The `pscbo_stats` Package
Frequently Asked Questions

Table of Contents

Frequently Asked Questions

This section explores common questions and typical use cases for the pscbo_stats tool.

1 - How do I add a table to the Statistics Control Table?

The Statistics Control Table, `SYSADM.PSCBO_STATS_CONTROL`, holds a list of tables under the control of the pscbo_stats package. It is very easy to add additional tables to the list that are controlled by pscbo_stats.

Procedure provided:

```
pscbo_stats.stats_control_ins()
```

Sample script (run as SYSADM) - add a single table for dynamic sampling

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-stats_control_ins.log;
exec pscbo_stats.stats_control_ins('PS_TABLENAME');
exec pscbo_stats.gather_table_stats('PS_TABLENAME');
QUIT;
```

Sample script (run as SYSADM) - add multiple tables for dynamic sampling

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-stats_control_ins.log;
exec pscbo_stats.stats_control_ins('PS_TABLENAME1');
exec pscbo_stats.stats_control_ins('PS_TABLENAME2');
exec pscbo_stats.sync_schema_stats();
QUIT;
```

Again, notice that you must initiate the gathering of statistics for that table(s) separately after executing this procedure. This procedure will not automatically re-gather the statistics on the associated table.

2 - How do I modify an entry in the Statistics Control Table?

The Statistics Control Table, `SYSADM.PSCBO_STATS_CONTROL`, is intended to be a repository of all tables that were - at one time - intended for dynamic sampling. By design, it is intended that any tables once flagged for dynamic sampling remain in this table as an historical record.

If, after flagging a table for dynamic sampling, you determine that only static sampling should again be used for the table, use the set_stats_control_flag() procedure. By setting the flag to "Y" you indicate the desire to statically sample the table: "Y" means YES to statistics. A non-blank reason is required so a record of this decision will be maintained.

Note: Do not use an SQL Query tool to simply remove entries from this table. During its differential compare processing, the sync_schema_stats() procedure may repopulate it, resulting in dynamic sampling again being imposed on the object.

Sample script (run as SYSADM)

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-stats_control_flag.log;
```

```
EXEC pscbo_stats.set_stats_control_flag ('PS_TABLENAME','Y', 'dyn sampling
impacted GPPDPRUN processing negatively');
QUIT;
```

3 - How do I exclude a Table from pscbo_stats control?

To prevent pscbo_stats from controlling the statistics on a table, simply lock the statistics of the table. The `gather_table_stats()` procedure will honor the locked statistics and exit gracefully. Obviously, you will need to make other arrangements to manage the statistics on this locked table.

Procedure provided:

```
dbms_stats.lock_table_stats -- <- note 'dbms_stats'
```

Sample script (run as SYSADM)

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-lock_table_stats.log;
EXEC dbms_stats.lock_table_stats('PS_TABLENAME');
QUIT;
```

4 - What do I need to know about using the Histogram Exception Table?

Histogram gathering differences between Oracle 10g and 11g

There are differences in the ways that Oracle 10g and Oracle 11g manage histograms. The Histogram Exception Table was specifically added to allow for granular control of histograms in Oracle 10g, but this feature will also work in Oracle 11g. But in Oracle 11g, we recommend the use of table preferences to control specific use cases of histograms.

Note: Typically, in Oracle 11g, the default settings for histogram collection are adequate.

In Oracle 11g, if `gather_table_stats()` does not find an entry in the Histogram Exception Table, then METHOD_OPT = AUTO will be used when gathering statistics for that table. The use of AUTO implies that Oracle table preferences will control this value. If the procedure finds an entry in the Histogram Exception Table, then ONLY that column, or columns, will have a histogram gathered and the sample size will be 100%. Naturally, in the case where a histogram exception is identified, then the table preference will *not* be used.

In Oracle 10g, histograms will not be collected by default, i.e. METHOD_OPT=1 is used. If the procedure finds an entry in the Histogram Exception Table, then ONLY that column, or columns, will have a histogram gathered and the sample size will be 100%.

Adding Table/Columns to the Histogram Exception Table

To indicate the collection of histogram data from a table/column combination, use the `histogram_col_ins()` procedure to add the row from the Histogram Exception Table.

Procedure provided:

```
pscbo_stats.histogram_col_ins()
```

Sample script (run as SYSADM):

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-insert_histogram_exception_table.log;
EXEC pscbo_stats.histogram_col_ins ('PS_TABLENAME','COLUMN_NAME');
EXEC pscbo_stats.gather_table_stats('PS_TABLENAME');
SPOOL OFF;
```

Note: The `histogram_col_ins()` procedure will not automatically re-gather the statistics on the associated table. You must explicitly gather the statistics for that table after executing this procedure.

Removing Table/Columns to the Histogram Exception Table

To stop the collection of histogram data from a table/column combination by the pscbo_stats package, use the `histogram_col_del()` procedure to remove the row from the Histogram Exception Table.

Procedure provided:
`pscbo_stats.histogram_col_del()`

Sample script (run as SYSADM):

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-delete_histogram_exception_table.log;
EXEC pscbo_stats.histogram_col_del ('PS_TABLENAME','COLUMN_NAME1');
EXEC pscbo_stats.histogram_col_del ('PS_TABLENAME','COLUMN_NAME2');
EXEC pscbo_stats.gather_table_stats('PS_TABLENAME');
SPOOL OFF;
```

Note: The `histogram_col_ins()` procedure will not automatically re-gather the statistics on the associated table. You must explicitly gather the statistics for that table after executing this procedure.

5 - How is the log file size managed?

Each time that statistics are gathered for a given table, an entry is made into the `SYSADM.PSCBO_LOG` table. When the `gather_schema_stats()` procedure runs, it purges values from this table that are more than 50 days old, which is typically adequate to control this table's size. (This interval is hard-coded in the package and not externally configurable.) Should this interval be too long, directly manipulating the table with SQL is necessary.

Sample script (run as SYSADM)

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO pscbo_stats-truncate_pscbo_log_table.log;
TRUNCATE PSCBO_LOG; -- sample for syntax only, not recommended!!
QUIT;
```

6 - How do I gather Database Dictionary Statistics?

Since `gather_schema_stats` gathers statistics for only PeopleSoft application tables, dictionary statistics must be gathered separately. Since the AUTOSTATS job has been disabled and retargeted, this needs to be done manually, or scheduled separately.

If a large amount of changes has been made to the Oracle data dictionary, e.g. a major PeopleSoft

Application migration has occurred, gather those statistics manually.

Procedure provided:

```
dbms_stats.gather_dictionary_stats -- <- note 'dbms_stats'
```

Sample script (run as SYSDBA)

```
SET SERVEROUT ON TRIMS ON LINES 1000;
SPO gather_dict_stats.log;
EXEC dbms_stats.gather_dictionary_stats;
QUIT;
```

7 - What options are used to gather statistics in the pscbo_stats package?

Default Options

By default, on Oracle 11g when table-level statistics are gathered, the following options are used:
- a.) on all columns,
- b.) with histograms,
- c.) auto sample size
- d.) no_invalidate=FALSE

By default, on Oracle 10g when table-level statistics are gathered, the following options are used:
- a.) on all columns,
- b.) with no histograms
- c.) sample size per the "Sample Size Override Schedule", i.e.

```
100% for tables < 1M rows
 30% for tables up to 10M rows
 10% for tables up to 100M rows
  3% for tables up to 1B rows
  1% for tables > 1B rows
```

- d.) no_invalidate=FALSE

Histograms

If a histogram is specified in the histogram exception table, then histograms will be collected on that column with `SIZE=254`; otherwise `SIZE=1` is used. If no histogram exception is found and a table preference has been used (Oracle 11g), then the preference will control the histogram for that table.

Non-Stale Override Schedule

During the normal collection of schema statistics, to improve performance the pscbo_stats package does not gather statistics for objects that are not marked as stale. But there is one very important exception. If the statistics are marked as “not stale” for a reasonable length of time, the package will force the regathering of them.

The reason for this override is primarily for relatively large tables. By default, DBMS_STATS considers a whole table as stale only when the number of inserts, updates and deletes exceeds 10% of the number of rows in the table. But on large transaction tables, both the DML activity and the query activity can be concentrated in the most recently created extent(s) for a given table segment. In reality, the characteristics of the data in the newer blocks in these new extents may diverge in a statistically significant fashion from the meta-data that is visible to the optimizer, even though the entire table is not yet marked as stale.

To handle this situation, the `gather_schema_stats()` procedure forces the gathering of statistics based on the following "Non-Stale Sampling Override Schedule" that has been established as a “best practice” by Oracle Software Support:

```
every 3 weeks for tables < 1M rows
every 4 weeks for tables up to 10M rows
every 5 weeks for tables up to 100M rows
every 6 weeks for tables up to 1B rows
every 7 weeks for tables > 1B rows
```

8 - How do I know if I need to populate the various configuration tables?

Two configuration tables can be populated to tailor the `pscbo_stats` package to meet specific needs. Additionally, locking the table's statistics prevents the package from managing the statistics on that table.

Statistics Control Table

The setup process populates this table with many tables. Add additional tables to this list when you observe the following characteristics:

a.) a small number of rows, i.e. less then 1M rows,
b.) statistics often go stale between the normal gathering of statistics.
c.) are used in predicates, or WHERE clauses, *after* they are populated and *before* the statistics are normally updated.
d.) are not used in normally very quickly executing statements in large loops.

Examples:

- A small set up table that doesn't go stale often does not make good candidates for dynamic sampling. Static sampling may be a better approach in this case.
- Dynamic sampling can impact the run time of statements that execute very quickly in a large looping structure where the plan does not need to change during the loop. Static sampling may be a better approach in this case.
- Tables used to pass information from one program to another could make particularly good candidates, especially if there is a high degree of concurrency of program execution.

Histogram Exception Table

Typically, histograms do not need to be individually controlled. The defaults are usually adequate and need to be overridden only with a specific, reproducible test case.

Exclusions/Locked Status

If cases exist where the capabilities of the `pscbo_stats` package are not sufficient, it may be necessary to exclude the table from `pscbo_stats` processing.

To do this, simply lock the statistics of the table and the `gather_schema_stats()` procedure will skip them. Ensure that arrangements are made to manage the statistics external to the pscbo_stats package.

9 - What is the difference between `setup_schema_stats()`, `sync_schema_stats()`, and `gather_schema_stats()`?

To answer this question, two aspects of statistics management need to be addressed:

a.) validate the Statistics Control Table - This table needs to be differentially (re)populated to ensure that any new tables intended for dynamic sampling are added and any missing tables are removed.
b.) update table and index statistics - Statistics need be be gathered on all tables and their associated indexes when the table is flagged as being "stale." Additionally, for tables indicated for dynamic sampling, those statistics need to be dropped.

The three procedures work as follows:

- `gather_schema_stats()` - Gathers statistics on stale tables and associated indexes that are not locked or marked for dynamic sampling. There is an optional parameter to force the collection of statistics on all tables, stale or not.
- `sync_schema_stats()` - Validates the Statistics Control Table and temp tables' statistics status, and then calls `gather_schema_stats()` for STALE tables only and the associated indexes.
- `setup_schema_stats()` - Validates the Statistics Control Table and temp tables' statistics status, and then calls `gather_schema_stats()` for ALL tables and associated indexes., stale or not

After the installation is complete, `sync_schema_stats()` is the procedure typically employed to ensure that the system is configured correctly. It may be scheduled instead of `gather_schema_stats()`. since the addition time to validate the locked status is normally less than 1 minute.

10 - How can I improve the speed of statistics gathering?

On Oracle 11g, table preferences are honored for DEGREE. One common way to improve statistics gathering is to increase the default of DEGREE used for large objects. This needs to be set before the `pscbo_stats.setup_schema_stats()` is run.

Procedure provided:

```
dbms_stats.set_table_prefs -- <- note 'dbms_stats'
```

Sample usage:

```
exec dbms_stats.set_table_prefs('SYSADM','PS_BIGTBL','DEGREE','8');
```

Sample script to identify the largest objects (run as SYSADM):

```
SET SERVEROUT ON TRIMS ON LINES 1000 PAGES 50000;
SPO block_counts.out;
SELECT BLOCKS, TABLE_NAME
FROM DBA_TABLES
WHERE BLOCKS > 100000
AND OWNER = 'SYSADM'
ORDER BY BLOCKS
```

```
;
SPOOL OFF;
```

Sample script to generate a script for altering the preferences (run as SYSADM):

```
-- Run output file as a script as SYSADM
-- modify degree, num blocks, and owner as appropriate
SET SERVEROUT ON TRIMS ON LINES 1000 PAGES 50000;
SPO set_prefs_degree.sql;
SELECT 'exec DBMS_STATS.SET_TABLE_PREFS(''SYSADM'', ''' ||TABLE_NAME
||''', ''DEGREE'', ''8'');'
AS SCRIPT_TEXT
FROM DBA_TABLES
WHERE BLOCKS > 100000
AND OWNER = 'SYSADM'
;
SPOOL OFF;
-- for more detail, refer to:
--
http://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_stats.htm#sthref8804
```

11 - What table preferences are honored by the pscbo_stats package?

On Oracle 11g, the following table preferences are honored and not overridden by the package

- GRANULARITY for sub-partition support
- METHOD_OPT for histogram support, and
- DEGREE for speed of statistics gathering.
- ESTIMATE_PERCENT for altering the sample size (not recommended in any version)

12 - What attributes are considered when automatically populating the Statistics Control Table?

During the `pscbo_stats.setup_schema_stats()` procedure, two different groups of tables are inserted into the statistics control table: *"AE" Tables and "WK" Tables*.

"AE" Tables

Tables used in Application Engine as temporary storage are algorithmically identified based on their record type and inserted into the statistics control table. These tables are identified algorithmically based on a series of queries involving:

- PSRECDEFN.RECTYPE=7,
- PSOPTIONS.TEMPTBLINSTANCES, and
- PSAEAPPLDEFN. TEMPTBLINSTANCES.

These entries have a `PSCBO_STATS_CONTROL.TABLE_TYPE='AE'` to differentiate them from tables added manually. These rows should never be deleted manually from this table as they will be automatically inserted when the `sync_schema_stats()` or `sync_schema_stats()` is run. These tables can be marked to disable dynamic sampling using the `pscbo_stats.set_stats_control_flag()` procedure.

"WK" Tables

When tables other than the previously described "AE" tables are added to the statistics control table, they will be inserted with a `PSCBO_STATS_CONTROL.TABLE_TYPE='WK'`. These inserts can occur in one of two places, during installation with the `sync_stats_control_table()` procedure, or manually with the `stats_control_ins()` procedure.

The pscbo_stats package attempts to identify tables that "look" like working storage. It filters the `recname` for various naming conventions used for volatile tables or working storage by programs such as COBOL and nVision. The query used looks like this:

```
SELECT owner, dd_table_name, 'WK', 'N'
  FROM pscbo_psrecdefn r, sys.dba_tables t
WHERE
  (
    r.recname LIKE '%TMP'
    OR r.recname LIKE '%TMP_'        -- e.g. REC_TMP1
    OR r.recname LIKE '%TMP__'       -- e.g. REC_TMP12
    OR r.recname LIKE '%TMP!__'  ESCAPE '!' -- e.g. REC_TMP_1
    OR r.recname LIKE '%TMP!___' ESCAPE '!' -- e.g. REC_TMP_12
    OR r.recname LIKE '%TEMP'
    OR r.recname LIKE '%TEMP_'        -- e.g. REC_TEMP1
    OR r.recname LIKE '%TEMP__'       -- e.g. REC_TEMP12
    OR r.recname LIKE '%TAO'
    OR r.recname LIKE '%TAO_'         -- e.g. REC_TAO1
    OR r.recname LIKE '%TAO__'        -- e.g. REC_TAO12
    OR r.recname LIKE '%WRK'
    OR r.recname LIKE '%WRK_'         -- e.g. REC_WRK1
    OR r.recname LIKE '%WRK__'        -- e.g. REC_WRK12
    OR r.recname LIKE 'PSTREESELECT%' -- nVision Tree Selects
  )
  AND r.recname NOT LIKE 'AE%'
  AND r.recname NOT LIKE '%VW'
  AND r.recname NOT LIKE '%TMPLT'
  AND r.recname NOT LIKE '%WL'
  AND r.recname NOT LIKE '%MC'
  AND r.rectype = 0                -- SQL Tables, not type 7
  AND t.owner = l_owner
  AND t.table_name = dd_table_name
MINUS                              -- to avoid duplicates
  SELECT lowner, dd_table_name, 'WK', 'N'
    FROM pscbo_stats_control;
```

Notice that these entries are inserted with `PSCBO_STATS_CONTROL.TABLE_TYPE='WK'`. Once inserted, these tables can be marked to disable dynamic sampling using the `pscbo_stats.set_stats_control_flag()` procedure.

13 - What options should I use when gathering statistics on locked/excluded tables?

While there are several nuances, the following script will capture the essence of the way that pscbo_stats gathers statistics on Oracle 11g.

```
exec DBMS_STATS.GATHER_TABLE_STATS(
  ownname => 'SYSADM',
  tabname => 'PS_TABLE',
  estimate_percent => NULL, -- defaults to AUTO SAMPLE SIZE
  method_opt => 'FOR ALL COLUMNS SIZE AUTO',
  degree => SYS.DBMS_STATS.DEFAULT_DEGREE,
  granularity => 'AUTO',
  cascade => TRUE,
  no_invalidate => FALSE,
  force => TRUE  -- if needed to override a lock
);
```

For Oracle 10g, the following options should be used:

```
exec DBMS_STATS.GATHER_TABLE_STATS(
  ownname => 'SYSADM',
  tabname => 'PS_TABLE',
  estimate_percent => 100 -- See FAQ 7 for Sample Size Override Schedule
  method_opt => 'FOR ALL COLUMNS SIZE 1',
  degree => SYS.DBMS_STATS.DEFAULT_DEGREE,
  granularity => 'AUTO',
  cascade => TRUE,
  no_invalidate => FALSE,
  force => TRUE
);
```

14 - What side effects exist with the use of pscbo_stats package?

There are a few "corner cases" where program behavior can degrade after the use of the pscbo_stats package. Generally, any side effects are the result of the dynamic sampling uncovering either a weakness in the underlying code or exposing a limitation of the optimizer.

SQL with morphing table contents and no binds

This corner case can be caused when a SQL statement has ALL the following characteristics:

a.) uses no bind variables,
b.) involves a temp table that is being dynamically sampled,
c.) %UpdateStats() was previously issued on the temp table, and
d.) the temp table data varies significantly between subsequent executions of the statement

When statistics are gathered on an object, all plans associated with that object are invalidated. When the data in the table changes significantly, but there is nothing to "alert" the optimizer of that change, it will typically reuse the existing plan. This is part of the design of the optimizer.

This symptom has been seen when COBOL Stored Statements. Here are some suggestions to remediate this side effect:

- COBOL Stored Statement - When this occurs, the best solution is to disable the dynamic sampling on that temp object and allow the %UpdateStats() to execute. This will force the plan to recompile and avoid the issue.

- Application Engine - Add a hint such as **`/* %UuidGen() */`** to the statement causes the SQL to change between each run and forces the statement to recompile.

SQL using binds that indicate significantly different ranges of data in the same context

This corner case can be caused when a SQL statement has ALL of the following characteristics:

a.) the statement uses bind variables to indicate ranges, e.g. EMPLID BETWEEN :1 and :2, or WHERE EFFDT < :1,
b.) once parsed, the statement subsequently executes with bind values that select significantly different amounts of data
c.) %UpdateStats() was previously issued on the temp table, and
b.) `pscbo_stats.gather_table_stats()` prevents the updating of statistics of a table in the query that was previously statically sampled.

Oracle will consider or "peek" at the binds when compiling a plan. These binds are called the "peeked binds." As a result, the first execution of such a statement is typically good based. Bind variables used to execute an existing plan are called "captured binds." If the selectivity of *peeked* binds differ significantly from *captured* binds, then the plan *can* subsequently perform quite poorly.

A feature introduced in Oracle 11g called Adaptive Cursor Sharing addresses this situation. But, the algorithms used to signal the need for multiple plans is very conservative, so much so that in some in PeopleSoft, it is often ineffective.

The solution is to instruct the optimizer to compile the statement to be "bind aware," accelerating the process that would naturally occurs in the optimizer. This change can be done at the statement level with the /*+ BIND_AWARE */ hint. A session-level methodology is explained on the following Note:

"E-COB Addressing SQL Performance issue with Global Payroll (GPPDPRUN) on Oracle 11g (Doc ID 1462442.1)

When this characteristic is not properly understood, it is common to see excessive gathering of statistics. in an attempt to force the invalidation of the execution plans. A table under control of pscbo_stats and dynamic sampling will expose this overuse, since that plan will not be invalidated as expected.

<paginated for readability>

15 - How can I detect which tables are going stale?

The following script will display the “stale factor” when greater than 3% as calculated by:

```
(DELETES + UPDATES + INSERTS) / NUM_ROWS * 100)
```

By default, a table is considered as “stale” when this value reaches 10%, so rows will be returned by this query before they are considered stale by the `dbms_stats.gather_schema_stats()` procedure.

Sample script (run as SYSDBA):

```
SET SERVEROUT ON TRIMS ON LINES 1000 PAGES 50000;
SPO show_stale_tables.out;
COL OWNER FORMAT A10;
COL TABLE_NAME FORMAT A30;
COL TIMESTAMP FORMAT A16;
SET LINESIZE 9999

SELECT DT.OWNER,  DT.TABLE_NAME,  DELETES,  UPDATES,  INSERTS,  NUM_ROWS,
  ROUND ( (DELETES + UPDATES + INSERTS) / NUM_ROWS * 100) PERCENTAGE,
  TO_CHAR(ATM.TIMESTAMP, 'MM-DD-YY HH24:MI') TIMESTAMP,
  TRUNC((SYSDATE - ATM.TIMESTAMP),1) DAYS_SINCE,
  TRUNCATED
FROM DBA_TABLES DT,
  ALL_TAB_MODIFICATIONS ATM
WHERE DT.OWNER = ATM.TABLE_OWNER
AND DT.TABLE_NAME  = ATM.TABLE_NAME
AND NUM_ROWS  > 0
AND ROUND ( (DELETES + UPDATES + INSERTS) / NUM_ROWS * 100) >= 3
AND OWNER NOT IN ('SYS','SYSTEM','DBSNMP','OSMMON','PERFSTAT')
ORDER BY 7 DESC;
SPOOL OFF;
```

<paginated for readability>

16 - Which tables are being dynamically sampled?

It is very helpful to know which tables are consuming dynamic sampling resources. For example, a large table that does not statistically change would be better handled with static statistics, or a very small table in a very tight loop may perform poorly because of the dynamic sampling overhead. This script can help identify those conditions exist and equip you to make a customizations to better handle these corner cases. (credit: Andres L.)

Sample script (run as SYSDBA):

```
col OWNER format a15
col TABLE_NAME format a28

select p.interval "DATE", sum(s.executions_delta) dyn_samp_ops
  from dba_hist_sqltext t, DBA_HIST_SQLSTAT s, (
  select distinct DBID, SNAP_ID, trunc(BEGIN_INTERVAL_TIME) interval
    from DBA_HIST_SNAPSHOT
   where DBID = (select DBID from v$database)) p
 where t.sql_text like 'SELECT /* OPT_DYN_SAMP */%'
   and s.sql_id = t.sql_id
   and p.SNAP_ID = s.SNAP_ID
   and p.DBID = s.DBID
 group by p.interval
 order by 1
/
select h.owner, h.table_name, h.dyn_samp_ops, g.blocks blocks_alloc,
t.last_analyzed
  from (
  select l.table_name, l.owner, sum(s.executions_delta) dyn_samp_ops
    from (
    select t.sql_id, p.OBJECT_NAME table_name, p.OBJECT_OWNER owner
      from dba_hist_sqltext t, DBA_HIST_SQL_PLAN p
     where t.command_type = 3
       and t.sql_text like 'SELECT /* OPT_DYN_SAMP */%'
       and p.sql_id = t.sql_id
       and p.object_type = 'TABLE'
         ) l
       , DBA_HIST_SQLSTAT s
       , (
    select distinct DBID, SNAP_ID
      from DBA_HIST_SNAPSHOT
     where DBID = (select DBID from v$database)
       and trunc(BEGIN_INTERVAL_TIME) = '&date') p
     where s.sql_id = l.sql_id
       and p.SNAP_ID = s.SNAP_ID
       and p.DBID = s.DBID
     group by table_name, owner) h, dba_segments g, dba_tables t
 where g.segment_name = h.table_name
   and g.owner = h.owner
   and g.segment_type = 'TABLE'
   and t.table_name = h.table_name
   and t.owner = h.owner
 order by 3 desc
/
```

17 - How can I query the `PSCBO_LOG` table?

The following sample queries demonstrate ways to extract various useful data from the log table.

```
-- show when major events were completed
SELECT * FROM PSCBO_LOG
WHERE LINE_TYPE = 'S'  -- summary lines only
AND TSTAMP > (SYSDATE - 7) -- in the last week
AND LINE LIKE '%<=%'  -- will show completions of gather_schema_stats
ORDER by tstamp ASC;

-- show how long stats take to be collected.
SELECT * FROM PSCBO_LOG
WHERE LINE_TYPE = 'S'
AND LINE LIKE '%=%gather_schema_stats%'
ORDER by tstamp ASC;

-- select activity in the 5 seconds
SELECT * FROM PSCBO_LOG
WHERE TSTAMP > (SYSDATE - 5/24/60/60)
-- AND LINE_TYPE = 'S' -- only summary lines
AND LINE_TYPE = 'D' -- only detail lines
ORDER by tstamp ASC;

-- show the most recent 10 lines of the log
SELECT * FROM
  (SELECT * FROM PSCBO_LOG ORDER BY TSTAMP DESC)
WHERE rownum <= 10;

-- show the most recent entry for a given table
SELECT * FROM
  (SELECT *
  FROM PSCBO_LOG
  WHERE line LIKE '%tabname => ''"PSOPRDEFN"''%'
  AND line_type = 'D'
  ORDER BY TSTAMP DESC)
WHERE rownum <= 1;

-- show configuration data
SELECT * FROM PSCBO_LOG
WHERE LINE_TYPE = 'S'
AND (LINE LIKE '%user:%'
OR LINE LIKE '%owner:%'
OR LINE LIKE '%rdbms:%'
OR LINE LIKE '%pscbo:%'
OR LINE LIKE '%execute:%')
ORDER BY TSTAMP
;
```

18 - How can I schedule pscbo_stats to run automatically?

The following scripts demonstrate how to schedule a simple job to run `PSCBO_STATS.SYNC_SCHEMA_STATS()` on a daily basis.

(Run as SYSDBA)

```
-- schedule a simple job to run  PSCBO_STATS.SYNC_SCHEMA_STATS
-- change the start date to desired execution times
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name        => 'SYSADM.SYNC_SCHEMA_STATS',
  job_type        => 'PLSQL_BLOCK',
  job_action      => 'BEGIN PSCBO_STATS.SYNC_SCHEMA_STATS(''SYSADM''); END;',
  start_date      => '01-JAN-14 5.00.00AM US/Eastern',
  repeat_interval => 'FREQ=DAILY',
  enabled         =>  TRUE,
  comments        => 'PSCBO_STATS NIGHTLY SYNC AND GATHER SCHEMA STATS');
END;

-- sample query to view job details
SET SERVEROUT ON TRIMS ON LINES 1000;
COL OWNER FORMAT A8;
COL JOB_NAME FORMAT A20;
COL START_DATE FORMAT A20;
COL REPEAT_INTERVAL FORMAT A12;
COL JOB_ACTION FORMAT A80;
SELECT OWNER, JOB_NAME, TO_CHAR(START_DATE, 'MM/DD/YYYY HH24:MI') START_DATE,
REPEAT_INTERVAL, JOB_ACTION
FROM DBA_SCHEDULER_JOBS WHERE
JOB_NAME = 'SYNC_SCHEMA_STATS';

-- sample command to remove existing job
EXEC DBMS_SCHEDULER.DROP_JOB('SYSADM.SYNC_SCHEMA_STATS', TRUE);
```

For additional details see:
http://docs.oracle.com/cd/E18283_01/server.112/e17120/schedadmin005.htm

<paginated for readability>

19 - How can I better understand how pscbo_stats has configured the various tables?

The Statistics Control Table holds most of the information used to control the gathering of statistics. The following script displays examples how the Stats Control Table is configured, how many tables are locked, how many have stats deleted so dynamic sample will occur, and version and perhaps some installation details (if not yet removed).

(Run as SYSADM)

```
SPO pscbo_stats_cfg_info.out;
SET SERVEROUT ON TRIMS ON LINES 1000 PAGES 50000;
COL OWNER FORMAT A10;
COL TABLE_NAME FORMAT A30;
COL TIMESTAMP FORMAT A16;
SET LINESIZE 9999
PRO CONFIRM NUM OF ROWS, WK = WORK TABLES, AE = RELATED TO RECTYPE=7
PRO
SELECT TABLE_TYPE, COUNT(*)
FROM PSCBO_STATS_CONTROL
GROUP BY TABLE_TYPE;
PRO
PRO DISPLAY LOCKED STATUS - SHOULD BE ALL SINCE IT IS LOCKED
PRO
SELECT STATTYPE_LOCKED, COUNT(*) FROM DBA_TAB_STATISTICS
WHERE OWNER LIKE 'SYSADM'
GROUP BY STATTYPE_LOCKED;
PRO
PRO DISPLAY GLOBAL_STATS STATUS - YES MEANS STATIC STATS, NO MEANS DYN STATS
PRO
SELECT GLOBAL_STATS, COUNT(*) FROM DBA_TAB_STATISTICS
WHERE OWNER LIKE 'SYSADM'
GROUP BY GLOBAL_STATS;
PRO
SPOOL OFF
```

20 - Why is it necessary to both disable the AUTOTASK and re-target it to ORACLE? Isn't that redundant?

See FAQ #21.

21 - What is the correct configuration of AUTOTASK with pscbo_stats?

Prior to v3.1 the `pscbo_stats.gather_schema_stats()` procedure would log an error when the AUTOTASK was not disabled. During installation, the task was disabled and the target altered to reduce the risk that the task might gather the statistics on the dynamically sampled tables.

To allow some flexibility, it was communicated both on the forums and interactively that enabling the AUTOTASK for statistics collection would function correctly *if and only if* the target were "ORACLE." The logging code would still log the error message.

In v3.1, a change was made to recommend the AUTOTASK remain enabled. An error is logged only if the target is not ORACLE. The status and target of the AUTOTASK is logged with each execution of `pscbo_stats.gather_schema_stats()`.

Again, the recommendation is to leave the task enabled, but alter the target to be "ORACLE." Step 2.4

of the implementation instructions reflects this change.

22 - How can I add PeopleTools Run Control tables to pscbo_stats control?

Some users have benefited by dynamically sampling statistics of PeopleTools tables, i.e. those that hold run control information. These tables run into the hundreds. Often they are empty and populated only when there is an active, running process, but there is global mechanism for updating those stats when they are populated.

The following script attempts to identify those tables - that also have less than 5 records - and adds them to the stats control table. The scrip can be rerun to incrementally add new tables since it relies on `USER_TABLES.NUM_ROWS` as a filter. Remember to regather the statistics to ensure that the stats have been dropped. (credit: Paul M.)

Periodically running the script in FAQ #16 after running this script is highly recommended to ensure that the impact of dynamic sampling operations is not excessive.

(Run as SYSADM)

```
-- the next three settings produce a cleaner script, reset when done
SET PAGES 0
SET FEEDBACK OFF
SET HEADING OFF
SPO pscbo_stats-addPtRunCtlTbls.out;

SELECT 'exec PSCBO_STATS.STATS_CONTROL_INS('''  || TABLE_NAME  || ''');'
FROM USER_TABLES
WHERE TABLE_NAME IN
  (SELECT PSCBOREC.DD_TABLE_NAME
  FROM PSCBO_PSRECDEFN PSCBOREC, PS_PRCSDEFNPNL PRCPNL, PSPNLGROUP PNLGRP,
    PSPNLFIELD PNLFLD, PSRECDEFN RECDEFN
  WHERE PSCBOREC.RECNAME  = RECDEFN.RECNAME
  AND PNLGRP.PNLGRPNAME = PRCPNL.PNLGRPNAME
  AND PNLFLD.PNLNAME    = PNLGRP.PNLNAME

  AND PNLFLD.RECNAME   <> ' '
  AND PNLFLD.FIELDNAME <> ' '
  AND PNLFLD.RECNAME    = RECDEFN.RECNAME
  AND RECDEFN.RECTYPE       = 0
  AND EXISTS
    (SELECT 'X'
    FROM PSRECFIELD
    WHERE RECNAME = PNLFLD.RECNAME
    AND FIELDNAME = 'OPRID'
    AND FIELDNUM  = 1)
  AND PSCBOREC.DD_TABLE_NAME NOT IN
    (SELECT TABLE_NAME FROM PSCBO_STATS_CONTROL)
  )
AND NUM_ROWS < 5
ORDER BY 1
/
-- Remember to sync (read: drop) the statistics for the newly added tables
PRO exec PSCBO_STATS.SYNC_SCHEMA_STATS()
PRO /
PRO
SPOOL OFF
```