TECHNICAL NOTE

MANAGING OPTIMIZER STATISTICS FOR PEOPLESOFT ON ORACLE (11G OR HIGHER)

Prepared By David Kurtz, Go-Faster Consultancy
Technical Note
http://www.go-faster.co.uk/GFCPSSTATS11.manual.pdf
Monday 8 March 2021
(E-mail: david.kurtz@go-faster.co.uk, telephone +44-7771-760660)
File: gfcpsstats11.docx, 8 March 2021

Contents

Introduction

I have been considering how to collects optimizer statistics for a PeopleSoft system running on an Oracle 11g database. Despite 11g being several years old, most of my current customers are still using 10g, though some are looking at the upgrade to 11g. I believe a slightly different approach is required.

In 2009, I wrote a series of blog postings[1] on the subject of collecting statistics. However these were all based on Oracle 10g. I proposed a PL/SQL package[2] that would use meta-data in a database table to determine how to collect statistics on a table, or deliberate supress collection of statistics.

I also recommended that statistics on tables created for use as temporary records in Application Engine programs should have their statistics deleted and locked[3] to prevent system-wide jobs refreshing their statistics.

Update 4.2.2015: *Since I first wrote this document I have implemented this technique in PeopleSoft systems at two customer site, so I think we are now beyond the experimental stage. I would still welcome any feedback, and the opportunity to work with someone on a PeopleSoft system on Oracle 11g.*

New Features in 11g

Oracle considerably enhanced the delivered DBMS_STATS package in 11g. It became possible to specify default values for parameters in the *dbms_stats.gather_table_stats* program for for each table.

Partitioning is not used by default in PeopleSoft, but if you have introduced it then you will be glad that some of the problems with global statistics have been corrected in 11g[4]. There is a new feature in 11g called incremental statistics that produces better global statitics on partitioned tables by maintaining additional data called synopses.

---

ss[1] Controlling How %UpdateStats Collects Optimizer Statistics
http://blog.psftdba.com/2009/06/controlling-how-updatestats-collects.html

[2] http://www.go-faster.co.uk/scripts.htm#wrapper848meta.sql

[3] Statistics Management for PeopleSoft Temporary Records in Application Engine Programs
http://blog.psftdba.com/2009/04/statistics-management-for-peoplesoft.html.

[4] Gathering Aggregated Cost-Based Optimiser Statistics on Partitioned Objects in Oracle 11gR2 http://www.go-faster.co.uk/docs.htm#Partition.Statistics.11g

Collecting Statistics on Oracle 11g

The guiding principle in 11g, and one that is not specific to PeopleSoft, is that instead of calling *dbms_stats.gather_table_stats* with the desired parameters, we should set table preferences with the desired parameters and then just call *dbms_stats* without table specific parameters. We can then just leave the default database and schema-wide procedures get on with the job of collecting statistics.

Statistics on working storage tables cannot be maintained by scheduled jobs because the data in the table will have changed by the process that uses them. Statistics on such tables should be deleted and locked so they are omitted from schema and database-wide statistics maintenance processes. Thus, unless the process itself collects statistics, overriding the lock, there will not be any statistics on the table and Oracle will use Optimizer Dynamic Sampling. However, where the process does collect statistics, table preferences can still apply.

Collecting Statistics in PeopleSoft on Oracle 11g

When it comes to applying this principle in PeopleSoft, we have a few challenges to overcome, but nothing that is impossible.

When using PeopleSoft's Application Designer to migrate it is typical to rebuild a table in order to add or change a column. Any related objects will be lost. Application Designer will rebuild the indexes. However, auditing triggers will be lost. The same is true for table statistics preferences that are lost when a table is dropped. So we need to hold the preferences as meta-data in a table. Then we can apply the preferences to the tables as they are created using a DDL trigger.

PeopleSoft temporary records can correspond to many tables that are used as non-shared temporary working storage tables. The same table preferences will need to be applied to all the temporary table instances. So the meta-data needs to be held for each record.

The *%UpdateStats* macro is used to collect statistics during Application Engine programs. This uses the DDL model to call the *dbms_stats* PL/SQL procedure. The simplistic approach would be to change the delivered DDL model to remove the *sample_size* and *method_opt* parameters but to specify *force=>TRUE*. However, I have retained the functionality developed in the original *wrapper* package to optionally suppress collection of statistics, or only to collect statistics when they are stale.

In the case of COBOL programs, the %UpdateStats macro is put into stored statements. These can be changed to call the same PL/SQL package as the DDL models.

I generally recommend increasing the setting of the Oracle initialisation parameter OPTIMIZER_DYNAMIC_SAMPLING from the default of 2 to 4. However, I have experienced some problems with nVision where there can be hundreds of predicates. In that specific case I would revert to the default level of 2.

I have created a new packaged procedure GFCPSSTATS11, although it is based on the wrapper package that I produced for 10g.

Oracle's PSCBO_STATS package

I can't discuss collecting statistics for PeopleSoft without discussing Oracle's CBO_STATS package. Oracle published document 1322888.1 "*pscbo_stats - Improving Statistics in Oracle RDBMS for PeopleSoft Enterprise*". It takes a similar approach to the package I proposed in the second edition of PeopleSoft for the Oracle DBA. A PL/SQL package is used to collect statistics. A number of tables control whether and how statistics are collected on each record. The package is also intended to be used to collect schema-wide statistics.

The PSCBO_STATS package is a valiant attempt to solve a genuine problem, and it has continued to evolve since its initial release. However, I have a number of objections to it.

- It is fundamentally a 10g solution. It does not use 11g table preferences.
- It does use the Oracle automatic sample size in 11g if histograms are not to be collected. Otherwise, it defaults to the previous behaviour of either using 100% sample size for when called by %UpdateStats with the 'high' sample size, or a variety of fixed sampe sizes based on internal rules and the number of rows in the table.
    - In 11g, automatic sample size produces better values, and if necessary a specific sample size can be set with a table preference.
- The package contains a procedure that collects statistics on all objects in the schema that also refreshes statistics that are not stale but which have not been refreshed for a period of time determined by the size of the table.
    - In 11g, this can be handled by setting a table specific stale threshold. A low threshold might be appropriate for larger tables.
- When it collects histograms it always sets the maximum bucket size of 254. This may not always be desirable for height balanced histograms.
    - In 11g, the collection of histograms can be controlled via the METHOD_OPT table preference.
- There is no support for collecting aggregated or incremental statistics on partitioned objects.
    - In 11g, table preferences for granularity and incremental statistics can be set.

I think that the 11g table preferences offer better and finer control over the collection of statistics.

APPENDIX

Implementation

All of the scripts are available on Github at https://github.com/davidkurtz/gfcpsstats

1. Create metadata table PS_GFC_STATS_OVRD, and index, using script gfcpsstats11_metadata.sql.
    a. You may also choose to create a record in PeopleSoft Application Designer to correspond to this table.
2. Create PL/SQL packaged procedure SYSADM.PS_GFC_STATS_OVRD using gfcpsstats11.sql.
3. Update PeopleSoft DDL models 4 and 5 for Oracle to call the new package instead of DBMS_STATS.GATHER_TABLE_STATS using
    a. Either data mover script ddlora-gfcpsstats11.dms
    b. Or SQL script ddlora-gfcstats11.sql

Now, other scripts to lock statistics from https://github.com/davidkurtz/psscripts can be implemented

4. Run locktemprecstats.sql to lock and delete statistics on all PeopleSoft temporary records.
5. Run gfc_locktemprecstats_triggerjob.sql to create DDL trigger to submit a job to Oracle job scheduler to call procedure that locks and deletes statistics on tables associated with PeopleSoft temporary records as soon as they are created.
    a. This can be tested with gfc_locktemprecstats_triggerjob_test.sql
6. Run deltempstats.sql to create a trigger on table PS_AETEMPTBLMGR to lock and delete statistics on a temporary table as it is allocated to an Application Engine program.

Meta-Data

PS_GFC_STATS_OVRD

This table contains the meta-data that will be used to create the table preferences. It is keyed on record name. A record will only have table preferences created if a row exists in this table.

This table also controls whether the PS_STATS program in the GFCPSSTATS11 procedure will refresh statistics on a table.

The table is structured in line with PeopleTools so that it could also be created by defining a record in Application Designer and creating it like any other table in a PeopleSoft system.

Column	Datatype	Description
RECNAME*	VARCHAR2(15)	PeopleSoft Record Name
GATHERS_STATS	VARCHAR2(1)	This parameter controls the behaviour of the PS_STATS procedure (see page 12) that is designed to be called from the DDL models that are invoked by the *%UpdateStats* macro G=Gather statistics. This is effectively the same behaviour as the default DDL model. R=Refresh statistics. The *ps_stats* package will only gather statistics if they are marked as stale on the database. N=Don't Gather statistics
ESTIMATE_PERCENT	VARCHAR2(30)	Value passed to ESTIMATE_PERCENT table preference. If blank, any existing table override will be removed and Oracle will revert to default behaviour.
BLOCK_SAMPLE	VARCHAR2(1)	This does not correspond to a table preference. This value is passed to dbms_stats.gather_table_stats when it called by PS_STATS program. Y=TRUE N=FALSE
METHOD_OPT	VARCHAR2(1000)	Value passed to METHOD_OPT table preference. If blank, any existing table override will be removed and Oracle will revert to default behaviour.

GRANULARITY	VARCHAR2(30)	Value passed to GRANULARITY table preference. If blank, any existing table override will be removed and Oracle will revert to default behaviour.
INCREMENTAL	VARCHAR2(5)	Value passed to INCREMENTAL table preference. If blank, any existing table override will be removed and Oracle will revert to default behaviour.
STALE_PERCENT	NUMBER	Value passed to STALE_PERCENT table preference. If blank, any existing table override will be removed and Oracle will revert to default behaviour.

Example Meta-Data

Example Time & Labor Meta-Data

This the meta-data I defined for a Time & Labor system where excessive amounts of time were spent collecting statistics during TL_TIMEADMIN, the main T&L calculation process. We first tried to suppress statistics collection in all Application Engine processes and use only Optimizer Dymanic Sampling. We found that we still needs statistics, though not histograms, on certain tables.

RECNAME	GATHER_STATS	METHOD_OPT
TL_IPT1	G	FOR ALL COLUMNS SIZE 1
TL_MTHCD	G	FOR ALL COLUMNS SIZE 1
TL_PMTCH1_TMP	G	FOR ALL COLUMNS SIZE 1
TL_PMTCH2_TMP	G	FOR ALL COLUMNS SIZE 1
TL_PMTCH_TMP2	G	FOR ALL COLUMNS SIZE 1
TL_PROF_WRK	G	FOR ALL COLUMNS SIZE 1
TL_RESEQ2_WRK	G	FOR ALL COLUMNS SIZE 1
TL_RESEQ5_WRK	G	FOR ALL COLUMNS SIZE 1
TL_WRK01_RCD	G	FOR ALL COLUMNS SIZE 1
WRK_SCHRS_TAO	G	FOR ALL COLUMNS SIZE 1
TL_FRCS_PYBL_TM	R	
TL_ST_PCHTIME	R	
TL_VALID_TR	R	
All other temporary records in TL_TIMEADMIN	N	

This data can be loaded into the meta-data table with the script *gfcpsstats11_metadata.sql*.

DDL Models

This data mover script replaces DDL model 4 and 5 which are called from the Application Engine %UpdateStats macro. Note that both models are the same so it does not matter if the HIGH or LOW parameter is specified.

```
--  ****************************************************************
--  ddlora-gfcpsstats11.dms (c) Go-Faster Consultancy 2012
--  ReLoads the PeopleTools DDL tables for Analyze statements for Oracle
--  calling gfcpsstats11 package
--  ****************************************************************

SET LOG DDLORA.LOG;

DELETE FROM PSDDLMODEL
WHERE PLATFORMID=2
AND STATEMENT_TYPE IN (4,5)
;

INSERT INTO PSDDLMODEL (
STATEMENT_TYPE,
PLATFORMID,
SIZING_SET,
PARMCOUNT,
MODEL_STATEMENT)
VALUES(
:1,
:2,
:3,
:4,
:5)
\
$DATATYPES NUMERIC,NUMERIC,NUMERIC,NUMERIC,CHARACTER
4,2,0,0,$long
gfcpsstats11.ps_stats(p_ownname=>[DBNAME],p_tabname=>[TBNAME],p_verbose=>TRUE);
//
5,2,0,0,$long
gfcpsstats11.ps_stats(p_ownname=>[DBNAME],p_tabname=>[TBNAME],p_verbose=>TRUE);
//
/
```

Updating Statistics in COBOL

PeopleSoft COBOL programs also collect statistics on working storage tables. The stored statements appear to use the UpdateStats macro, but in fact this invokes a hard coded routine in the COBOL that calls DBMS_STATS.

```
STORE PSPLDTL2_U_STATIST
%UPDATESTATS(PS_PY_PYBL_TM_WRK)
;
```

However, these steps can be changed to call a PL/SQL procedure.

```
STORE PSPLDTL2_U_STATIST
BEGIN
gfcpsstats11.ps_stats(p_ownname=>user,p_tabname=>'PS_PY_PYBL_TM_WRK');
END;;
```

One option would be to manually change the delivered scripts that are loaded into the database using Data Mover.

Stored statements that have already been loaded can be updated as follows:

```
UPDATE   ps_sqlstmt_tbl
SET      stmt_text = 'BEGIN gfcpsstats11.ps_stats(p_ownname=>user,p_tabname=>'''
||       SUBSTR(stmt_text,14,LENGTH(stmt_text)-14)
||       '''); END;;'
WHERE    stmt_text LIKE '%UPDATESTATS(%)'
/
```

A trigger can be used to convert the stored statements as they are loaded by data mover (see Trigger GFC_STAT_OVRD_STORED_STMT on page 16).

Packaged Procedure GFCPSSTATS11

The GFCPSSTATS11 package contains a number of public procedures that can be called extermally.

- PS_STATS
- REFRESH_STATS
- SET_TABLE_PREFS
- SET_RECORD_PREFS
- GENERATE_METADATA

NB: This procedure calls the PSFTAPI package; www.go-faster.co.uk/scripts.htm#psftapi.sql.

PS_STATS Procedure

This procedure is designed to be called from the DDL model for %UpdateStats by Application Engine processes and from within Cobol Stored Statements. It collects statistics on the named table according to the meta-data and table preferences.

Syntax

```
ps_stats
(p_ownname      IN VARCHAR2 /*owner of table*/
,p_tabname      IN VARCHAR2 /*table name*/
,p_verbose      IN BOOLEAN DEFAULT FALSE /*if true print SQL*/
);
```

Parameters

Parameter	Data Type	Description
p_ownname	VARCHAR2	Table Owner
p_tabname	VARCHAR2	Table Name
p_verbose	BOOLEAN	If true print debug messages

REFRESH_STATS Procedure

This procedure refreshs unlocked stale statistics on physical tables, partitions and subpartitions and any global statistics.

Syntax

```
set_defaults(
(p_ownname      IN VARCHAR2 /*owner of table*/
,p_tabname      IN VARCHAR2 /*table name*/
,p_block_sample IN BOOLEAN  DEFAULT FALSE /*if true block sample stats*/
,p_force        IN BOOLEAN  DEFAULT FALSE
,p_verbose      IN BOOLEAN  DEFAULT FALSE /*if true print SQL*/
);
```

Parameters

Parameter	Data Type	Description
p_ownname	VARCHAR2	Table Owner
p_tabname	VARCHAR2	Table Name
p_block_sample	BOOLEAN	Same as block_sample from *dbms_stats.gather_table_stats*. Whether or not to use random block sampling instead of random row sampling.
p_force	BOOLEAN	Same as force from *dbms_stats.gather_table_stats*. Gather statistics even if locked.
p_verbose	BOOLEAN	If true print debug messages

SET_TABLE_PREFS Procedure

This procedure sets table preferences on named tables according to the meta-data in PS_GFC_STATS_OVRD.

Syntax

```
set_table_prefs
(p_tabname      IN VARCHAR2 /*table name*/
,p_recname      IN VARCHAR2 DEFAULT NULL /*record of table if known*/
);
```

Parameters

Parameter	Data Type	Description
p_tabname	VARCHAR2	Table Name
p_recname	VARCHAR2	PeopleSoft Record Name

SET_RECORD_PREFS Procedure

This procedure to set table preferences on tables relating to named record to values specified in the meta-data. If there is no meta-data for the record the preferences are removed.

Syntax

```
set_record_prefs
(p_recname          IN VARCHAR2 /*record name*/
);
```

Parameters

Parameter	Data Type	Description
p_recname	VARCHAR2	PeopleSoft Record Name

GENERATE_METADATA Procedure

This procedure updates PS_GFC_STATS_OVRD so that the meta-data matches the actual table preferences.

Syntax

```
generate_metadata;
```

Triggers

A number of triggers have been used to automate the application of table preferences.

Required Privileges

The triggers require that the following privileges are granted explicitly to the SYSADM schema.

```
REM gfcpsstats11_privs.sql
REM (c) Go-Faster Consultancy 2008-2012

GRANT EXECUTE ON dbms_scheduler TO sysadm;
GRANT CREATE JOB TO sysadm;
```

GFC_STATS_OVRD_METADATA

This trigger submits database jobs to reapply the table preferences when the metadata is updated.

```
CREATE OR REPLACE TRIGGER gfc_stats_ovrd_metadata
AFTER INSERT OR UPDATE OF RECNAME, ESTIMATE_PERCENT, METHOD_OPT, DEGREE, GRANULARITY, INCREMENTAL, STALE_PERCENT
             OR DELETE ON ps_gfc_stats_ovrd
FOR EACH ROW
DECLARE
  l_cmd VARCHAR2(100) := '';
  l_jobno NUMBER;
BEGIN
  IF DELETING THEN
    dbms_job.submit(l_jobno,'gfcpsstats11.unset_record_prefs('''||:old.recname||''');');
  ELSE
    IF :new.recname != :old.recname THEN
      dbms_job.submit(l_jobno,'gfcpsstats11.unset_record_prefs('''||:old.recname||''');');
    END IF;
    dbms_job.submit(l_jobno,'gfcpsstats11.set_record_prefs('''||:new.recname||''');');
  END IF;
END gfc_stats_ovrd_metadata;
/
```

Setting and deleting table preferences include an implicit commit. Therefore, I have used DBMS_JOB to submit the jobs because the job is only executed when the trigger issues a commit. Thus, the procedure can just directly read the meta-data from the table. There will be a small lag between commiting meta-data changes, and the table preference being applied.

GFC_STATS_OVRD_CREATE_TABLE

This trigger applies table preferences to the table as it is created. A database job is used because otherwise the trigger fires while the table is being created but it exists.

```
CREATE OR REPLACE TRIGGER gfc_stats_ovrd_create_table
AFTER CREATE ON sysadm.schema
BEGIN
  IF ora_dict_obj_type = 'TABLE' THEN
    --submit one-time job to set table preferences as table will not have been created by time trigger runs
    sys.dbms_scheduler.create_job
    (job_name   => 'SET_PREFS_'||ora_dict_obj_name
    ,job_type   => 'PLSQL_BLOCK'
    ,job_action => 'BEGIN gfcpsstats11.set_table_prefs(p_tabname=>'''||ora_dict_obj_name||'''); END;'
    ,start_date => SYSTIMESTAMP --run job immediately
    ,enabled    => TRUE --job is enabled
    ,auto_drop  => TRUE --request will be dropped when complete
    ,comments   => 'Set table preferneces on table '||ora_dict_obj_owner||'.'||ora_dict_obj_name
    );
  END IF;
END;
/
```

GFC_STAT_OVRD_STORED_STMT

This trigger changes COBOL stored statements to call GFCPSSTATS11 instead of the internal routine that calls DBMS_STATS.

```
CREATE OR REPLACE TRIGGER gfc_stat_ovrd_stored_stmt
BEFORE INSERT ON ps_sqlstmt_tbl
FOR EACH ROW
WHEN (new.stmt_text LIKE '%UPDATESTATS(%)')
BEGIN
  :new.stmt_text := 'BEGIN gfcpsstats11.ps_stats(p_ownname=>user,p_tabname=>'''
                    ||SUBSTR(:new.stmt_text,14,LENGTH(:new.stmt_text)-14)||'''); END;;';
END;
/
```

Delivered Files

File Name	Description
gfcpsstats11.doc	This document.
gfcpsstats11.sql	Script that contains packaged procedure and triggers.
gfcpsstats11_privs.sql	Script to grant privileges to SYSADM schema.
gfcpsstats11_metadata.sql	Sample T&L metadata.
gfcpsstats11_testdata.sql	Script to test behaviour package.
ddlora-gfcpsstats11.dms	Data Mover script to replace DDL Models 4 and 5.