-SEP-08 10:03:30.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_0120d33565ce0001</td>
<td>8.7290E+18</td>
<td>SELECT dat.sample_time, narr(dat.wait_class, 'CPU Used') wait_class, dat.session_</td>
<td>22-APR-09 10:00:31.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011a655325d000B</td>
<td>6.4886E+18</td>
<td>SELECT TASK_LIST.TASK_ID FROM (SELECT /*+ NO_MERGE(T) */ ORDERED D */ T.TASK_ID FROM</td>
<td>06-JUL-08 06:03:53.000000 AM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011fbb13e9c30000</td>
<td>3.3255E+18</td>
<td>SELECT d_REPORT_10B.NAME, d_REPORT_10B.REGION, d_REPORT_10B.SITE, d_REPO</td>
<td>28-FEB-09 06:10:56.000000 AM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011a79099a90000</td>
<td>1.7581E+19</td>
<td>SELECT * FROM WMX_VI SIT_ENTRY_EXIT WEE WHERE EXISTS (SELECT 1 FROM WMX_VISIT SU</td>
<td>11-JUN-08 10:11:12.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
<tr>
<td>SYS_SGPROF_011a7910b3f70001</td>
<td>4.6939E+18</td>
<td>SELECT * FROM WMX_VI SIT_SESSION WVS WHERE EXISTS (SELECT 1 FROM WMX_VISIT SUMMAR</td>
<td>11-JUN-08 10:18:35.000000 PM</td>
<td>AUTO</td>
<td>ENABLED</td>
<td>NO</td>
</tr>
</tbody>
</table>
11g automatic SQL tuning

- The 3 main tunable parameters related to automatic SQL tuning
- Defaults are FALSE, 20 and 10000

```sql
SQL> SELECT parameter_name, parameter_value
2   FROM dba_advisor_parameters
3   WHERE task_name = 'SYS_AUTO_SQL_TUNING_TASK'
4   AND parameter_name IN ('ACCEPT_SQL_PROFILES',
5                        'MAX_SQL_PROFILES_PER_EXEC',
6                        'MAX_AUTO_SQL_PROFILES');

PARAMETER_NAME                      PARAMETER_VALUE
----------------------------------------------------------
ACCEPT_SQL_PROFILES                   TRUE
MAX_SQL_PROFILES_PER_EXEC             20
MAX_AUTO_SQL_PROFILES                 10000
```

- EXECUTION_DAYS_TO_EXPIRE: Specifies the number of days for which to save the task history in the advisor framework schema. By default, the task history is saved for 30 days before it expires
11g automatic SQL tuning

• Tip 4: use the SET_TUNING_TASK_PARAMETER procedure of the DBMS_SQLTUNE package in order to control the behavior of the SQL tuning advisor

• ACCEPT_SQL_PROFILES - Automatically accept SQL profiles (default FALSE): set it always to TRUE

• MAX_SQL_PROFILES_PER_EXEC - The maximum number of SQL profiles automatically implemented per run (default 20)

• MAX_AUTO_SQL_PROFILES - The maximum number of automatic SQL profiles allowed on the system (default 10000)
11g automatic SQL tuning

• To disable automatic SQL tuning, use the DISABLE procedure in the DBMS_AUTO_TASK_ADMIN package.

```sql
BEGIN
    DBMS_AUTO_TASK_ADMIN.DISABLE(
        client_name => 'sql tuning advisor',
        operation => NULL,
        window_name => NULL);
END;
/
```

• To re-enable automatic SQL tuning, use the ENABLE procedure in the DBMS_AUTO_TASK_ADMIN package.

```sql
BEGIN
    DBMS_AUTO_TASK_ADMIN.ENABLE(
        client_name => 'sql tuning advisor',
        operation => NULL,
        window_name => NULL);
END;
/
```

• Setting the STATISTICS_LEVEL parameter to BASIC will disable automatic statistics gathering by the AWR and, as a result, also disable automatic SQL tuning.
11g automatic SQL tuning

- Why disable automatic SQL tuning?
- Still there are unfixed bugs: sql too large to fit into the window
- WARNING: Could not set the asynch I/O limit to 2 for SQL direct I/O. It is set to 0

<table>
<thead>
<tr>
<th>Severity</th>
<th>Category</th>
<th>Time</th>
<th>Alert Log Error Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>❌</td>
<td>Generic Alert Log Error</td>
<td>Aug 18, 2009 11:00:12 PM</td>
<td>SYS_AUTO_SQL_TUNING_TASK exiting with error &quot;1760&quot; for execution &quot;EXEC_7681&quot;. See DBA_ADVISOR_EXECUTIONS for more details. End automatic SQL Tuning Advisor run for special tuning task &quot;SYS_AUTO_SQL_TUNING_TASK&quot;. Errors in file /rman/oracle11/diag/rdbms/rman1o/RMAN10/trace/RMAN10_j002_3432.trc: ORA-12012: error on auto execute of job 63907 ORA-01760: illegal argument for function ORA-06512: at &quot;SYS.PRVT_ADVISOR&quot;, line 2677 ORA-06512: at &quot;SYS.DBMS_ADVISOR&quot;, line 241 ORA-06512: at &quot;SYS.DBMS_SQLTUNE&quot;, line 702 ORA-06512: at line 4</td>
</tr>
</tbody>
</table>
11g automatic SQL tuning

- Workaround: none

<table>
<thead>
<tr>
<th>Bug No.</th>
<th>8713800</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filed</td>
<td>21-JUL-2009</td>
</tr>
<tr>
<td>Product</td>
<td>Oracle Server - Enterprise Edition</td>
</tr>
<tr>
<td>Platform</td>
<td>Sun Solaris SPARC (64-bit)</td>
</tr>
<tr>
<td>Database Version</td>
<td>11.1.0.7.0</td>
</tr>
<tr>
<td>Severity</td>
<td>Severe Loss of Service</td>
</tr>
<tr>
<td>Base Bug</td>
<td>N/A</td>
</tr>
<tr>
<td>Updated</td>
<td>23-JUL-2009</td>
</tr>
<tr>
<td>Product Version</td>
<td>11.1.0.7.0</td>
</tr>
<tr>
<td>Platform Version</td>
<td>10</td>
</tr>
<tr>
<td>Affects Platforms</td>
<td>Generic</td>
</tr>
<tr>
<td>Status</td>
<td>Description Phase</td>
</tr>
<tr>
<td>Fixed in Product Version</td>
<td>No Data</td>
</tr>
</tbody>
</table>

Problem statement:

SQL TUNING ADVISOR JOB FAILS WITH ORA-1760
11g automatic SQL tuning

- The steps performed by Oracle Database during the automatic SQL tuning process
DBMS_SQLTUNE reports

- The REPORT_AUTO_TUNING_TASK function of the DBMS_SQLTUNE package returns a CLOB containing a report from the specified automatic tuning task.
- Information found in the CLOB report:
  - General information - High-level information about the SQL tuning task.
  - Summary - A summary of the SQL statements tuned during the task, including the estimated benefit associated with the tuning operation.
  - Tuning finding - Information about findings, acceptance of the profile, implementation of the profile, and detailed execution statistics for each analyzed statement.
  - Explain plans - The old and new execution plans for each analyzed statement.
  - Errors - Any errors encountered during the task.
DBMS_SQLTUNE reports

SQL> PRINT :l_report

L_REPORT
---------------------------------------------------------------
GENERAL INFORMATION SECTION
---------------------------------------------------------------
Tuning Task Name : SYS_AUTO_SQL_TUNING_TASK
Tuning Task Owner : SYS
Workload Type : Automatic High-Load SQL Workload
Execution Count : 30
Current Execution : EXEC_1663
Execution Type : TUNE_SQL
Scope : COMPREHENSIVE
Global Time Limit(seconds) : 3600
Per-SQL Time Limit(seconds) : 1200
Completion Status : COMPLETED
Started at : 07/08/2009 22:00:02
Completed at : 07/08/2009 22:00:19
Number of Candidate SQLs : 4
Cumulative Elapsed Time of SQL (s) : 56
How to group profiles

• SQL profiles are grouped in categories
• The initial category is defined when profiles are accepted
• The default category is called DEFAULT: it means that all user sessions where the SQLTUNE_CATEGORY initialization parameter is set to DEFAULT can use the profile
• The active category is defined with the init.ora parameter SQLTUNE_CATEGORY (dynamic)
• Profiled can be moved from one category to another
• Only one category can be active
• Is that really needed 😊 Probably: By altering the category of a SQL profile, you can determine which sessions are affected by the creation of a profile
Managing profiles

- Use the DBMS_SQLTUNE package to manage profiles
- ACCEPT_SQL_PROFILE
- ALTER_SQL_PROFILE: change status, name, description or category
- DROP_SQL_PROFILE
- CREATE_STGTAB_SQLPROF: create the staging table for the profiles
- PACK_STGTAB_SQLPROF: copy the profiles from the DD to the table
- REMAP_STGTAB_SQLPROF: change name/category in the staging table
- UNPACK_STGTAB_SQLPROF: copy the profiles from the table to the DD
Privileges for SQL profiles

- No object privileges exist
- The system privilege ADVISOR is needed for using the SQL Tuning Advisor
- For SQL profiles the following exist:
  - CREATE ANY SQL PROFILE
  - DROP ANY SQL PROFILE
  - ALTER ANY SQL PROFILE
Tip 5: note that in order to use automatic SQL tuning and in particular SQL profiles the Tuning and the Diagnostics packs need to be licensed!

Automatic SQL tuning via profiles is the most efficient way to massively tune SQL statements: CBO learns from its mistakes 😊

One of the best new features in 11g

Best 11g new feature for Data Center DBAs
Hi,

After 11g upgrade query 1 takes 46min (previously timed out in Toad) and query 2 takes 17min (previously 50min).

Explain plans now.