Delivering Oracle Success

Automatic SQL Tuning in Oracle Database 10g and 11g

Lucy Feng

RMOUG Training Days
February 15-17, 2011
About DBAK

- Oracle solution provider
- Co-founded in 2005
- Based in Englewood, CO
- 2008 “Emerging Business of the Year” – South Metro Denver Chamber of Commerce
- More than 130 implementations, upgrades, conversions, and support projects for 85+ clients
- Average 15 years of Oracle expertise
- Oracle Gold Partner
Automatic SQL Tuning

• SQL Tuning Advisor -- 10G
The optimizer can operate in two modes:

  Normal mode - uses statistics to generate execution plans
  Tuning mode – carries out in-depth analysis to come up with ways to optimize execution plans

• SQL Tuning Advisor – 11G
  New automations
Automatic SQL Tuning Advisor

• Automatically identifying problematic SQL
  – Uses AWR to identify SQL candidates
    Top queries for the past week
    Top queries for any day in the past week
    Top queries for any hour
    Top queries by average single execution
    They are combined into a single set and are weighted

  – Targets repeating SQL with a significant impact on database

• Recursive statements or ad hoc queries are not selected
Automatic SQL Tuning Advisor

• Automatically invoking SQL Tuning Advisor
  – Statistics analysis - determines freshness of statistics
  – SQL profiling
  – Access path analysis - may recommend additional indexing
  – SQL structure analysis - may recommend changes to the structure of the SQL statement

• Automatically testing and implementing SQL Profiles (Optional)
Automatic SQL Tuning Advisor

• Implementing a SQL profile when performance improvement is at least threefold (cpu + I/O cost saving; neither becomes worse), provided accept_sql_profiles task parameter is set to TRUE.

```
SELECT parameter_name, parameter_value
FROM   dba_advisor_parameters
WHERE  task_name = 'SYS_AUTO_SQL_TUNING_TASK'

PARAMETER_NAME        PARAMETER_VALUE
ACCEPT_SQL_PROFILES   TRUE
```
Automatic SQL Tuning Advisor

SYS_AUTO_SQL_TUNING_TASK by default runs in the maintenance windows for no more than one hour.

-- Disable automatic SQL tuning advisor
exec DBMS_AUTO_TASK_ADMIN.DISABLE(
    client_name => 'sql tuning advisor',
    operation => NULL,
    window_name => NULL);

-- Enable it for Saturday maintenance windows
window_name => 'SATURDAY_WINDOW'
Automatic SQL Tuning Advisor

-- Change automatic SQL Tuning Advisor time limit

exec DBMS_SQLTUNE.set_tuning_task_parameter(
    task_name => 'SYS_AUTO_SQL_TUNING_TASK',
    parameter => 'TIME_LIMIT',
    value     => 14400);
Automatic SQL Tuning Advisor

-- Enable automatic SQL profile implementation
exec DBMS_SQLTUNE.set_tuning_task_parameter(
  task_name => 'SYS_AUTO_SQL_TUNING_TASK',
  parameter => 'ACCEPT_SQL_PROFILES',
  value => 'TRUE');

Other parameters you can set:

<table>
<thead>
<tr>
<th>PARAMETER_NAME</th>
<th>PARAMETER_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_SQL_PROFILES_PER_EXEC</td>
<td>20</td>
</tr>
<tr>
<td>MAX_AUTO_SQL_PROFILES</td>
<td>10000</td>
</tr>
</tbody>
</table>
### Automatic SQL Tuning Result Details

**Begin Date**: Nov 15, 2009 8:46:51 AM (UTC-07:00)  
**End Date**: Nov 16, 2009 1:26:18 PM (UTC-07:00)

**Recommendations**

Only profiles that significantly improve SQL performance were implemented.

<table>
<thead>
<tr>
<th>Select</th>
<th>SQL Text</th>
<th>Parsing Schema</th>
<th>SQL ID</th>
<th>Statistics</th>
<th>SQL Profile</th>
<th>Index</th>
<th>Restructure SQL</th>
<th>Miscellaneous</th>
<th>Error</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
<td><code>CREATE_INDEX (SELECT ...)</code></td>
</tr>
</tbody>
</table>

**Legend**

- ✓ Recommended
- ✓ Implemented
SQL Profile

- Among SQL Tuning Advisor recommendations, SQL Profile is the only one that can be automatically accepted.
- The scope of a SQL Profile is limited to an individual SQL statement.
- With 11g Automatic SQL Tuning Advisor, more SQL Profiles will be created and put to use.
SQL Profile

- The optimizer under normal mode makes estimates about cardinality, selectivity and cost that can be off significantly sometimes.
- SQL Tuning Advisor uses three methods to verify its estimates.
  - Dynamic data sampling – uses a data sample and applies correction
  - Partial execution – checks whether a plan is the best possible plan
  - Past execution history statistics
- Stores corrections/adjustments in SQL Profiles.
SQL Profile

--Find out what adjustments are stored in a SQL Profile in 10g:

```sql
SELECT attr_val
FROM sys.sqlprof$ p, sys.sqlprof$attr a
WHERE p.sp_name = '&sql_profile_name'
AND p.signature = a.signature
AND p.category = a.category;
```
SQL Profile

-- Find out what adjustments are stored in a SQL Profile in 11g:

```
SELECT extractValue(value(h),'.') AS hint
FROM sys.sqlobj$data od, sys.sqlobj$ so,
table(xmlsequence(extract(xmltype(od.comp_data),'/outline_data/hint'))) h
WHERE so.name = '&sql_profile_name'
AND so.signature = od.signature
AND so.category = od.category
AND so.obj_type = od.obj_type
AND so.plan_id = od.plan_id;
```
Examples of Hints Used in SQL Profiles

• Setting optimizer mode
  \texttt{ALL\_ROWS}

• Disabling hints embedded in SQL
  \texttt{IGNORE\_OPTIM\_EMBEDDED\_HINTS}

• Setting \texttt{optimizer\_features\_enable}
  \texttt{OPTIMIZER\_FEATURES\_ENABLE}(default)
Examples of Hints Used in SQL Profiles

- Adjusting the number of rows returned from a table
  ```sql
  OPT_ESTIMATE("SEL$AD385D36", TABLE, "T"."SEL$5", SCALE_ROWS=32.94409938)
  ```

- Adjusting the number of rows returned from an index scan
  ```sql
  OPT_ESTIMATE("SEL$AD385D36", INDEX_SCAN, "B"."SEL$5", FND_ID_FLEX_STRUCTURES_U2, SCALE_ROWS=32.94409938)
  ```

- Adjusting the number of rows returned from a join
  ```sql
  OPT_ESTIMATE("SEL$A05B8819", JOIN, ("INV"."SEL$6", "VNDR"."SEL$6", "VNDR_SC"."SEL$6"), SCALE_ROWS=15773.78428)
  ```
Examples of Hints Used in SQL Profiles

• Auxiliary statistics
  For a table
  `TABLE_STATS("CER_ADMIN"."EMAIL", scale, blocks=7, rows=1460)`
  For a column
  `COLUMN_STATS("CER_ADMIN"."EMAIL","EMAIL_ID", scale, length=5)`
  For an index
  `INDEX_STATS("CER_ADMIN"."EMAIL","EMAIL_PK", scale, blocks=10, index_rows=1860)`
SQL Profile Advantages

- No change to application code. Transparent to user. Ideal for tuning packaged application.
- Specific to an individual SQL statement. No wider impact.
- Does not freeze the execution plan. A SQL profile is different from a stored outline.
A Word of Caution

• When a database is configured correctly, the normal mode of the optimizer generates a reasonable plan for the vast majority of SQL statements. A small percentage of statements need additional help from the SQL Tuning Advisor.

• It is time and resource intensive to invoke SQL Tuning Advisor. It is meant to be used for complex and high-load SQL statements that have non-trivial impact on the system.
Proactive SQL Tuning in 10G

• Automatic SQL Tuning Advisor is a step towards proactive SQL Tuning.

• In 10g, you can simulate automatic SQL Tuning with programming using DBMS_SQLTUNE package.

• DBMS_SQLTUNE is the interface to SQL tuning.
Proactive SQL Tuning in 10G

• The speaker developed a custom package that uses DBMS_SQLTUNE

```sql
CREATE OR REPLACE PACKAGE AUTO_TUNE_PKG IS
    PROCEDURE CREATE_STS
        (p_sqlset_name1      in varchar2 DEFAULT 'TOP_SQL'
        ,p_sqlset_owner      in varchar2
        ,p_ranking_measure   in varchar2 DEFAULT 'ELAPSED'
        ,p_sql_count         in number DEFAULT 5
        );
    PROCEDURE EXEC_TUNE_TASK
        (p_sqlset_name1      in varchar2
        ,p_sqlset_owner      in varchar2
        );
END AUTO_TUNE_PKG;
/
```
DBMS_SQLTUNE vs. OEM

- Oracle Enterprise Manager provides the easier way to access SQL Tuning Advisor.
- Use DBMS_SQLTUNE when
  - OEM is not available
  - Scripting can save you time if you get into the habit of proactively looking for top queries to tune.
  - Features are not available through OEM.
License and Privileges

- Oracle Diagnostic Pack and Oracle Tuning pack must be licensed, whether you invoke SQL Tuning Advisor through OEM or from the APIs

- Use SQL Tuning Advisor
  - ADVISOR privilege
- Create, drop and modify an SQL Profile
  - CREATE ANY SQL PROFILE
  - DROP ANY SQL PROFILE
  - ALTER ANY SQL PROFILE
  - A single privilege “ADMINISTER SQL MANAGEMENT OBJECT” in 11g
- Manage SQL Tuning Sets
  - ADMINISTER SQL TUNING SET or ADMINISTER ANY SQL TUNING SET
DBMS_SQLTUNE Subprogram Groups – 10GR2

- SQL Tuning Advisor Subprograms
- SQL Profile Subprograms
- SQL Tuning Set Subprograms
Create a Tuning Task

Use DBMS_SQLTUNE.CREATE_TUNING_TASK to
• create a tuning task for a single statement using its text, for a single statement from the cursor cache or from the workload repository given the sql_id.
• Or create a tuning task for a SQL tuning set.
Create a Tuning Task from a SQL Text

DECLARE
  l_task_name VARCHAR2(30);
  l_sqltext CLOB;
BEGIN
  l_sqltext := 'SELECT FROM employees e, locations l, departments d ' ||
    'WHERE e.department_id = d.department_id AND ' ||
    'l.location_id = d.location_id AND ' ||
    'e.employee_id < :bnd';

  l_task_name := DBMS_SQLTUNE.CREATE_TUNING_TASK(
    sql_text => l_sqltext,
    bind_list => sql_binds(anydata.ConvertNumber(100)),
    user_name => 'HR',
    scope => 'COMPREHENSIVE',
    time_limit => 60,
    task_name => 'my_sql_tuning_task');
END;
/

© DBAK 2011
Execute a Tuning Task

BEGIN
    DBMS_SQLTUNE.EXECUTE_TUNING_TASK(
        task_name => 'my_sql_tuning_task'
    );
END;
/

Display the Results of a SQL Tuning Task

SET LONG 1000
SET LONGCHUNKSIZE 1000
SET LINESIZE 100
SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK(
   'my_sql_tuning_task')
FROM DUAL;
Create a SQL Profile

DBMS_SQLTUNE.ACCEPT_SQL_PROFILE (   
    task_name IN VARCHAR2,   
    object_id IN NUMBER := NULL,   
    name IN VARCHAR2 := NULL,   
    description IN VARCHAR2 := NULL,   
    category IN VARCHAR2 := NULL,   
    task_owner IN VARCHAR2 := NULL, 
    replace IN BOOLEAN := FALSE, 
    force_match IN BOOLEAN := FALSE); 

exec DBMS_SQLTUNE.ACCEPT_SQL_PROFILE( task_name => 'my_sql_tuning_task' );
SQL Profile Category

- Default category is DEFAULT
- ACCEPT_SQL_PROFILE or ALTER_SQL_PROFILE can change the category of a SQL profile.
- The active category is set through INIT parameter SQLTUNECATEGORY
  Can be altered at system or session level.
SQL Tuning Set

• DBMS_SQLTUNE provides more flexibility compared to OEM when tuning multiple SQL statements.

• To tune multiple SQL statements first capture them in a SQL tuning set
## SQL Tuning Set Input Sources

<table>
<thead>
<tr>
<th>SQL query</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SELECT_CURSOR_CACHE</code></td>
<td>Select cursor cache input sources</td>
</tr>
<tr>
<td><code>SELECT_WORKLOAD_REPOSITORY</code></td>
<td>Select workload repository input sources</td>
</tr>
<tr>
<td><code>SELECT_SQLSET</code></td>
<td>Select SQL set input sources</td>
</tr>
</tbody>
</table>

**SELECT_CURSOR_CACHE**
- basic_filter
- object_filter
- ranking_measure1
- ranking_measure2
- ranking_measure3
- result_percentage
- result_limit
- attribute_list

**SELECT_WORKLOAD_REPOSITORY**
- begin_snap
- end_snap
- basic_filter
- object_filter
- ranking_measure1
- ranking_measure2
- ranking_measure3
- result_percentage
- result_limit
- attribute_list

**SELECT_SQLSET**
- sqlset_name
- basic_filter
- object_filter
- ranking_measure1
- ranking_measure2
- ranking_measure3
- result_percentage
- result_limit
- attribute_list
- plan_filter
- sqlset_owner
SQL Tuning Set Input Sources - continued

• Another source is a user-defined workload.
• In 11g, you can get the input from one or more trace files using the new function “SELECT_SQL_TRACE.”
Basic Filter

- desc sqlset_row

<table>
<thead>
<tr>
<th>sql_id</th>
<th>disk_reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sql_text</td>
<td>direct_writes</td>
</tr>
<tr>
<td>parsing_schema_name</td>
<td>rows_processed</td>
</tr>
<tr>
<td>module</td>
<td>fetches</td>
</tr>
<tr>
<td>action</td>
<td>executions</td>
</tr>
<tr>
<td>elapsed_time</td>
<td>optimizer_cost</td>
</tr>
<tr>
<td>cpu_time</td>
<td>buffer_gets</td>
</tr>
</tbody>
</table>
Create a SQL Tuning Set

BEGIN
    dbms_sqltune.create_sqlset (sqlset_name => 'PEAK_LOAD',
                               description => 'peak work load',
                               sqlset_owner => 'LFENG');
END;
/

© DBAK 2011
Select SQL from Cursor Cache

DECLARE
    cur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
    OPEN cur FOR
    select value(P)
    from table(DBMS_SQLTUNE.SELECT_CURSOR_CACHE(
        'PARSING_SCHEMA_NAME not in ("SYS")',
        null,
        'DISK_READS',
        null,
        null,
        1,
        10,
        'ALL')) P;
Populate the SQL Tuning Set

```sql
DBMS_SQLTUNE.LOAD_SQLSET(
    sqlset_name  =>'PEAK_LOAD',
    populate_cursor=>cur,
    sqlset_owner  =>'LFENG'
);

CLOSE cur;
END;
/
```
Create SQL Tuning Sets – more examples

-- select sql starting with 'select /*MY_CRITICAL_SQL*/%'

    DECLARE
cur dbms_sqltune.sqlset_cursor;
BEGIN
OPEN stscur FOR
SELECT VALUE(P)
FROM TABLE(dbms_sqltune.select_cursor_cache(  
'sql_text like "select /*MY_CRITICAL_SQL*/%"'',  
null, null, null, null, null, null, 'ALL')) P;

    dbms_sqltune.load_sqlset(sqlset_name => 'TOP_STS',  
    populate_cursor => cur,  
    sqlset_owner => 'LFENG');
END;
/


Create SQL Tuning Sets – more examples

declare
cur DBMS_SQLTUNE.SQLSET_CURSOR;
begin
open cur for
select VALUE(p) from
table(DBMS_SQLTUNE.SELECT_WORKLOAD_REPOSITORY(22853, 23034,
'BUFFER_GETS>10000 AND parsing_schema_name <> "SYS"',NULL,NULL,NULL,NULL,NULL,30,'ALL'))
p;

DBMS_SQLTUNE.LOAD_SQLSET('TOP_STS', cur);
end;
/
Verify a SQL Tuning Set

-- Verify SQL statements in STS.

```sql
select sql_id, substr(sql_text,1, 15) text
from dba_sqlset_statements
where sqlset_name = 'PEAK_LOAD'
order by sql_id;
```

-- Verify the execution Plan of a SQL_ID in the STS for an user sql

```sql
SELECT * FROM table (DBMS_XPLAN.DISPLAY_SQLSET('PEAK_LOAD','dmqch2g6rtvzf'));
```
Specify an Order for Tuning Statements

EXEC :sts_task :=
DBMS_SQLTUNE.CREATE_TUNING_TASK(
sqlset_name => 'my_workload',
rank1 => 'BUFFER_GETS',
time_limit => 3600,
description => 'tune my workload ordered by buffer gets');
SQL Profile Force Match

DBMS_SQLTUNE.ACCEPT_SQL_PROFILE (  
task_name IN VARCHAR2,  
object_id IN NUMBER := NULL,  
name IN VARCHAR2 := NULL,  
description IN VARCHAR2 := NULL,  
category IN VARCHAR2 := NULL,  
task_owner IN VARCHAR2 := NULL,  
replace IN BOOLEAN := FALSE,  
force_match IN BOOLEAN := FALSE);

force_match true: Substitutes literals with bind variables
Transport SQL Tuning Sets

CREATE_STGTAB_SQLSET
  – creates a staging table
PACK_STGTAB_SQLSET
  – copies SQL tuning sets to the staging table

Move staging table from source to destination

UNPACK_STGTAB_SQLSET
  -- copies SQL tuning sets from staging table to
  the SQL tuning sets schema
Transport a SQL Tuning Set

-- In the source database
    BEGIN
    DBMS_SQLTUNE.CREATE_STGTAB_SQLSET (
        table_name => 'STS_TAB',
        schema_name => 'LFENG',
        tablespace_name => 'USERS');

    DBMS_SQLTUNE.PACK_STGTAB_SQLSET (
        sqlset_name => 'PEAK_LOAD',
        sqlset_owner => 'LFENG',
        staging_table_name => 'STS_TAB',
        staging_schema_owner => 'LFENG');
    END;
    /
Transport a SQL Tuning Set

-- in the destination database
BEGIN
DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET(
  sqlset_name =>'PEAK_LOAD',
  sqlset_owner =>'LFENG',
  replace =>TRUE,
  staging_table_name =>'STS_TAB',
  staging_schema_owner =>'LFENG'
);
END;
/

© DBAK 2011
Transport SQL Profiles

CREATE_STGTAB_SQLPROF
  – creates a staging table

PACK_STGTAB_SQLPROF
  – copies SQL profiles to the staging table

Move staging table from source to destination

UNPACK_STGTAB_SQLPROF
  -- copies profile data in the staging table to create profiles on the system
CAPTURE_CURSOR_CACHE_SQLSET procedure

- Collects SQL statements from the cursor cache over a specified time interval, building a realistic system workload.

- **CAPTURE_CURSOR_CACHE_SQLSET** (
  sqlset_name IN VARCHAR2,
  time_limit IN POSITIVE := 1800,
  repeat_interval IN POSITIVE := 300,
  capture_option IN VARCHAR2 := 'MERGE',
  capture_mode IN NUMBER := MODE_REPLACE_OLD_STATS,
  basic_filter IN VARCHAR2 := NULL,
  sqlset_owner IN VARCHAR2 := NULL);
CAPTURE_CURSOR_CACHE_SQLSET
Example

EXEC DBMS_SQLTUNE.CAPTURE_CURSOR_CACHE_SQLSET ( sqlset_name =>'MY_WORKLOAD', time_limit =>300, repeat_interval =>10, capture_option =>'INSERT');
Creative Use of DBMS_SQLTUNE

- Capture SQL in a SQL tuning set and use STS as a repository of SQL and explain plans
- Compare SQL Tuning Advisor recommendations for critical queries to get a preview of the performance impact of an upgrade
DBMS_SQLTUNE Documentation

• “PL/SQL Packages and Types Reference” has a complete description of subprograms.
• “Performance Tuning Guide” has many examples.
Q & A
Contact

Lucy Feng
lfeng@dbaknow.com

www.dbaknow.com