MANAGING COST-BASED OPTIMIZER STATISTICS FOR PEOPLESOFT

David Kurtz

psadmin.conf 2021

/*Go-Faster*/ Consultancy

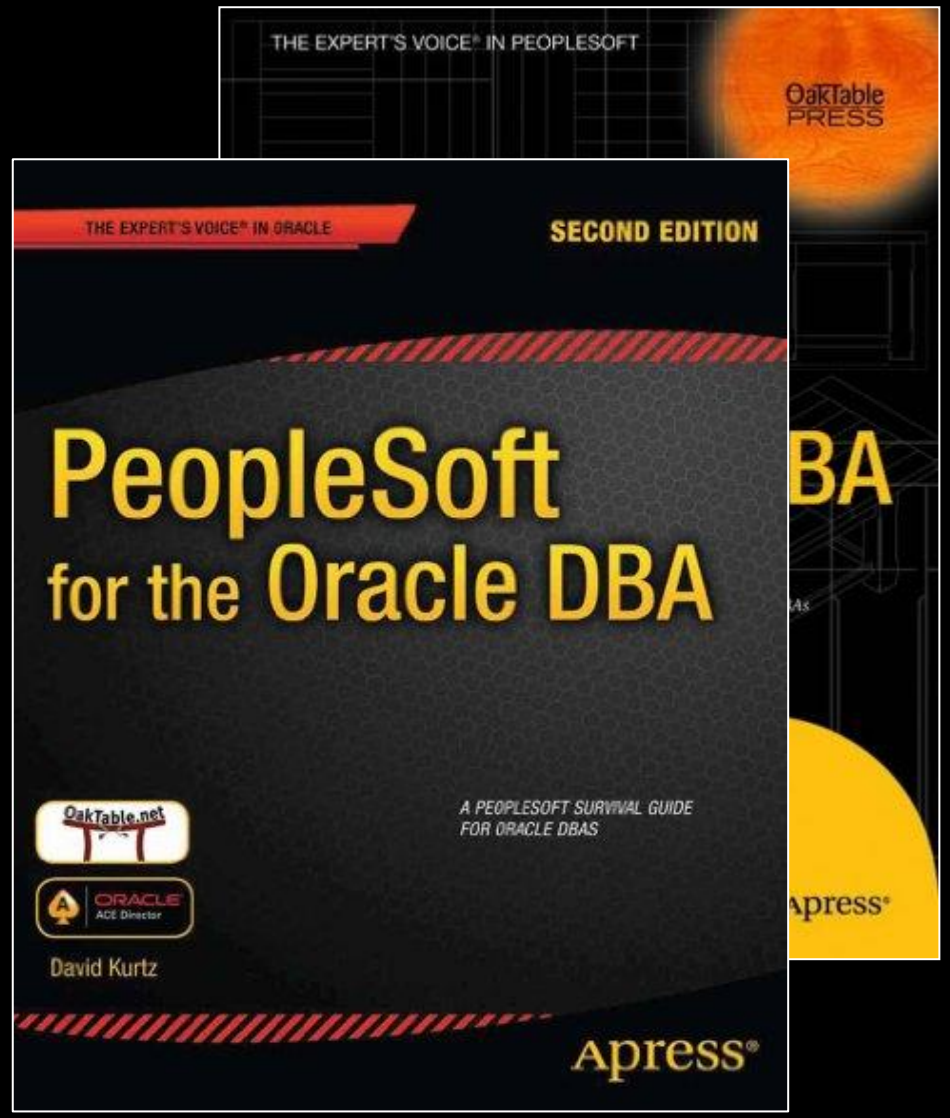
WHO AM I

Go-Faster Consultancy

- Performance tuning
  - Oracle RDBMS
  - PeopleSoft ERP
- www.go-faster.co.uk
- blog.go-faster.co.uk
- blog.psftdba.com

Oak Table

ORACLE ACE Director

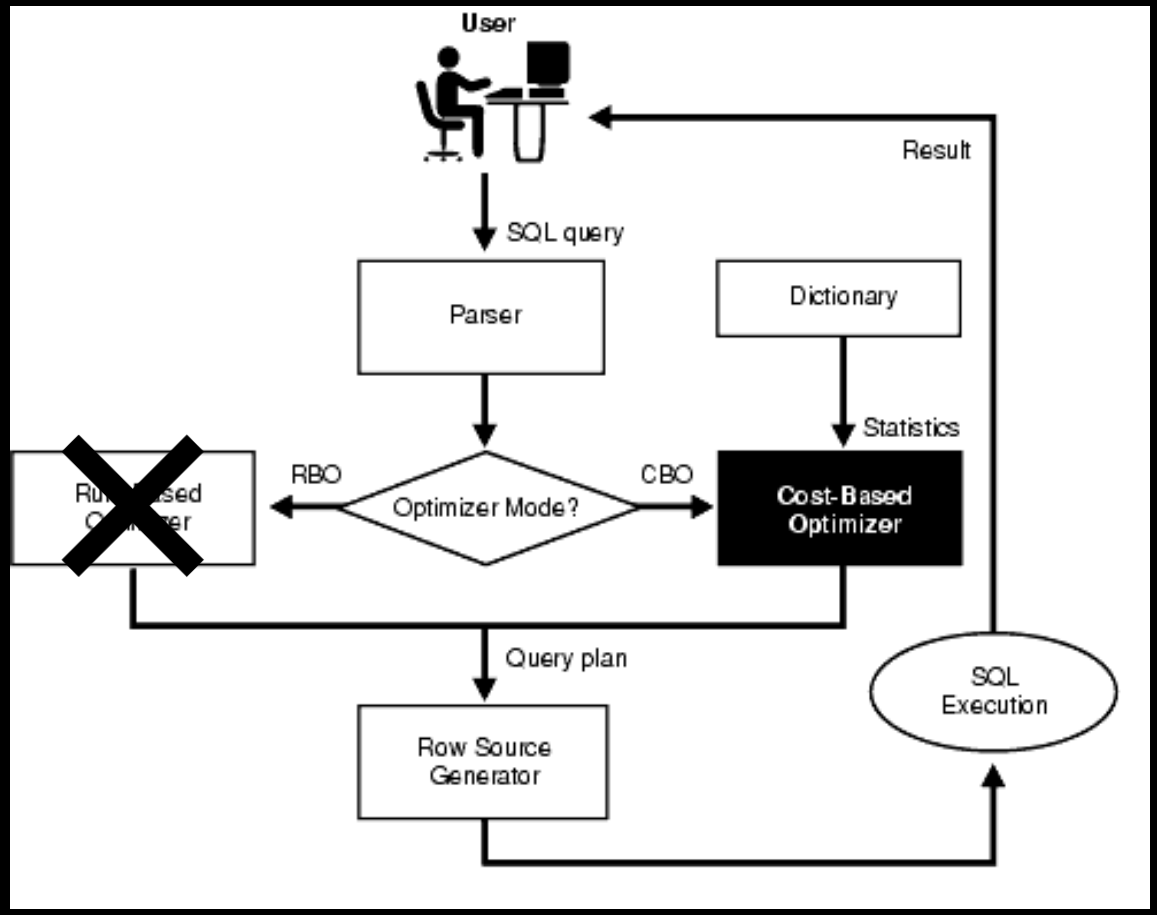


AGENDA

- How the Oracle Optimizer works
  - What is Cost
  - What are statistics
- Collecting Stats
  - Maintenance Window
  - At runtime
  - My recommendations
- Management Packages
  - GFCPSSTATS11
  - Oracle's PSCBO_STATS
- Not Statistics (but related)
  - Adaptive Optimization
  - SQL Plan Directives
  - Parameters

SQL PROCESSING

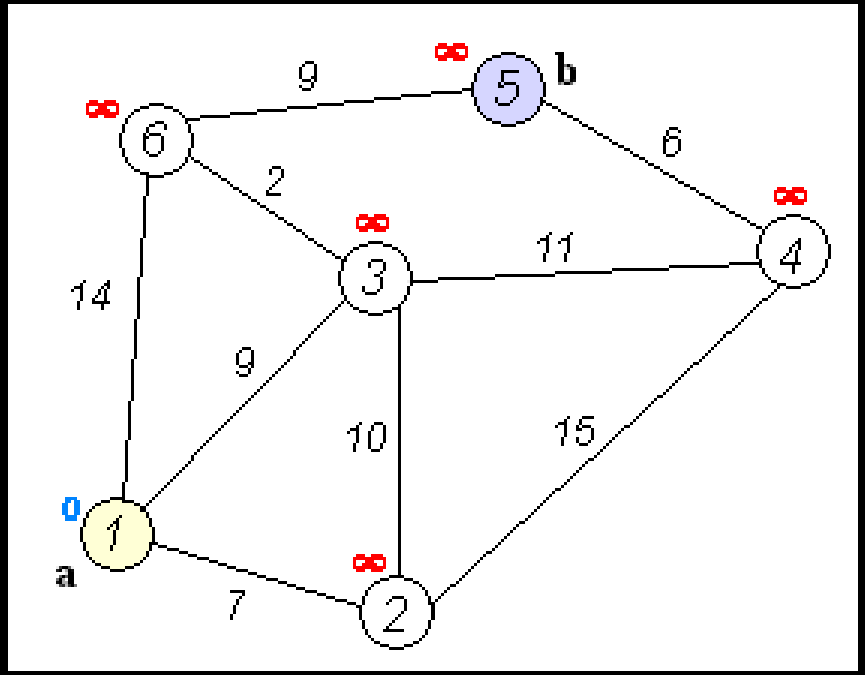
- The Parser checks both syntax and semantic analysis.
- The Optimizer uses costing methods to determine the most efficient way of producing the result of the query.
- The Row Source Generator receives the optimal plan from the optimizer and outputs the execution plan for the SQL statement.
- The SQL Execution Engine operates on the execution plan associated with a SQL statement and then produces the results of the query.
  - (Diagram from 9i Tuning Guide – things are more complex today)



COST-BASED OPTIMISATION

Car GPS

- Calculate route A → B
- It calculates all possible routes
- Chooses the least expensive
- Each arc in the route has a cost
- More expensive routes are pruned out as early as possible
- (Dijkstra's Algorithm – diagram from Wikipedia)



WHAT DO WE MEAN BY ‘COST’?

Car GPS

- Distance
- Time
- Blended measure, also including
  - Fuel Cost
  - Tolls
- Some route planners also factor in time based congestion.

SQL Optimizer

- Time
  - Physical I/O
  - CPU Cost
  - Built in heuristics
- Very roughly, optimizer cost is an estimate of execution time, where the unit of time is the duration of a single block read

Statistics

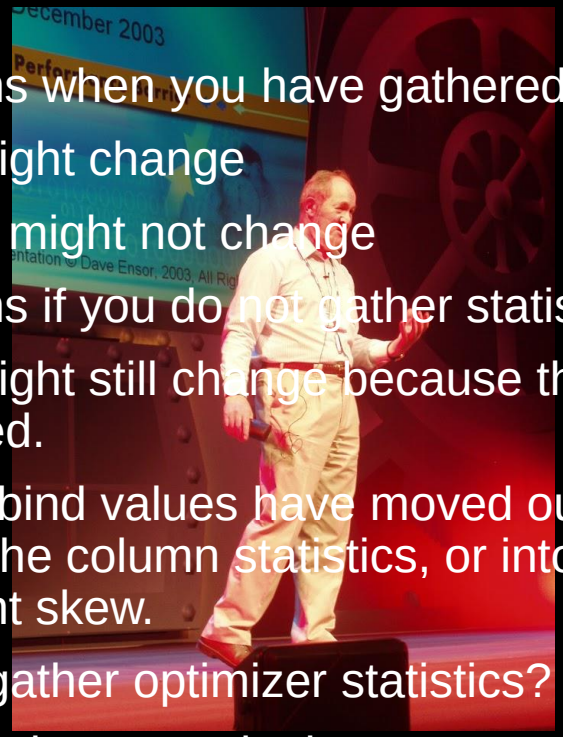
- Number of rows in a table
- Number of blocks in a table
- Range of data values on a column (min/max)
- Distribution of data values on a column
- Number of rows in an index
- Number of distinct keys
- How many table blocks will we visit if we scan through an index (clustering factor).

ENSOR PARADOX

Dave Ensor

"the grand old man of performance optimization methodology in the Oracle world" – Jonathan Lewis

- **"The only time that it is safe to gather statistics is when to do so would make no difference."**
- What happens when you have gathered statistics?
  - The plan might change
  - Or the plan might not change
- What happens if you do not gather statistics?
  - The plan might still change because the data has changed.
    - Literal or bind values have moved outside the range of the column statistics, or into a region of different skew.
- Why do we gather optimizer statistics?
  - So that the plans remain the same!



COST-BASED OPTIMIZER STATISTICS

Oracle Supplied PL/SQL Packaged Procedure *DBMS_STATS* that calculates and manages statistics.

- Calculate Statistics on
  - Object
  - Objects in schema
  - Whole database in schema
  - Eg. *DBMS_STATS.GATHER_TABLE_STATS()*
- Delete Statistics
- Lock Statistics
- Export/Import Statistics

MAINTAINING OPTIMIZER STATISTICS

Up to **Oracle 9*i*** it was common to write custom scripts to refresh all statistics

- Could consume a lot of resource and take a long time.
- Inevitably computed statistics on too many tables too often.

Oracle 10*g*/11*g*

- Maintenance Window
  - 10pm-6am weekdays, all day at weekends
- Stale (by default, if 10% change)
  - From 11g can be set per table
- Statistics Locking
  - Tables omitted from schema-wide gather processes if statistics locked
  - use FORCE option to override lock

Oracle 19c Exadata

- Real-Time Statistics
- High-Frequency Automatic Statistics Collection

WHAT STATISTICS TO COLLECT

DBMS_STATS parameters

- *METHOD_OPT* – Histograms
- *DEGREE* – parallelism
- *GRANULARITY* – partitions

Only via table statistics preferences

- *INCREMENTAL* - partitioning
- *STALE_PERCENT* - Stale Threshold

DO NOT SPECIFY

- *SAMPLE_SIZE*
  - always use default of *DBMS_STATS.AUTO_SAMPLE_SIZE* to invoke new hash-based number of distinct values (NDV) algorithm and top-n histograms
- *CASCADE*
  - Controls whether index statistics collected at same time as table stats
  - true by default

SHOULD WE COLLECT HISTOGRAMS BY DEFAULT?

Oracle Default:

- histograms created automatically where skew and usage
  - set_global_prefs
    - *method_opt => 'for all columns size auto'*
  - set_table_prefs
    - *method_opt => 'for all columns size auto, for columns XXX size 1'*

http://coug.us/wp/wp-content/uploads/2019/03/MColgan_Best_Practices_for_Manging_statistics.pdf

Jonathan Lewis:

- Do not gather histograms
  - Unless Frequency or Top-Frequency
  - Get rid of all "size repeat" in 12c+
- Change default
  - set_global_prefs
    - *method_opt => 'for all columns size 1'*
  - set_table_prefs
    - *method_opt => 'for all columns size 1, for columns XXX size N'*
  - Write scripts for "unusual" histograms

https://cdn.ymaws.com/ukoug.org/resource/resmgr/scotland_19/jonathan_lewis.pdf

COLLECTING STATISTICS AT RUNTIME

Temporary working storage tables in all batch processing but especially Application Engine

- Truncated,
- repopulated,
- and then queried.

Explicitly collect stats during batch processes

- In AE with *%UPDATESTATS*
  - Command in DDL Models 4 & 5
  - In some delivered programs
  - You may need to add more
- Some Cobol programs collect statistics
  - From 8.55 also uses DDL Model

General Principle:

Tell the optimizer the truth

When you work with any table:

- Populate the table
- Collect Statistics on the table*
- Use the table

DELIVERED DDLMODEL

Model 4

- %UpdateStats([table],LOW);

```
DBMS_STATS.GATHER_TABLE_STATS
(ownname=> [DBNAME], tabname=>[TBNAME]
,estimate_percent=>1
,method_opt=> 'FOR ALL INDEXED COLUMNS SIZE 1'
,cascade=>TRUE);
```

Model 5

- %UpdateStats([table],HIGH);

```
DBMS_STATS.GATHER_TABLE_STATS
(ownname=> [DBNAME], tabname=>[TBNAME]
,estimate_percent=> dbms_stats.auto_sample_size
,method_opt=> 'FOR ALL INDEXED COLUMNS SIZE 1'
,cascade=>TRUE);
```

Problems

- LOW uses 1% sample size
  - this may be greater than *AUTO_SAMPLE_SIZE*
  - It disables the new hash-based number of distinct values algorithm introduced in 11g
    - so still uses a sort, that can have a higher overhead
    - Prevents use of Top-N and Hybrid histograms from Oracle 12c
- *FOR ALL INDEXED COLUMNS SIZE 1*
  - Only updates columns statistics on index columns.
  - Non-indexed columns are not updated and will thus be out of date.
  - Removes all histograms from a table, irrespective of the statistics preferences

RECOMMENDED SIMPLE DDLMODEL

Model 4 & 5

- *%UpdateStats([table],LOW|HIGH);*
- Same for both

```
DBMS_STATS.GATHER_TABLE_STATS
(ownname=> [DBNAME]
,tabname=> [TBNAME]
,force  => TRUE
);
```

Recommendations

- No parameters are specified
- Defaults are always used
  - As amended by table statistics preferences
  - *AUTO_SAMPLE_SIZE*
  - *CASCADE=TRUE*
  - *METHOD_OPT=GATHER AUTO*
- Specify *FORCE=>TRUE* to override locked statistics

WHAT HAPPENS IF YOU DON'T COLLECT STATISTICS AT RUNTIME?

Yesterday

- Ran a batch program
- It truncated and populated a table used for temporary working storage
- So the statistics on the table are now stale

Today

- Ran the same batch program again
- It truncated the working storage table
  - This does not remove the statistics
- It repopulated the working storage table
  - New data, new process instance and old statistics

Last Night

- Statistics collected on during maintenance window
- On tables with unlocked statistics
  - without statistics or
  - with stale statistics
- Statistics updated on working storage table in batch process

Conclusion:

- The only time to collect statistics on working storage tables is in the process that populates them.
- It is too late to collect statistics on temporary working storage tables during the maintenance window, or even with Real-Time statistics

WHAT IS THE PROBLEM WITH STALE STATISTICS ON TEMPORARY TABLES?

For most temporary working storage tables

- PROCESS_INSTANCE is the first column in the unique key.
- All the rows will have the same PROCESS_INSTANCE
- Most processes will select rows from table with a criterion on PROCESS_INSTANCE
  - let's assume there are no other predicates

```
SELECT * FROM <table>
WHERE process_instance = %PROCESS_INSTANCE
```

- Every column has a min/max value

Statistics on the temporary table

- Min/max values on PROCESS_INSTANCE are yesterday's process
- Therefore current values of PROCESS_INSTANCE in the table > column maximum value in the statistics
- The optimizer will
  - calculate that no rows exist for today's process instance.
  - and assume that it will get a single row.
  - It will probably use an index
- In fact
  - All the rows will be returned.
  - It might have been better to full scan this table

OPTIMIZER DYNAMIC SAMPLING

Designed to augment rather than replace optimizer statistics

- Oracle Optimizer Blog: Dynamic sampling and its impact on the Optimizer
- Basic statistics collected during SQL compilation
  - Default level 2: if no statistics exist
  - Level 4: complex predicates, multiple predicates on same table

My Opinion

- Very effective for PeopleSoft
- Generally recommend level 4
  - However, this can cause parse problem in nVision with very large trees and ‘use literal values’ performance option.
    - Trigger to set it back to 2 (s**ee Github set_prcs_sess_parm_trg.sql & set_prcs_sess_parm.sql**)
- You may find some cases where explicit object optimizer statistics are still needed

RECOMMENDATION FOR STATISTICS ON TABLES USED FOR TEMPORARY WORKING STORAGE

- **Lock Statistics**
  - to exclude them from maintenance windows statistics job
    - Temporary Table records
    - Normal Table records used for working storage (%AET, %TAO)
- **Delete Statistics**
  - either the statistics are collected during the batch processes
  - Or they are never collected and we can rely on dynamic statistics

Change DDL Model

- Use FORCE=>TRUE option in DBMS_STATS to override lock
- Consider using
  - GFCPSSTATS11 package

Never

- Call DBMS_STATS directly in Application Engine because it implies a commit
  - use %UPDATESTATS instead.
  - The commit is why %UPDATESTATS is suppressed inside a DO WHILE loop

PREFERENCE_OVERRIDES_PARAMETER

From Oracle 12.2

- New statistics preference PREFERENCE_OVERRIDES_PARAMETER
  - If set then DBMS_STATS parameters ignored
- Recommendation
  - Set it to override any scripts/DDL model that set inappropriate parameters
    - Even if you don't have this problem, set this preference to prevent it occurring.
  - Any settings you want to keep (mostly METHOD_OPT) should be converted to table preferences.
- See
  - https://blog.go-faster.co.uk/2020/11/oracle-122-new-statistic-preference.html
  - Marian Colgan: https://sqlmaria.com/2017/07/11/overriding-dbms_stats-parameter-settings/
- But what happens if Application Designer rebuilds a table?

STATISTICS HISTORY

By Default

- Every time statistics are collected, old statistics are copied to statistics history tables (*WRI$_OPTSTAT_%*)
- History is retained for 31 days

Problem

- Statistics on are collected some tables can be collected very frequently.
  - In some cases, thousands of times per process
  - *DBMS_STATS.GATHER_TABLE_STATS* is also responsible for purge.
    - I have seen it blocked by locks from other instances of itself.

Recommendation

- Disable Statistics History
- Set history retention to 0 days

```
exec dbms_stats.alter_stats_history_retention(0);
exec dbms_stats.purge_stats(SYSDATE);
```

- Reorganise statistics history tables to relinquish space and reset high water mark
  - John Hallas' Blog: Oracle DBA – A lifelong learning experience: Purging statistics from the SYSAUX tablespace

Drawback

- Statistics History Retention is a global setting
  - Can’t set it per table
- Lose the ability to determine table growth rates from the statistics history

‘THE EVIL THAT MEN DO LIVES AFTER THEM; THE GOOD IS OFT INTERRED WITH THEIR BONES.’

- Huge numbers of SQL Statements processed.
- The optimizer does a perfectly good job with most of them
- Generally, performance problems occur when the optimizer makes poor choices on a few statements.

The optimizer has to be generic.

- It has to work with every application that has ever and will ever be written
- It is principally designed to work with applications that are written the way Oracle recommend.

However, PeopleSoft is not such an application.

- Not least because of its platform agnostic design.
- Including but no limited to:
  - Special Values
    - 0 and Space for NULL
  - Nearly all columns are Not Null
  - Unique Indexes instead of Primary Keys
  - Nullable columns in Unique Keys

HINTS

Hints are Directives to the Optimizer

- They mostly force certain operations
  - Eg. *FULL, INDEX, USE_NL*
- They can prevent certain transformations
  - Eg. *NO_EXPAND, NO_UNNEST*
- The can change optimizer environmental parameters for a statement
  - Eg. *OPT_PARAM('parallel_degree_policy','AUTO')*

Hint Paradox

- Why do we use hints?
  - To Improve Performance
- What happens to cost when we add a hint?
  - It usually increases*
- So the cost goes up but the runtime goes down!
- What does that say about optimizer cost?
  - There are problems with the cost model where the estimates are not correct, and it this is often at the root of performance issues.

HINT RECOMMENDATIONS

In those relatively few cases you need to intervene.

- Look for where the estimate cost diverges from the actual cost
- Make sure statistics are up to date
- Consider providing/removing histograms
- Consider indexes
- It may be necessary to use a hint to force or prevent optimizer behaviour.
- Hints are directives. That can be applied via
  - SQL Baselines (match SQL_ID)
  - SQL Patches (match SQL_ID)
  - SQL Profiles (match FMS, but requires tuning pack)
    - force matching doesn't work well with a mixture of bind variables and literals
  - Directly in the code

Personal View

- I try to avoid hinting code.
  - Sometimes add hints to Application Engine steps with dynamic code
    - Sometimes, hints also have to be generated dynamically in dynamically generated SQL.
  - Note %SQLHint AE Macro
- I will not change COBOL source code
  - Use SQL profiles on dynamic SQL (they never contain binds).
    - Dynamically generated sets of SQL profiles
  - Can add hints into stored statements, but can also use SQL profiles or SQL baselines

HINTING APPLICATION ENGINE CODE

Application Engine Code

```
%InsertSelect(DMK,JOB J)
FROM PS_JOB J
…
```

Resolved SQL

```
INSERT INTO PS_DMK
(EMPLID, EMPL_RCD, EFFDT, EFFSEQ,
SETID_DEPT, DEPTID)
SELECT J.EMPLID, J.EMPL_RCD, J.EFFDT,
J.EFFSEQ, J.SETID_DEPT, J.DEPTID
FROM PS_JOB J
…
```

HINTING APPLICATION ENGINE CODE

Application Engine Code

```
%InsertSelect(DMK,JOB J, EMPLID=
/*+LEADING(J)*/ J.EMPLID)
FROM PS_JOB J
…
```

Resolved SQL

```
INSERT INTO PS_DMK
(EMPLID, EMPL_RCD, EFFDT, EFFSEQ,
SETID_DEPT, DEPTID)
SELECT /*+LEADING(J)*/ J.EMPLID,
J.EMPL_RCD, J.EFFDT, J.EFFSEQ,
J.SETID_DEPT, J.DEPTID
FROM PS_JOB J
…
```

HINTING APPLICATION ENGINE CODE %SQLHINT()

Application Engine Code

```
%InsertSelect(DISTINCT DMK,JOB J,
EMPLID= /*+LEADING(J)*/ J.EMPLID)
FROM PS_JOB J
…
```

Resolved SQL

```
INSERT INTO PS_DMK
(EMPLID, EMPL_RCD, EFFDT, EFFSEQ,
SETID_DEPT, DEPTID)
SELECT DISTINCT /*+LEADING(J)*/
J.EMPLID, J.EMPL_RCD, J.EFFDT,
J.EFFSEQ, J.SETID_DEPT, J.DEPTID
FROM PS_JOB J
…
```

The hint is now in the wrong place, so it is only a comment!

HINTING APPLICATION ENGINE CODE

Application Engine Code

```
%SqlHint(SELECT, 1,
         '/*+LEADING(J)*/', ORACLE)
%InsertSelect(DISTINCT DMK,JOB J)
FROM PS_JOB J
…
```

Resolved SQL

```
INSERT INTO PS_DMK
(EMPLID, EMPL_RCD, EFFDT, EFFSEQ,
SETID_DEPT, DEPTID)
SELECT /*+LEADING(J)*/ DISTINCT
J.EMPLID, J.EMPL_RCD, J.EFFDT,
J.EFFSEQ, J.SETID_DEPT, J.DEPTID
FROM PS_JOB J
…
```

Now, the hint is now in the right place!

HINTING APPLICATION ENGINE CODE %SQLHINT()

Application Engine Code

```
%SqlHint(INSERT, 1, '/*+APPEND*/', ORACLE, ENABLE)
%SqlHint(INSERT, 1,
       '/*Developer Comment*/',ORACLE,DISABLE)
%SqlHint(SELECT, 1, '/*+LEADING(J)*/', ORACLE)
%SqlHint(SELECT, 2, '/*+UNNEST(J1)*/', ORACLE)
%SqlHint(SELECT, 3, '/*+UNNEST(J2)*/', ORACLE)
%InsertSelect(DISTINCT, DMK,JOB J)
  FROM PS_JOB J
 WHERE %Sql(DMK_CURJOB,JOB,J,J1,J2)
…
```

Resolved SQL

```
INSERT /*+APPEND*/ INTO PS_DMK
(EMPLID, EMPL_RCD, EFFDT, EFFSEQ , SETID_DEPT, DEPTID)
 SELECT /*+LEADING(J)*/ DISTINCT J.EMPLID, J.EMPL_RCD,
J.EFFDT, J.EFFSEQ, J.SETID_DEPT, J.DEPTID
  FROM PS_JOB J
 WHERE J.EFFDT = (
 SELECT /*+UNNEST(J1)*/ MAX(J1.EFFDT)
  FROM PS_JOB J1
 WHERE J1.EMPLID = J.EMPLID
   AND J1.EMPL_RCD = J.EMPL_RCD
   AND J1.EFFDT <= TO_DATE(TO_CHAR(SYSDATE,'YYYY-MM-
DD'),'YYYY-MM-DD'))
   AND J.EFFSEQ = (
 SELECT /*+UNNEST(J2)*/ MAX(J2.EFFSEQ)
  FROM PS_JOB J2
 WHERE J2.EMPLID = J.EMPLID
   AND J2.EMPL_RCD = J.EMPL_RCD
   AND J2.EFFDT = J.EFFDT)…
```

SOMETIMES, WHEN COLLECTING STATISTICS YOU NEED MORE CONTROL

Example: Time & Labor Calculation (TL_TIMEADMIN)

- Looped processing of batches of rules
- Statistics on a set of 20+ working storage tables collected for every loop
- Overhead of statistics collect can become dominant performance issue
- But dynamic statistics (optimizer dynamic sampling) not enough on some tables - need explicit statistics on some tables
- Don’t want to change delivered code.

Change DDL Model

- Call custom PL/SQL Package from DDL Model
- Call DBMS_STATS from custom PL/SQL package
- Logic in package to determine whether to collect statistics

It is time to talk about

- *GFCPSSTATS11*
- *PSCBO_STATS*

GFCPSSTATS11

Written by Me

- PL/SQL Package
  - Superseded wrapper script designed for 10g
  - Rewritten and simplified for Oracle >=11g
- Metadata table (like a PeopleTools table)
  - Specifies statistics preferences
  - Also controls per PeopleTools record whether it
    - collect statistics
    - suppress statistics collection
    - delete statistics
    - or only collects statistics if stale
- Triggers to apply preferences
  - DDL Trigger on table creation
  - DML Trigger on metadata change
- Called from DDL Models 4 & 5
- Now also controls locking and deleting statistics on tables

Advantages

- Leave Oracle maintenance window in place
- As far as is possible, manage statistics on PeopleSoft database in the same way as any other Oracle database in your estate.
- Control statistics collection behaviour declaratively with table statistics preferences
- Work with the database rather than fight against it.

Download

- Documentation: http://www.go-faster.co.uk/docs.htm#Managing.Statistics.11g
  From Github: https://github.com/davidkurtz/gfcpsstats

PSCBO_STATS

Written by Oracle

- Carlos Sierra, Oracle “Center of Excellence”
- PL/SQL Package
  - *Based on script for Siebel*
  - Dates back to Oracle 10
  - Enhanced for Oracle >=11g
- Control tables to specify statistics behaviour
- Delivered with PT8.57(?)
  - Documented in PeopleBooks
  - Download from Oracle Support Note ***1322888.1***

Concept broadly similar to *GFCPSSTATS11*

- Both were developed completely independently
- To be fair, some of my original criticisms have been dealt with in the V2 release for Oracle 11g

Disadvantages

- Support *‘Note: Oracle does not certify or test the pscbo_stats utility for support purposes. Its use is considered to be a customization. If there is a problem or question with pscbo_stats, the Global Support Center will not be handling service requests regarding its use.’*
- No support for
  - Table Statistics Preferences
  - Extended Statistics
- Must always use package rather than *DBMS_STATS* directly.
  - Replaces Default Oracle Maintenance Windows Stats Job

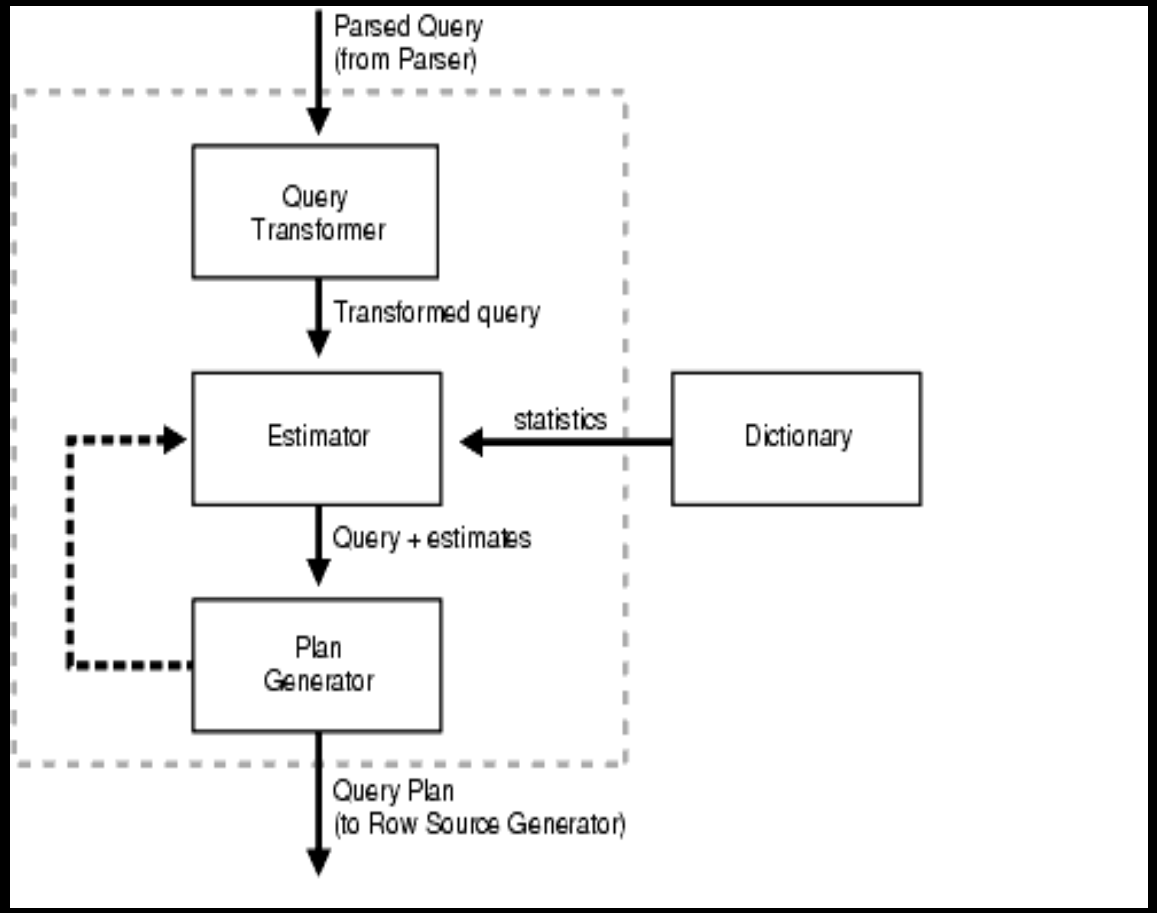
NOT ABOUT STATISTICS, BUT OPTIMIZER RELATED

ADAPTIVE OPTIMIZATION IN 12.1

Originally known as ‘Cardinality Feedback’

- Additional statistics gathered
- Used to change decisions made by optimizer

- In a parse intensive system (such as PeopleSoft) this can become a significant overhead.



OPTIMIZER ADAPTIVE FEATURES SPLIT IN 12.2 (OR 12.1 WITH PATCHES)

OPTIMIZER_ADAPTIVE_PLANS=TRUE

- Very Beneficial
  - Nested loop/Hash join selection
- Less Important in PeopleSoft
  - Adaptive Parallel distribution
  - Star Transformation Bitmap Pruning

(see PeopleSoft and Adaptive Query Optimization in Oracle 12c)

OPTIMIZER_ADAPTIVE_STATISTICS=FALSE

- SQL Plan Directives
- Statistic Feedback for Joins
- Performance Feedback
- Adaptive Sampling for Parallel Execution
- Enabling this causes performance problems at compile time
  - Enabled by default in 12.1 when Adaptive Features enabled
  - Disabled by default in 12.2

(see Oracle Optimizer Blog: Optimizer Adaptive Features in Oracle Database 12c Release 2)

ADAPTIVE OPTIMIZATION RECOMMENDATIONS

From 12.2 (or 12.1 with patches)

- Use the Defaults
  - *OPTIMIZER_ADAPTIVE_STATISTICS=FALSE*
  - *OPTIMIZER_ADAPTIVE_PLANS=TRUE*

  - Set *OPTIMIZER_DYNAMIC_SAMPLING=4*
    - Or leave default if parse problems in nVision

SQL PLAN DIRECTIVES

SQL Plan Directives

- Still generated but not used if Optimizer Adaptive Statistics disabled
- Matched on SQL_ID, so not effective with
  - dynamically generated literal values
  - Other dynamically generated code
- Will continue to build up over time
- We experienced a progressive increase in parse CPU in 12.1.0.2.
  - Resolved by dropping all directives, but couldn't reproduce on another environment!

Recommendation

- Disable directive generation by setting
  - ***_sql_plan_directive_mgmt_control=0***
  - (or ***_optimizer_dsdir_usage_control=0***)
- Drop Existing Plan Directives
  - Expect 3s / directive

```
set serveroutput on timi on
DECLARE
  l_count INTEGER := 0;
BEGIN
  FOR i IN(select * from dba_sql_plan_directives) LOOP
    dbms_output.put_line('Dropping SPD '||i.directive_id);
    dbms_spd.drop_sql_plan_directive(i.directive_id);
    l_count := l_count+1;
  END LOOP;
  dbms_output.put_line(l_count||' directives dropped');
END;
/
```

DATABASE INITIALISATION PARAMETERS

_unnest_subquery=FALSE

- Required at Install since Oracle 9i
- **It is critically important that this parameter is set**
  - But I still find PeopleSoft environments where this hasn't been set
- Prevents correlated sub-queries being unnested into in-line views.
  - The optimizer does this because it miscosts the cardinality of correlated columns.
  - If not set, especially severe performance effect in HR, but all products affected

_gby_hash_aggregation_enabled=TRUE

- Oracle have changed their advice.
  - Now recommend leaving the default
  - Previously Recommended FALSE since Oracle 9i
  - But only because some SQL statements do not have ORDER BY clause.
  - This setting prevents GROUP BY and DISTINCT being done with a HASH if cheaper and reverts to the original sort method.
  - When using a sort to perform a GROUP BY or DISTINCT on a non-partitioned object, the rows are usually returned in that order.
  - But a different result set order can occur if the plan changes even if this parameter is set!
  - No longer needed because modern PeopleSoft applications have been updated to user ORDER BY
- SQL92 standard says that query result order is arbitrary.
- **If you care about order, add an ORDER BY clause.**

19C NEW EXADATA FEATURES

Real-Time Statistics

- More accurate statistics before they can be recollected
- Augments normal statistics
  - Increase in volumes
  - Increase in range of column values
- Direct path insert from 12.1, DML from 19
- Based on table monitoring
  - Low impact
- Enabled by default up to 19.9, disabled from 19.10, set parameter
  - *OPTIMIZER_REAL_TIME_STATISTICS=TRUE*

High-Frequency Automatic Statistics Collection

- More accurate statistics by more frequent statistics collection.
- Continuous maintenance process
- Collects stats on stalest tables first
- Disabled by default, enable/configure via API
  - Job runs every 15 minutes, maximum runtime 60 minutes.

Initial testing suggests both features advantageous to PeopleSoft.

Separate presentation

- Oracle 19c: Real-Time and High-Frequency Statistics Collection
  - https://www.go-faster.co.uk/p/oracle-19c-real-time-and-high-frequency.html)

CONCLUSION

CONCLUSION

Do it the Oracle way, work with the database as far as is possible.

- **Collect optimizer statistics in the maintenance window**
- **Use GFCPSSTATS11 instead of PSCBO_STATS**
  - **To manage table statistic preferences**
  - **Finer grain control of when to collect statistics during processing**
  - **Data driven**
- **Disable statistics history retention**
  - **(unfortunate but necessary)**
- **Hint with SQL Baselines/Patches/Profiles where possible**
  - **Embed in code where you must, use %SQLHint macro in AE**

QUESTIONS