

# DBA Knowledge

## Automatic SQL Tuning

Presented by Lucy Feng



# Automatic SQL Tuning in 11G

- SQL Tuning Advisor -- 10G
  - The optimizer can operate in two modes:
    - Normal mode
    - Tuning mode
- SQL Tuning Advisor – 11G
  - New automations

# Automatic SQL Tuning Advisor (1)

- Automatically identifying problematic SQL
  - Analyzing statistics in AWR
  - Targeting repeating SQL with a significant impact on database
- Automatically invoking SQL Tuning Advisor
  - Statistics analysis      SQL profiling
  - Access path analysis    SQL structure analysis
- Automatically implementing SQL Profiles (Optional)

# Automatic SQL Tuning Advisor (2)

- Implementing an SQL profile when performance improvement is at least threefold, if `accept_sql_profiles` task parameter is TRUE.

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

| <u>PARAMETER_NAME</u> | <u>PARAMETER_VALUE</u> |
|-----------------------|------------------------|
| ACCEPT_SQL_PROFILES   | TRUE                   |

## Automatic SQL Tuning Advisor (3)

SYS\_AUTO\_SQL\_TUNING\_TASK by default runs in the maintenance windows for no more than one hour.

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

Enable it for Saturday maintenance windows:

```
  window_name => 'SATURDAY_WINDOW'
```

# Automatic SQL Tuning Advisor (4)

- Enable automatic SQL profile implementation

**BEGIN**

```

DBMS_SQLTUNE.set_tuning_task_parameter(
  task_name => 'SYS_AUTO_SQL_TUNING_TASK',
  parameter => 'ACCEPT_SQL_PROFILES',
  value      => 'TRUE');

```

**END;**

**/**

| <u>PARAMETER_NAME</u>     | <u>PARAMETER_VALUE</u> |
|---------------------------|------------------------|
| ACCEPT_SQL_PROFILES       | FALSE                  |
| MAX_SQL_PROFILES_PER_EXEC | 20                     |
| MAX_AUTO_SQL_PROFILES     | 10000                  |

Recommendations for SQL ID:948kryk9vu0bm

[Return](#)

Only one recommendation should be implemented.

SQL Text

`select /*+ USE_NL(s c) FULL(s) FULL(c) AST */ c.cust_id, sum(s.quantity_sold) from sh.sales s, sh.customers c where s.cust_id = c.cust_id and c.cust_id > 200000 group by c.cust_id`

Select Recommendation

[Original Explain Plan \(Annotated\)](#)

[Implement](#)

| Select                           | Type        | Findings  | Recommendations  | Rationale  | New Benefit (%) | Explain Plan            | Compare Explain Plans   |
|----------------------------------|-------------|---|--|--|-----------------|-------------------------|-------------------------|
| <input checked="" type="radio"/> | SQL Profile | A potentially better execution plan was found for this statement.                     | Consider accepting the recommended SQL profile.  |  | 97.44           | <a href="#">Explain</a> | <a href="#">Compare</a> |
| <input type="radio"/>            | Index       | The execution plan of this statement can be improved by creating one or more indices. | Consider running the Access Advisor to improve the physical schema design or creating the recommended index. SH.SALES("CUST_ID") | Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption. | 83.59           | <a href="#">Explain</a> | <a href="#">Compare</a> |

[Return](#)

## 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.

[View Recommendations](#) [Implement All](#)

| Select                           | SQL Text                                    | Parsing Schema | SQL ID                        | Statistics | SQL Profile | Index     | Restructure SQL | Miscellaneous | Error | Date     |
|----------------------------------|---|----------------|-------------------------------|------------|-------------|-----------|-----------------|---------------|-------|----------|
| <input checked="" type="radio"/> | select /*+ USE_NL(s c) FULL(s) FULL(c) A... | AST            | <a href="#">948kryk9vu0bm</a> |            | (97.4%) ✓   | (83.6%) ✓ |                 |               |       | 11/15/09 |
| <input type="radio"/>            | select /*+ USE_NL(s c) FULL(s) FULL(c) A... | AST            | <a href="#">65p7urqzb1h3j</a> |            | (96.6%) ✓   |           |                 |               |       | 11/15/09 |
| <input type="radio"/>            | SELECT NULL AS table_cat, t.owner...        | SYSMAN         | <a href="#">0prhvny3f97z</a>  |            | (81.9%) ✓   |           |                 |               |       | 11/15/09 |
| <input type="radio"/>            | select /*+ USE_NL(s c) FULL(s) FULL(c) A... | SYS            | <a href="#">by9m5m597zh19</a> |            |             |           |                 | ✓             |       | 11/15/09 |

Legend ✓ Recommended ✓ Implemented

# SQL Profile

- Contains corrections for poor optimizer estimates discovered during Automatic SQL Tuning.
- No change to application code. Transparent to user.
- Specific to an individual SQL statement
- Does not freeze the execution plan

# SQL Profile

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

```
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;
```

# Example Hints Used in SQL Profiles (1)

- Setting optimizer mode  
`ALL_ROWS`
- Disabling hints embedded in SQL  
`IGNORE_OPTIM_EMBEDDED_HINTS`
- Setting optimizer\_features\_enable  
`OPTIMIZER_FEATURES_ENABLE(default)`

## Example Hints Used in SQL Profiles (2)

- Adjusting the number of rows returned from a table  

```
OPT_ESTIMATE(@"SEL$AD385D36", TABLE, "T"@"SEL$5",
SCALE_ROWS=32.94409938)
```
- Adjusting the number of rows returned from an index scan  

```
OPT_ESTIMATE(@"SEL$AD385D36", INDEX_SCAN, "B"@"SEL$5",
FND_ID_FLEX_STRUCTURES_U2, SCALE_ROWS=32.9440
9938)
```
- Adjusting the number of rows returned from a join  

```
OPT_ESTIMATE(@"SEL$A05B8819", JOIN, ("INV"@"SEL$6",
"VNDR"@"SEL$6", "VNDR_SC"@"SEL$6"), SCALE_ROWS=1
5773.78428)
```

## Example Hints Used in SQL Profiles (3)

- 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)
```

# Proactive SQL Tuning in 10G

- Simulate automatic SQL Tuning using DBMS\_SQLTUNE package

- Custom package that uses DBMS\_SQLTUNE  

```

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 Subprograms

- SQL Tuning Advisor Subprograms

  - CREATE\_TUNING\_TASK

  - EXECUTE\_TUNING\_TASK

  - REPORT\_TUNING\_TASK

- SQL Profile Subprograms

  - ACCEPT\_SQL\_PROFILE

  - ALTER\_SQL\_PROFILE

  - DROP\_SQL\_PROFILE

- SQL Tuning Set Subprograms

  - CREATE\_SQLSET

  - LOAD\_SQLSET

# Privilege

- Use SQL Tuning Advisor – system privilege  
**ADVISOR**
- Create, drop and modify an SQL Profile – system privileges  
**CREATE ANY SQL PROFILE**  
**DROP ANY SQL PROFILE**  
**ALTER ANY SQL PROFILE**
- Manage SQL Tuning Sets  
**ADMINISTER SQL TUNING SET**  
or **ADMINISTER ANY SQL TUNING SET**

# DBMS\_SQLTUNE – Sample Code

**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;**

1/7/2008

## **DBMS\_SQLTUNE – Sample Code**

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

  

```
SET LONG 1000  
SET LONGCHUNKSIZE 1000  
SET LINESIZE 100  
SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK(  
  'my_sql_tuning_task')  
FROM DUAL;
```

## **DBMS\_SQLTUNE – Sample Code**

**BEGIN**

```
DBMS_SQLTUNE.ACCEPT_SQL_PROFILE(  
  task_name => 'my_sql_tuning_task' );
```

**END;**

**/**

# SQL Tuning Set

- DBMS\_SQLTUNE provides more flexibility compared to OEM when tuning multiple SQL statements.
- To tune multiple SQL first capture them in a SQL tuning set

# SQL Tuning Set Input Sources

| SELECT_CURSOR_CACHE | SELECT_WORKLOAD_REPOSITORY | SELECT_SQLSET     |
|---------------------|----------------------------|-------------------|
| basic_filter        | begin_snap                 | sqlset_name       |
| object_filter       | end_snap                   | basic_filter      |
| ranking_measure1    | basic_filter               | object_filter     |
| ranking_measure2    | object_filter              | ranking_measure1  |
| ranking_measure3    | ranking_measure1           | ranking_measure2  |
| result_percentage   | ranking_measure2           | ranking_measure3  |
| result_limit        | ranking_measure3           | result_percentage |
| attribute_list      | result_percentage          | result_limit      |
|                     | result_limit               | attribute_list    |
|                     | attribute_list             | plan_filter       |
|                     |                            | sqlset_owner      |

# Basic Filter

- **desc sqlset\_row**

sql\_id

Sql\_text

parsing\_schema\_name

module

action

elapsed\_time

cpu\_time

buffer\_gets

disk\_reads

direct\_writes

rows\_processed

fetches

executions

optimizer\_cost

# Create SQL Tuning Set

```
BEGIN
```

```
  dbms_sqltune.create_sqlset (  
    sqlset_name =>'PEAK_LOAD',  
    description =>'peak work load',  
    sqlset_owner=>'FEN508');
```

```
END;
```

```
/
```

# Load SQL Tuning Set

```
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;
```

# Load SQL Tuning Set

```
DBMS_SQLTUNE.LOAD_SQLSET(  
  sqlset_name    =>'PEAK_LOAD',  
  populate_cursor=>cur,  
  sqlset_owner   =>'FEN508'  
);  
  
  CLOSE cur;  
END;  
/
```

# CREATE\_TUNING\_TASK

## -- Sql tuning set format

```
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');
```

# DBMS\_SQLTUNE

- 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);
```

Substitutes literals with bind variable

## DBMS\_SQLTUNE

- Transport SQL Tuning Set

CREATE\_STGTAB\_SQLSET

PACK\_STGTAB\_SQLSET

Move staging table from source to destination

UNPACK\_STGTAB\_SQLSET

# Transport SQL Tuning Set (1)

```
-- In source database
BEGIN
DBMS_SQLTUNE.CREATE_STGTAB_SQLSET (
table_name      =>'STS_TAB',
schema_name     =>'FEN508',
tablespace_name =>'USERS');

DBMS_SQLTUNE.PACK_STGTAB_SQLSET (
sqlset_name     =>'PEAK_LOAD',
sqlset_owner    =>'FEN508',
staging_table_name =>'STS_TAB',
staging_schema_owner =>'FEN508');
END;
```

## Transport SQL Tuning Set (2)

```
-- Unpack in the destination database
BEGIN
DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET (
sqlset_name      =>'PEAK_LOAD',
sqlset_owner     =>'FEN508',
replace         =>TRUE,
staging_table_name =>'STS_TAB',
staging_schema_owner =>'FEN508'
);
END;
/
```

# DBMS\_SQLTUNE

- SQL Profile Category
  - Default category is DEFAULT
  - ACCEPT\_SQL\_PROFILE or ALTER\_SQL\_PROFILE can change the category of an SQL profile.
  - The active category is set through INIT parameter SQLTUNE\_CATEGORY  
alter session set  
SQLTUNE\_CATEGORY=TEST

# DBMS\_SQLTUNE

- 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);

# DBMS\_SQLTUNE

- 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');
```

## Other uses of DBMS\_SQLTUNE

- Repository of SQL and explain plans
- Upgrade performance testing

# DBMS\_SQLTUNE Documentation

- “PL/SQL Packages and Types Reference” has a complete description of subprograms.
- “Performance Tuning Guide” has many examples.

You can find my paper at DBA Knowledge  
web site:

<http://dbaknow.com/>

## Contact Us

Let us help you stay on the road to success

DBA Knowledge, Inc.  
385 Inverness Parkway,  
Suite 190  
Englewood, CO 80112

720-475-8600  
info@dbaknow.com  
www.dbaknow.com

