197
DBMS_STATS

With the `DBMS_STATS` package you can view and modify optimizer statistics gathered for database objects.

Users can also collect statistics on Global Temporary Tables (GTTs) using the DBMS_STATS package. However, `DBMS_STATS` cannot collect statistics on Private Temporary Tables (PTTs).

This chapter contains the following topics:

> **See Also:**
>
> - *Oracle Database SQL Tuning Guide*
> - *Oracle Database Administrator's Guide*

DBMS_STATS Overview

To improve performance, the database enables you to collect optimizer statistics.

> **Note:**
>
> By default, the database collects statistics automatically, so this package is intended only for specialized cases.

Optimizer statistics can reside in the data dictionary or in a table created in the user's schema. You can also collect and manage user-defined statistics for tables and domain indexes using this package. For example, if you invoke the `DELETE_COLUMN_STATS` procedure on a column for which an association is defined, the database deletes both user-defined and standard statistics for this column.

Only optimizer statistics stored in the data dictionary have an effect on the cost-based optimizer. You can also use `DBMS_STATS` to gather statistics in parallel.

Optimizer Statistics Advisor inspects the statistics gathering process, automatically diagnoses problems in the existing practices for gathering statistics, and then generates a report of findings and recommendations. The advisor task runs automatically in the maintenance window. However, you can also run the job on demand.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to manage optimizer statistics

DBMS_STATS Deprecated Subprograms

Oracle recommends that you do not use deprecated subprograms. Support for deprecated features is for backward compatibility only.

The following subprograms are obsolete:

- GET_PARAM Function

  Instead, use GET_PREFS Function

The following subprogram is deprecated:

- `GENERATE_STATS`

  This procedure is replaced by the `GATHER_INDEX_STAT` procedure.

> **See Also:**
>
> "GATHER_INDEX_STATS Procedure"

DBMS_STATS Types

The following are DBMS_STATS types for histograms, stale tables, statistics difference reports, and optimizer statistics advisor.

Histograms

Types for the minimum and maximum values and histogram endpoints include the following:

```
TYPE numarray  IS VARRAY(2050) OF NUMBER;
TYPE datearray IS VARRAY(2050) OF DATE;
TYPE chararray IS VARRAY(2050) OF VARCHAR2(4000);
TYPE rawarray  IS VARRAY(2050) OF RAW(2000);
TYPE fltarray  IS VARRAY(2050) OF BINARY_FLOAT;
TYPE dblarray  IS VARRAY(2050) OF BINARY_DOUBLE;
```

Stale Tables

Types for listing stale tables include the following:

```
TYPE ObjectElem IS RECORD (
  ownname     VARCHAR2(30),     -- owner
  objtype     VARCHAR2(6),      -- 'TABLE' or 'INDEX'
  objname     VARCHAR2(30),     -- table/index
```

```
  partname    VARCHAR2(30),     -- partition
  subpartname VARCHAR2(30));    -- subpartition
TYPE ObjectTab IS TABLE OF ObjectElem;
```

Statistics Difference Reports

Use the following type to displays a statistics difference report:

```
TYPE DiffRepElem IS RECORD (
  report      CLOB,            -- stats difference report
  maxdiffpct  NUMBER);         -- max stats difference (percentage)
TYPE DiffRepTab IS TABLE OF DiffRepElem;
```

Optimizer Statistics Advisor

The following type represents database objects for which you can gather statistics:

```
TYPE ObjectElem IS RECORD (
  ownname     dbms_quoted_id, -- owner
  objtype     VARCHAR2(6), -- 'TABLE' or 'INDEX'
  objname     dbms_quoted_id, -- table/index
  partname    dbms_quoted_id, -- partition
  subpartname dbms_quoted_id -- subpartition
);
TYPE ObjectTab IS TABLE OF ObjectElem;
```

> **Note:**
>
> Make sure to maintain `satisfy_obj_filter` when the `ObjectElem` type is changed

The following type represents an operation:

```
TYPE StatsAdvOpr IS RECORD (
  name  VARCHAR2(64), -- name of the operation
  param VARCHAR2(4000)
); -- XML containing parameters and their values
TYPE StatsAdvOprTab IS TABLE OF StatsAdvOpr;
```

The following type represents a filter list:

```
TYPE StatsAdvFilter IS RECORD (
  rulename VARCHAR2(64),   -- rule name
  objlist  ObjectTab,      -- object filter list
  oprlist  StatsAdvOprTab, -- operation filter list
  include  BOOLEAN);       -- include/exclude elements in the list
TYPE StatsAdvFilterTab IS TABLE OF StatsAdvFilter;
```

DBMS_STATS Constants

The `DBMS_STATS` package defines several constants to use specifying parameter values.

Table 197-1 DBMS_STATS Constants

Name	Type	Description
ADD_GLOBAL_PREFS	NUMBER	Copies global preferences
AUTO_CASCADE	BOOLEAN	Lets Oracle decide whether to collect statistics for indexes or not
AUTO_DEGREE	NUMBER	Lets Oracle select the degree of parallelism based on size of the object, number of CPUs and initialization parameters
AUTO_INVALIDATE	BOOLEAN	Lets Oracle decide when to invalidate dependent cursors
AUTO_SAMPLE_SIZE	NUMBER	Indicates that auto-sample size algorithms should be used
PURGE_ALL	TIMESTAMP WITH TIME ZONE	A flag that can be passed to the PURGE_STATS Procedure and unconditionally deletes all the history statistics. The deletion uses TRUNCATE statements on the various dictionary statistics tables holding the history of statistics.
RECLAIM_SYNOPSIS	TIMESTAMP WITH TIME ZONE	A constant used for reclaiming synopsis table space.

DBMS_STATS Operational Notes

Observe these operational notes when using the DBMS_STATS package.

The DBMS_STATS subprograms perform the following general operations:

- Gathering Optimizer Statistics
- Setting or Getting Statistics
- Deleting Statistics
- Transferring Statistics
- Locking or Unlocking Statistics
- Restoring and Purging Statistics History
- User-Defined Statistics
- Pending Statistics
- Comparing Statistics
- Extended Statistics
- Optimizer Statistics Advisor

Most of the DBMS_STATS procedures include the three parameters statown, stattab, and statid. These parameters enable you to store statistics in your own tables (outside of the dictionary), which does not affect the optimizer. Therefore, you can maintain and experiment with *sets* of statistics.

The `stattab` parameter specifies the name of a table in which to hold statistics, and it is assumed that it resides in the same schema as the object for which statistics are collected (unless the `statown` parameter is specified). You can create multiple tables with different `stattab` identifiers to hold separate sets of statistics.

Additionally, you can maintain different sets of statistics within a single `stattab` by using the `statid` parameter, which avoids cluttering the user's schema.

For the `SET` and `GET` procedures, if `stattab` is not provided (that is, `NULL`), then the operation works directly on the dictionary statistics; therefore, you do not need to create these statistics tables if they only plan to modify the dictionary directly. However, if `stattab` is not `NULL`, then the `SET` or `GET` operation works on the specified user statistics table, and not the dictionary.

You can change the default values of some of the parameters of `DBMS_STATS` procedures using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Most procedures in this package commit the current transaction, perform the operation, and then commit again.

Most of the procedures have a `force` parameter that enables you to override a lock on statistics. Whenever statistics in dictionary are modified, old versions of statistics are saved automatically for future restoring.

Gathering Optimizer Statistics

Use the following subprograms to gather certain classes of optimizer statistics, with possible performance improvements over the `ANALYZE` command:

GATHER_DATABASE_STATS Procedures
GATHER_DICTIONARY_STATS Procedure
GATHER_FIXED_OBJECTS_STATS Procedure
GATHER_INDEX_STATS Procedure
GATHER_SCHEMA_STATS Procedures
GATHER_SYSTEM_STATS Procedure
GATHER_TABLE_STATS Procedure

The `GATHER_*` procedures also collect user-defined statistics for columns and domain indexes.

The `statown`, `stattab`, and `statid` parameters instruct the package to back up current statistics in the specified table before gathering new statistics.

Oracle also provides the following procedure for generating statistics for derived objects when you have sufficient statistics on related objects:

GENERATE_STATS Procedure

Setting or Getting Statistics

Use the following subprograms to store and retrieve individual column-related, index-related, and table-related statistics:

PREPARE_COLUMN_VALUES Procedures
PREPARE_COLUMN_VALUES_NVARCHAR Procedure
PREPARE_COLUMN_VALUES_ROWID Procedure

SEED_COL_USAGE Procedure
SET_INDEX_STATS Procedures
SET_SYSTEM_STATS Procedure

SET_TABLE_STATS Procedure

GET_COLUMN_STATS Procedures
GET_INDEX_STATS Procedures
GET_SYSTEM_STATS Procedure
GET_TABLE_STATS Procedure

In the special versions of the `SET_*_STATS` procedures for setting user-defined statistics, the following, if provided, are stored in the dictionary or user statistics table:

- User-defined statistics
- Owner of statistics type
- Name of statistics type

The user-defined statistics and the corresponding statistics type are inserted into the `USTATS$` dictionary table. You can specify user-defined statistics without specifying the statistics type name.

The special versions of the `GET_*_STATS` procedures return user-defined statistics and the statistics type owner and name as `OUT` arguments corresponding to the schema object specified. If user-defined statistics are not collected, `NULL` values are returned.

Deleting Statistics

The `DELETE_*` procedures delete both user-defined statistics and the standard statistics for the given schema object.

DELETE_COLUMN_STATS Procedure
DELETE_DATABASE_STATS Procedure
DELETE_DICTIONARY_STATS Procedure
DELETE_FIXED_OBJECTS_STATS Procedure
DELETE_INDEX_STATS Procedure
DELETE_SCHEMA_STATS Procedure
DELETE_SYSTEM_STATS Procedure
DELETE_TABLE_STATS Procedure

Note that `DELETE_TABLE_STATS`, `DELETE_DICTIONARY_STATS`, `DELETE_DATABASE_STATS` and `DELETE_SCHEMA_STATS` have a parameter `stat_category` which specifies which statistics to delete. The parameter accepts multiple values separated by comma. The supported values are `'OBJECT_STATS'` (table statistics, column statistics and index statistics) and `'SYNOPSES'` (auxiliary statistics created when statistics are incrementally maintained). The default is `'OBJECT_STATS, SYNOPSES'`.

Transferring Statistics

Use the following procedures for creating and dropping the user statistics table.

CREATE_STAT_TABLE Procedure
DROP_STAT_TABLE Procedure

Use the following procedures to transfer statistics

- from the dictionary to a user statistics table (`EXPORT_*`)
- from a user statistics table to the dictionary (`IMPORT_*`)

EXPORT_COLUMN_STATS Procedure
EXPORT_DATABASE_STATS Procedure
EXPORT_DICTIONARY_STATS Procedure

EXPORT_FIXED_OBJECTS_STATS Procedure
EXPORT_INDEX_STATS Procedure
EXPORT_SCHEMA_STATS Procedure
EXPORT_SYSTEM_STATS Procedure
EXPORT_TABLE_STATS Procedure

IMPORT_COLUMN_STATS Procedure
IMPORT_DATABASE_STATS Procedure
IMPORT_DICTIONARY_STATS Procedure
IMPORT_FIXED_OBJECTS_STATS Procedure
IMPORT_INDEX_STATS Procedure
IMPORT_SCHEMA_STATS Procedure
IMPORT_SYSTEM_STATS Procedure
IMPORT_TABLE_STATS Procedure

> **Note:**
>
> Oracle does not support export or import of statistics across databases of different character sets.

Locking or Unlocking Statistics

Use the following procedures to lock and unlock statistics on objects.

LOCK_PARTITION_STATS Procedure
LOCK_SCHEMA_STATS Procedure
LOCK_TABLE_STATS Procedure

UNLOCK_PARTITION_STATS Procedure
UNLOCK_SCHEMA_STATS Procedure
UNLOCK_TABLE_STATS Procedure

The `LOCK_*` procedures either freeze the current set of the statistics or to keep the statistics untouched.When statistics on a table are locked, all the statistics depending on the table, including table statistics, column statistics, histograms and statistics on all dependent indexes, are considered to be locked.

Restoring and Purging Statistics History

Use the following procedures to restore statistics as of a specified timestamp. This is useful in case newly collected statistics leads to some sub-optimal execution plans and the administrator wants to revert to the previous set of statistics.

RESTORE_DATABASE_STATS Procedure
RESTORE_DICTIONARY_STATS Procedure
RESTORE_FIXED_OBJECTS_STATS Procedure
RESTORE_SCHEMA_STATS Procedure
RESTORE_SYSTEM_STATS Procedure
RESTORE_TABLE_STATS Procedure

Whenever statistics in dictionary are modified, old versions of statistics are saved automatically for future restoring. The old statistics are purged automatically at regular intervals based on the statistics history retention setting and the time of recent statistics gathering performed in the

system. Retention is configurable using the ALTER_STATS_HISTORY_RETENTION Procedure.

The other DBMS_STATS procedures related to restoring statistics are:

- PURGE_STATS Procedure: This procedure lets you manually purge old versions beyond a time stamp.
- GET_STATS_HISTORY_RETENTION Function: This function gets the current statistics history retention value.
- GET_STATS_HISTORY_AVAILABILITY Function: This function gets the oldest time stamp where statistics history is available. Users cannot restore statistics to a time stamp older than the oldest time stamp.

RESTORE_* operations are not supported for user defined statistics.

User-Defined Statistics

The DBMS_STATS package supports operations on user-defined statistics. When a domain index or column is associated with a statistics type (using the associate statement), operations on the index or column manipulate user-defined statistics. For example, gathering statistics for a domain index (for which an association with a statistics type exists) using the GET_INDEX_STATS Procedures invokes the user-defined statistics collection method of the associated statistics type. Similarly, delete, transfer, import, and export operations manipulate user-defined statistics.

SET_* and GET_* operations for user-defined statistics are also supported using a special version of the SET and GET interfaces for columns and indexes.

EXPORT_*, IMPORT_* and RESTORE_* operations are not supported for user defined statistics.

Pending Statistics

The package gathers statistics and stores it in the dictionary by default. User's can store these statistics in the system's private area instead of the dictionary by turning the PUBLISH option to FALSE using the SET*PREFS procedures. The default value for PUBLISH is TRUE.The statistics stored in private area are not used by Cost Based Optimizer unless parameter optimizer_use_pending_statistics is set to TRUE. The default value of this parameter is FALSE and this boolean parameter can be set at the session/system level. Users can verify the impact of the new statistics on query plans by using the pending statistics on a session.

Pending statistics provide a mechanism to verify the impact of the new statistics on query plans before making them available for general use. There are two scenarios to verify the query plans:

- Export the pending statistics (use the EXPORT_PENDING_STATS Procedure) to a test system, then run the query workload and check the performance or plans.
- Set optimizer_use_pending_statistics to TRUE in a session on the system where pending statistics have been gathered, run the workload, and check the performance or plans.

After the performance or query plans have been verified, you can publish the pending statistics using the PUBLISH_PENDING_STATS Procedure if the performance is acceptable, or delete the pending statistics using DELETE_PENDING_STATS Procedure if it is not.

Pending statistics can be published, exported, or deleted. The following procedures are provided to manage pending statistics:

- DELETE_PENDING_STATS Procedure

- EXPORT_PENDING_STATS Procedure
- PUBLISH_PENDING_STATS Procedure

Comparing Statistics

You can use the `DIFF_TABLE_STATS_*` statistics to compare statistics for a table from two different sources. The statistics can be from:

- Two different user statistics tables
- A single user statistics table containing two sets of statistics that can be identified using `statids`
- A user statistics table and dictionary history
- Pending statistics

The functions also compare the statistics of the dependent objects (indexes, columns, partitions). They display statistics of the objects from both sources if the difference between those statistics exceeds a certain threshold. The threshold can be specified as an argument to the function, with a default of 10%. The database uses the statistics corresponding to the first source (`stattab1` or `time1`) as a basis for computing the differential percentage.

Extended Statistics

This package enables you to collect statistics for column groups and expressions. The statistics collected for column groups and expressions are called "extended statistics".

Statistics on column groups are used by optimizer for accounting correlation between columns. For example, if a query has predicates c1=1 and c2=1 and if there are statistics on column group (c1, c2), the optimizer uses these statistics for estimating the combined cardinality of the predicates. The optimizer uses the expression statistics to estimate cardinality of predicates on those expressions. The extended statistics are similar to column statistics. The procedures that take columns names accept extended statistics names in place of column names.

Related subprograms:

- CREATE_EXTENDED_STATS Function
- DROP_EXTENDED_STATS Procedure
- SHOW_EXTENDED_STATS_NAME Function
- REPORT_COL_USAGE Function
- SEED_COL_USAGE Procedure
- RESET_COL_USAGE Procedure

Optimizer Statistics Advisor

Optimizer Statistics Advisor is built-in diagnostic software that helps use to use best practices to manage optimizer statistics. The advisor analyzes how you are currently gathering statistics (using manual scripts, explicitly setting parameters, and so on), the effectiveness of existing statistics gathering jobs, and the quality of the gathered statistics. The advisor generates findings for any issues it finds. Based on these findings, the advisor provides recommendations, which it stores in `DBA_ADVISOR_RECOMMENDATIONS`.

The advisor organizes rules into the following classes:

- System

This class checks the preferences for statistics collection, status of the automated statistics gathering job, use of SQL plan directives, and so on. Rules in this class have the value `SYSTEM` in `V$STATS_ADVISOR_RULES.RULE_TYPE`.

- Operation

  This class checks whether statistics collection uses the defaults, test statistics are created using the `SET_*_STATS` procedures, and so on. Rules in this class have the value `OPERATION` in `V$STATS_ADVISOR_RULES.RULE_TYPE`.

- Object

  This class checks for the quality of the statistics, staleness of statistics, unnecessary collection of statistics, and so on. Rules in this class have the value `OBJECT` in `V$STATS_ADVISOR_RULES.RULE_TYPE`.

All Optimizer Statistics Advisor subprograms require the `ADVISOR` privilege. All procedures and functions execute using the invoker's privilege for the operation instead of the task owner's privilege. For example, if a user without the `ANALYZE ANY DICTIONARY` privilege creates a task `t1`, and if a DBA then executes this task, then the task execution checks for `SYS` objects. Another example is a task that is executed by `user1`, interrupted, and then resumed by `user2`. In this case, the checks of the resumed execution are based on the privilege of `user2` rather than `user1`.

You can use the following subprograms to manage Optimizer Statistics Advisor:

- CANCEL_ADVISOR_TASK Procedure
- CONFIGURE_ADVISOR_FILTER Function
- CONFIGURE_ADVISOR_OBJ_FILTER Function
- CONFIGURE_ADVISOR_OPR_FILTER Functions
- CONFIGURE_ADVISOR_RULE_FILTER Function
- CREATE_ADVISOR_TASK Function
- DROP_ADVISOR_TASK Procedure
- EXECUTE_ADVISOR_TASK Function
- GET_ADVISOR_OPR_FILTER Procedure
- GET_ADVISOR_RECS Function
- IMPLEMENT_ADVISOR_TASK Function
- INTERRUPT_ADVISOR_TASK Procedure
- REPORT_ADVISOR_TASK Function
- RESET_ADVISOR_TASK Procedure
- RESUME_ADVISOR_TASK Procedure
- SCRIPT_ADVISOR_TASK Function
- SET_ADVISOR_TASK_PARAMETER Procedure

> **See Also:**
>
> *Oracle Database SQL Tuning Guide*to learn how to analyze statistics using Optimizer Statistics Advisor

DBMS_STATS Data Structures

The `DBMS_STATS` package defines a `RECORD` type.

RECORD Types

- STAT_REC Record Type

DBMS_STATS STAT_REC Record Type

This record type is provided for users in case they want to set column statistics manually. Its fields allow specifying column min/max values, as well as a histogram for a column.

Syntax

```
TYPE STATREC IS RECORD (
  epc    NUMBER,
  minval RAW(2000),
  maxval RAW(2000),
  bkvals NUMARRAY,
  novals NUMARRAY,
  chvals CHARARRAY,
  eavals RAWARRAY,
  rpcnts NUMARRAY,
  eavs   NUMBER);
```

Fields of the Record type COMPARISON_TYPE (STAT_REC Attributes)

Table 197-2 STAT_REC Attributes

Field	Description
`epc`	Number of buckets in histogram
`minval`	Minimum value
`maxval`	Maximum value
`bkvals`	Array of bucket numbers
`novals`	Array of normalized end point values
`chvals`	Array of dumped end point values
`eavals`	Array of end point actual values
`rpcnts`	Array of end point value frequencies
`eavs`	A number indicating whether actual end point values are needed in the histogram. If using the PREPARE_COLUMN_VALUES Procedures, this field will be automatically filled.

Summary of DBMS_STATS Subprograms

This table lists the `DBMS_STATS` subprograms and briefly describes them.

Table 197-3 ***DBMS_STATS Package Subprograms***

Subprogram	Description
ALTER_STATS_HISTORY_RETENTION Procedure	Changes the statistics history retention value
CANCEL_ADVISOR_TASK Procedure	Cancels an Optimizer Statistics Advisor execution
CONFIGURE_ADVISOR_FILTER Function	Configures the filter list for an Optimizer Statistics Advisor task
CONFIGURE_ADVISOR_OBJ_FILTER Function	Configures an object filter for an Optimizer Statistics Advisor task
CONFIGURE_ADVISOR_OPR_FILTER Functions	Configures an operation filter for an Optimizer Statistics Advisor task
CONFIGURE_ADVISOR_RULE_FILTER Function	Configures a rule filter for an Optimizer Statistics Advisor task
CREATE_ADVISOR_TASK Function	Creates an advisor task for the Optimizer Statistics Advisor
CONVERT_RAW_VALUE Procedures	Converts the internal representation of a minimum value, maximum value, or histogram endpoint actual value into a datatype-specific value
CONVERT_RAW_VALUE_NVARCHAR Procedure	Converts the internal representation of a minimum value, maximum value, or histogram endpoint actual value into a datatype-specific value
CONVERT_RAW_VALUE_ROWID Procedure	Converts the internal representation of a minimum value, maximum value, or histogram endpoint actual value into a datatype-specific value
COPY_TABLE_STATS Procedure	Copies the statistics of the source [sub] partition to the destination [sub] partition after scaling
CREATE_EXTENDED_STATS Function	Creates a virtual column for a user specified column group or an expression in a table
CREATE_STAT_TABLE Procedure	Creates a table with name `stattab` in `ownname`'s schema which is capable of holding statistics
DELETE_COLUMN_STATS Procedure	Deletes column-related statistics
DELETE_DATABASE_PREFS Procedure	Deletes the statistics preferences of all the tables
DELETE_DATABASE_STATS Procedure	Deletes statistics for the entire database
DELETE_DICTIONARY_STATS Procedure	Deletes statistics for all dictionary schemas ('`SYS`', '`SYSTEM`' and database component schemas)
DELETE_FIXED_OBJECTS_STATS Procedure	Deletes statistics of all fixed tables
DELETE_INDEX_STATS Procedure	Deletes index-related statistics
DELETE_PENDING_STATS Procedure	Deletes the private statistics that have been collected but have not been published
DELETE_PLSQL_PREFS Procedure	Deletes the preferences for a given function. The values then default to the global preferences.
DELETE_PROCESSING_RATE Procedure	Deletes the processing rate of a given statistics source. If the source is not specified, it deletes the statistics of all the sources
DELETE_SCHEMA_PREFS Procedure	Deletes the statistics preferences of all the tables owned by the specified owner name

Table 197-3 (Cont.) *DBMS_STATS Package Subprograms*

Subprogram	Description
DELETE_SCHEMA_STATS Procedure	Deletes schema-related statistics
DELETE_SYSTEM_STATS Procedure	Deletes system statistics
DELETE_TABLE_PREFS Procedure	Deletes statistics preferences of the specified table in the specified schema
DELETE_TABLE_STATS Procedure	Deletes table-related statistics
DIFF_TABLE_STATS_IN_HISTORY Function	Compares statistics for a table from two timestamps in past and compare the statistics as of that timestamps
DIFF_TABLE_STATS_IN_PENDING Function	Compares pending statistics and statistics as of a timestamp or statistics from dictionary
DIFF_TABLE_STATS_IN_STATTAB Function	Compares statistics for a table from two different sources
DROP_ADVISOR_TASK Procedure	Drops the specified Optimizer Statistics Advisor task
DROP_EXTENDED_STATS Procedure	Drops the statistics entry that is created for the user specified extension
DROP_STAT_TABLE Procedure	Drops a user statistics table created by `CREATE_STAT_TABLE`
EXECUTE_ADVISOR_TASK Function	Executes a previously created Optimizer Statistics Advisor task
EXPORT_COLUMN_STATS Procedure	Retrieves statistics for a particular column and stores them in the user statistics table identified by `stattab`
EXPORT_DATABASE_PREFS Procedure	Exports the statistics preferences of all the tables
EXPORT_DATABASE_STATS Procedure	Retrieves statistics for all objects in the database and stores them in the user statistics table identified by `statown.stattab`
EXPORT_DICTIONARY_STATS Procedure	Retrieves statistics for all dictionary schemas ('`SYS`', '`SYSTEM`' and RDBMS component schemas) and stores them in the user statistics table identified by `stattab`
EXPORT_FIXED_OBJECTS_STATS Procedure	Retrieves statistics for fixed tables and stores them in the user statistics table identified by `stattab`
EXPORT_INDEX_STATS Procedure	Retrieves statistics for a particular index and stores them in the user statistics table identified by `stattab`
EXPORT_PENDING_STATS Procedure	Exports the statistics gathered and stored as pending
EXPORT_SCHEMA_PREFS Procedure	Exports the statistics preferences of all the tables owned by the specified owner name
EXPORT_SCHEMA_STATS Procedure	Retrieves statistics for all objects in the schema identified by `ownname` and stores them in the user statistics table identified by `stattab`
EXPORT_SYSTEM_STATS Procedure	Retrieves system statistics and stores them in the user statistics table
EXPORT_TABLE_PREFS Procedure	Exports statistics preferences of the specified table in the specified schema into the specified statistics table

Table 197-3 (Cont.) *DBMS_STATS Package Subprograms*

Subprogram	Description
EXPORT_TABLE_STATS Procedure	Retrieves statistics for a particular table and stores them in the user statistics table
FLUSH_DATABASE_MONITORING_INFO Procedure	Flushes in-memory monitoring information for all the tables to the dictionary
GATHER_DATABASE_STATS Procedures	Gathers statistics for all objects in the database
GATHER_DICTIONARY_STATS Procedure	Gathers statistics for dictionary schemas 'SYS', 'SYSTEM' and schemas of RDBMS components
GATHER_FIXED_OBJECTS_STATS Procedure	Gathers statistics of fixed objects
GATHER_INDEX_STATS Procedure	Gathers index statistics
GATHER_PROCESSING_RATE Procedure	Starts the job of gathering the processing rates which end after interval defined in minutes
GATHER_SCHEMA_STATS Procedures	Gathers statistics for all objects in a schema
GATHER_SYSTEM_STATS Procedure	Gathers system statistics
GATHER_TABLE_STATS Procedure	Gathers table and column (and index) statistics
GENERATE_STATS Procedure	Generates object statistics from previously collected statistics of related objects
GET_ADVISOR_OPR_FILTER Procedure	Creates an operation filter for an Optimizer Statistics Advisor operation
GET_ADVISOR_RECS Function	Generates a recommendation report for the specified item
GET_COLUMN_STATS Procedures	Gets all column-related information
GET_INDEX_STATS Procedures	Gets all index-related information
GET_PARAM Function	Gets the default value of parameters of DBMS_STATS procedures (see DBMS_STATS Deprecated Subprograms)
GET_PLSQL_PREFS Procedure	Get the preferences for the given function.
GET_PREFS Function	Gets the default value of the specified preference
GET_STATS_HISTORY_AVAILABILITY Function	Gets the oldest timestamp where statistics history is available
GET_STATS_HISTORY_RETENTION Function	Returns the current statistics history retention value
GET_SYSTEM_STATS Procedure	Gets system statistics from stattab, or from the dictionary if stattab is NULL
GET_TABLE_STATS Procedure	Gets all table-related information
IMPLEMENT_ADVISOR_TASK Function	Implements the recommendations made by Optimizer Statistics Advisor

Table 197-3 (Cont.) *DBMS_STATS Package Subprograms*

Subprogram	Description
IMPORT_COLUMN_STATS Procedure	Retrieves statistics for a particular column from the user statistics table identified by `stattab` and stores them in the dictionary
IMPORT_DATABASE_PREFS Procedure	Imports the statistics preferences of all the tables
IMPORT_DATABASE_STATS Procedure	Retrieves statistics for all objects in the database from the user statistics table and stores them in the dictionary
IMPORT_DICTIONARY_STATS Procedure	Retrieves statistics for all dictionary schemas ('`SYS`', '`SYSTEM`' and RDBMS component schemas) from the user statistics table and stores them in the dictionary
IMPORT_FIXED_OBJECTS_STATS Procedure	Retrieves statistics for fixed tables from the user statistics table identified by `stattab` and stores them in the dictionary
IMPORT_INDEX_STATS Procedure	Retrieves statistics for a particular index from the user statistics table identified by `stattab` and stores them in the dictionary
IMPORT_SCHEMA_PREFS Procedure	Imports the statistics preferences of all the tables owned by the specified owner name
IMPORT_SCHEMA_STATS Procedure	Retrieves statistics for all objects in the schema identified by `ownname` from the user statistics table and stores them in the dictionary
IMPORT_SYSTEM_STATS Procedure	Retrieves system statistics from the user statistics table and stores them in the dictionary
IMPORT_TABLE_PREFS Procedure	Sets the statistics preferences of the specified table in the specified schema
IMPORT_TABLE_STATS Procedure	Retrieves statistics for a particular table from the user statistics table identified by `stattab` and stores them in the dictionary
INTERRUPT_ADVISOR_TASK Procedure	Interrupts a currently executing Optimizer Statistics Advisor task.
LOCK_PARTITION_STATS Procedure	Locks statistics for a partition
LOCK_SCHEMA_STATS Procedure	Locks the statistics of all tables of a schema
LOCK_TABLE_STATS Procedure	Locks the statistics on the table
MERGE_COL_USAGE Procedure	Merges column usage information from a source database, by means of a `dblink`, into the local database
PREPARE_COLUMN_VALUES Procedures	Converts user-specified minimum, maximum, and histogram endpoint datatype-specific values into Oracle's internal representation for future storage using the SEED_COL_USAGE Procedure
PREPARE_COLUMN_VALUES_NVARCHAR Procedure	Converts user-specified minimum, maximum, and histogram endpoint datatype-specific values into Oracle's internal representation for future storage using the SEED_COL_USAGE Procedure
PREPARE_COLUMN_VALUES_ROWID Procedure	Converts user-specified minimum, maximum, and histogram endpoint datatype-specific values into Oracle's internal representation for future storage using the SEED_COL_USAGE Procedure
PUBLISH_PENDING_STATS Procedure	Publishes the statistics gathered and stored as pending
PURGE_STATS Procedure	Purges old versions of statistics saved in the dictionary
REMAP_STAT_TABLE Procedure	Remaps the names of objects in the user statistics table

Table 197-3 (Cont.) *DBMS_STATS Package Subprograms*

Subprogram	Description
REPORT_ADVISOR_TASK Function	Reports the results of an Optimizer Advisor Task.
REPORT_COL_USAGE Function	Reports the recorded column (group) usage information
REPORT_GATHER_AUTO_STATS Function	Runs the auto statistics gathering job in reporting mode
REPORT_GATHER_DATABASE_STATS Functions	Runs the GATHER_DATABASE_STATS Procedures in reporting mode.
REPORT_GATHER_DICTIONARY_STATS Functions	Runs the GATHER_DICTIONARY_STATS Procedure in reporting mode
REPORT_GATHER_FIXED_OBJ_STATS Function	Runs the GATHER_FIXED_OBJECTS_STATS Procedure in reporting mode
REPORT_GATHER_SCHEMA_STATS Functions	Runs the GATHER_SCHEMA_STATS Procedures in reporting mode
REPORT_GATHER_TABLE_STATS Function	Runs the GATHER_TABLE_STATS Procedure in reporting mode
REPORT_STATS_OPERATIONS Function	Generates a report of all statistics operations that take place between two timestamps which may or may not have been provided
RESET_ADVISOR_TASK Procedure	Resets an Optimizer Statistics Advisor task execution to its initial state. Only reset a task that is not currently executing
RESET_COL_USAGE Procedure	Resets the recorded column (group) usage information
RESET_GLOBAL_PLSQL_PREFS_DEF Procedure	Resets PL/SQL global preferences to their default values.
RESET_PARAM_DEFAULTS Procedure	Resets global preferences to default values (see DBMS_STATS Deprecated Subprograms)
RESTORE_DICTIONARY_STATS Procedure	Restores statistics of all dictionary tables (tables of 'SYS', 'SYSTEM' and RDBMS component schemas) as of a specified timestamp
RESTORE_FIXED_OBJECTS_STATS Procedure	Restores statistics of all fixed tables as of a specified timestamp
RESTORE_SCHEMA_STATS Procedure	Restores statistics of all tables of a schema as of a specified timestamp
RESTORE_SYSTEM_STATS Procedure	Restores statistics of all tables of a schema as of a specified timestamp
RESTORE_TABLE_STATS Procedure	Restores statistics of a table as of a specified timestamp (as_of_timestamp), as well as statistics of associated indexes and columns
RESUME_ADVISOR_TASK Procedure	Resumes an interrupted task. It only resumes the execution that was most recently interrupted
SCRIPT_ADVISOR_TASK Function	Retrieves the script that implements the recommended actions for the problems found by Optimizer Statistics Advisor
SEED_COL_USAGE Procedure	Iterates over the SQL statements in the specified SQL tuning set, compiles them and seeds column usage information for the columns that appear in these statements
SET_ADVISOR_TASK_PARAMETER Procedure	Updates the value of an Optimizer Statistics Advisor task parameter

Table 197-3 (Cont.) *DBMS_STATS Package Subprograms*

Subprogram	Description
SET_COLUMN_STATS Procedures	Sets column-related information
SET_DATABASE_PREFS Procedure	Sets the statistics preferences of all the tables
SET_GLOBAL_PREFS Procedure	Sets the global statistics preferences
SET_INDEX_STATS Procedures	Sets index-related information
SET_PARAM Procedure	Sets default values for parameters of `DBMS_STATS` procedures (see DBMS_STATS Deprecated Subprograms)
SET_PLSQL_PREFS Procedure	Sets preferences for a specific function.
SET_GLOBAL_PLSQL_PREFS Procedure	Sets the global preference for PL/SQL functions.
SET_PROCESSING_RATE Procedure	Sets the value of rate of processing for a given operation
SET_SCHEMA_PREFS Procedure	Sets the statistics preferences of all the tables owned by the specified owner name
SET_SYSTEM_STATS Procedure	Sets system statistics
SET_TABLE_PREFS Procedure	Sets the statistics preferences of the specified table in the specified schema
SET_TABLE_STATS Procedure	Sets table-related information
SHOW_EXTENDED_STATS_NAME Function	Returns the name of the virtual column that is created for the user-specified extension
TRANSFER_STATS Procedure	Transfers statistics for specified table(s) from a remote database specified by `dblink` to the local database
UNLOCK_PARTITION_STATS Procedure	Unlocks the statistics for a partition
UNLOCK_SCHEMA_STATS Procedure	Unlocks the statistics on all the tables in schema
UNLOCK_TABLE_STATS Procedure	Unlocks the statistics on the table
UPGRADE_STAT_TABLE Procedure	Upgrades user statistics on an older table

ALTER_STATS_HISTORY_RETENTION Procedure

This procedure changes the statistics history retention value.

Statistics history retention is used by both the automatic purge and PURGE_STATS Procedure.

Syntax

```
DBMS_STATS.ALTER_STATS_HISTORY_RETENTION (
   retention       IN     NUMBER);
```

Parameters

Table 197-4 ALTER_STATS_HISTORY_RETENTION Procedure Parameters

Parameter	Description
retention	The retention time in days. The statistics history will be retained for at least these many number of days.The valid range is [1,365000]. Also you can use the following values for special purposes: • -1: Statistics history is never purged by automatic purge • 0: Old statistics are never saved. The automatic purge will delete all statistics history • NULL: Change statistics history retention to default value

Usage Notes

To run this procedure, you must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege.

Exceptions

ORA-20000: Insufficient privileges

CANCEL_ADVISOR_TASK Procedure

This procedure cancels an Optimizer Statistics Advisor execution. The advisor removes all intermediate results of the current execution from the task.

Syntax

```
DBMS_STATS.CANCEL_ADVISOR_TASK (
  task_name IN VARCHAR2);
```

Parameters

Table 197-5 CANCEL_ADVISOR_TASK Procedure Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Consider a case in which a task is executed by one user, interrupted, and then resumed by a different user. In this case, Optimizer Statistics Advisor bases its checks of the resumed execution on the privilege of the user who resumed the task.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Usage Notes

To be canceled or interrupted, the specified task must be currently executing.

Example 197-1 Canceling an Optimizer Statistics Advisor

In this example, you start a SQL*Plus session, and then create and execute an advisor task named my_task:

```
DECLARE
  v_tname   VARCHAR2(128) := 'my_task';
  v_ename   VARCHAR2(128) := NULL;
BEGIN
  -- create a task
  v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);

  -- execute the task
  v_ename := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
END;
/
```

In a separate terminal, you start a second SQL*Plus session, and then execute the following program:

```
EXEC DBMS_STATS.CANCEL_ADVISOR_TASK('my_task');
```

The first session returns an ORA-13632 to indicate the cancelation of the task:

```
ORA-13632: The user cancelled the current operation.
```

CONFIGURE_ADVISOR_FILTER Function

This function configures the filter list for an Optimizer Statistics Advisor task. Filters are useful for excluding irrelevant findings from a report.

Syntax

```
DBMS_STATS.CONFIGURE_ADVISOR_FILTER (
  task_name           IN   VARCHAR2,
  stats_adv_opr_type  IN   VARCHAR2,
  configuration_type  IN   VARCHAR2,
  filter              IN   StatsAdvFilterTab   := NULL)
RETURN CLOB;
```

Parameters

Table 197-6 CONFIGURE_ADVISOR_FILTER Function Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.
stats_adv_opr_type	The type of operation to configure. Possible values are EXECUTE, REPORT, SCRIPT, and IMPLEMENT. The function permits you to specify a combination of operation types by using the plus (+) operator, for example, EXECUTE +REPORT. If this parameter is null, then the filter applies to all types of advisor operations.
configuration_type	The type of configuration. Possible values are as follows: • SET: Sets the specified filter list values. The submitted filter overrides existing filter values. • CLEAR: Clears the existing values for the specified filter. • SHOW: Shows the current values of the specified filter.
filter	The list of filter items for the script.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Return Values

This function returns a CLOB that contains the configuration of the provided filter in XML format.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Usage Notes

To provide fine-grained control and a unified interface across all procedures, DBMS_STATS provides the StatsAdvFilter type. You can use this data type to instantiate and construct a table of filters. You can then pass a parameter of type StatsAdvFilter to CONFIGURE_ADVISOR_FILTER along with a Boolean variable that specifies either of the following:

- Inclusion list

  Only include these objects in the check.
- Exclusion list

  Do not include these objects in the check.

You can also pass in a parameter specifying whether to replace the existing list. This list only filters object-level and operation-level items. The advisor always checks system-level rules.

You can create the following types of filters:

- Rule filter

  This filter takes a rule name as input. Obtain rule names from the `V$STATS_ADVISOR_RULES` view.

- Operation filter

  This filter is an exact match filter that takes in the name of the operation and an XML string representation of all the parameter values in the call. To obtain the XML, see the notes section of the `DBA_OPTSTAT_OPERATIONS` view. To obtain the filter for an operation, use `DBMS_STATS.GET_ADVISOR_OPR_FILTER`.

- Object filter

  This filter accept an owner name and an object name. Wildcards (`%`) are supported in the owner name and object name. When an object name is null or `%`, it means a filter for all the objects in the specified schema. If the owner name is also null or `%`, it means a default filter for all objects in the system.

If none of the filters is specified, then the function recognizes the filter as setting the global default value of filtering (include or exclude). During the check, if no filter has been specified for a rule, operation, or object, then the function uses the default value to determine whether to include or exclude it.

Example 197-2 Enabling and Disabling Rules

You may want to turn off checks for all rules except for a specific rule. In this example, you want to check whether SQL plan directives have been disabled.

```
DECLARE
  v_task_name    VARCHAR2(128)                := 'my_task';
  v_ret          VARCHAR2(128);
  filter         DBMS_STATS.StatsAdvFilter    := null;
  filterTab      DBMS_STATS.StatsAdvFilterTab := null;
  v_counter      NUMBER                       := 0;
  v_filterReport CLOB;
BEGIN
  -- Create the advisor task
  v_ret := DBMS_STATS.CREATE_ADVISOR_TASK(v_task_name);

  -- Initialize the filter table
  filterTab := DBMS_STATS.StatsAdvFilterTab();

  -- First filter: set filters to be FALSE by default
  filter.include := FALSE;

  -- Add this filter to the filter table
  v_counter := v_counter + 1;
  filterTab.extend;
  filterTab(v_counter) := filter;

  -- Second filter: turn on filter for one rule
  filter.include := TRUE;
  filter.rulename := 'TurnOnSQLPlanDirective';

  -- Add the SQL plan directive filter to the filter table
  v_counter := v_counter + 1;
  filterTab.extend;
```

```
  filterTab(v_counter) := filter;

  v_filterReport := DBMS_STATS.CONFIGURE_ADVISOR_FILTER(
                      task_name          => v_task_name,
                      stats_adv_opr_type => NULL,
                      configuration_type => 'SET',
                      filter             => filterTab);
  -- Drop the task
  DBMS_STATS.DROP_ADVISOR_TASK(v_task_name);
END;
```

Example 197-3 Configuring an Operations Filter

In this example, your shop uses customized scripts to gather statistics for a table. If you do not want to see a specific statistics operation in the report, then you can specify an operations filter.

```
DECLARE
  v_task_name    VARCHAR2(128)                := 'my_task';
  v_ret          VARCHAR2(128);
  filter         DBMS_STATS.StatsAdvFilter    := null;
  filterTab      DBMS_STATS.StatsAdvFilterTab := null;
  opr            DBMS_STATS.StatsAdvOpr;
  oprTab         DBMS_STATS.StatsAdvOprTab;
  v_oprCnt       NUMBER                       := 0;
  TYPE numTab IS TABLE OF NUMBER;
  opr_tab        numTab;
  v_filterReport CLOB;

BEGIN
  -- Create the advisor task
  v_ret := DBMS_STATS.CREATE_ADVISOR_TASK(v_task_name);

  -- Initialize filter table
  filterTab := DBMS_STATS.StatsAdvFilterTab();

  -- Initialize operations filter
  oprTab := DBMS_STATS.StatsAdvOprTab();

  SELECT ID
    BULK COLLECT INTO opr_tab
  FROM   WRI$_OPTSTAT_OPR
  WHERE  OPERATION = 'set_table_stats'
  AND    TARGET = 'HR.EMPLOYEES';

  -- Populate the operations table
  FOR i IN 1..opr_tab.count LOOP

    -- Use the procedure GET_ADVISOR_OPR_FILTER to construct
    -- an operation filter
    DBMS_STATS.GET_ADVISOR_OPR_FILTER(opr_tab(i), opr);

    v_oprCnt := v_oprCnt + 1;
    oprTab.extend;
    oprTab(v_oprCnt) := opr;
```

```
  END LOOP;

  filter.include := FALSE;
  filter.oprlist := oprTab;

  -- Add to filter table
  filterTab.extend;
  filterTab(1) := filter;

  v_filterReport := DBMS_STATS.CONFIGURE_ADVISOR_FILTER(
                      task_name          => v_task_name,
                      stats_adv_opr_type => NULL,
                      configuration_type => 'SET',
                      filter             => filterTab);

  -- Drop the task
  DBMS_STATS.DROP_ADVISOR_TASK(v_task_name);

END;
```

Example 197-4 Reporting on a Specific Schema

In this example, you want to generate a report only for the `sh` schema. Also, you want to skip the `sh.products` table. You create an object filter as follows:

```
DECLARE
  v_task_name     VARCHAR2(128)                := 'my_task';
  v_ret           VARCHAR2(128);
  filter          DBMS_STATS.StatsAdvFilter    := null;
  filterTab       DBMS_STATS.StatsAdvFilterTab := null;
  v_filterReport  CLOB;
  v_counter       NUMBER                       := 0;
  obj             DBMS_STATS.ObjectElem;
  objTab          DBMS_STATS.ObjectTab;
  v_objCnt        NUMBER                       := 0;
BEGIN
  -- Create the advisor task
  v_ret := DBMS_STATS.CREATE_ADVISOR_TASK(v_task_name);

  -- Initialize filter table
  filterTab := DBMS_STATS.StatsAdvFilterTab();

  -- Set object filter to be off by default
  filter.include := FALSE;

  objTab := DBMS_STATS.ObjectTab();

  obj.ownname := NULL;
  obj.objname := NULL;

  -- Add to the object table

  v_objCnt := v_objCnt + 1;
  objTab.extend;
  objTab(v_objCnt) := obj;
```

```
filter.objlist := objTab;

-- Add the object filter to the filter table
v_counter := v_counter + 1;
filterTab.extend;
filterTab(v_counter) := filter;

-- In filter 1, turn on the check only for schema SH

filter.include := TRUE;

objTab := DBMS_STATS.ObjectTab();
v_objCnt := 0;

obj.ownname := 'SH';
obj.objname := NULL;

-- add to the object table
v_objCnt := v_objCnt + 1;
objTab.extend;
objTab(v_objCnt) := obj;

filter.objlist := objTab;

-- Add the object filter to the filter table
v_counter := v_counter + 1;
filterTab.extend;
filterTab(v_counter) := filter;

-- In filter 2, exclude the check for object sh.products

filter.include := FALSE;


objTab := dbms_stats.ObjectTab();
v_objCnt := 0;

-- Specify another object filter for sh.products
obj.ownname := 'SH';
obj.objname := 'PRODUCTS';

-- Add to the object table
v_objCnt := v_objCnt + 1;
objTab.extend;
objTab(v_objCnt) := obj;

filter.objlist := objTab;


-- Add the object filter to the filter table
v_counter := v_counter + 1;
filterTab.extend;
filterTab(v_counter) := filter;

v_filterReport :=
DBMS_STATS.CONFIGURE_ADVISOR_FILTER(
```

```
    task_name          => v_task_name,
    stats_adv_opr_type => NULL,
    configuration_type => 'SET',
    filter             => filterTab);

  -- Drop the task
  DBMS_STATS.DROP_ADVISOR_TASK(v_task_name);
END;
```

CONFIGURE_ADVISOR_OBJ_FILTER Function

This function configures an object filter for an Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.CONFIGURE_ADVISOR_OBJ_FILTER (
  task_name             IN VARCHAR2,
  stats_adv_opr_type    IN VARCHAR2,
  rule_name             IN VARCHAR2,
  ownname               IN VARCHAR2,
  tabname               IN VARCHAR2,
  action                IN VARCHAR2)
RETURN CLOB;
```

Parameters

Table 197-7 CONFIGURE_ADVISOR_OBJ_FILTER Function Parameters

Parameter	Description
`task_name`	The name of the Optimizer Statistics Advisor task.
`stats_adv_opr_type`	The type of operation to configure. Possible values are `EXECUTE`, `REPORT`, `SCRIPT`, and `IMPLEMENT`. See"CONFIGURE_ADVISOR_RULE_FILTER Function".
`rule_name`	The name of the rule to configure. If null, the function applies the filter to all operation-level rules.
`ownname`	The owner name of the operation target. If null, the function applies the filter to all owner names.
`tabname`	The table name of the operation target.
`action`	The configuration action to take for the specified rule. See"CONFIGURE_ADVISOR_RULE_FILTER Function".

Security Model

Note the following:

- To execute this subprogram, you must have the `ADVISOR` privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Return Values

This function returns an XML CLOB that contains the updated values of the filter.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

CONFIGURE_ADVISOR_OPR_FILTER Functions

This overloaded function configures an operation filter for an Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.CONFIGURE_ADVISOR_OPR_FILTER (
  task_name             IN VARCHAR2,
  stats_adv_opr_type    IN VARCHAR2,
  rule_name             IN VARCHAR2,
  operation_name        IN VARCHAR2,
  action                IN VARCHAR2)
RETURN CLOB;


DBMS_STATS.CONFIGURE_ADVISOR_OPR_FILTER (
  task_name             IN VARCHAR2,
  stats_adv_opr_type    IN VARCHAR2,
  rule_name             IN VARCHAR2,
  operation_name        IN VARCHAR2,
  ownname               IN VARCHAR2,
  tabname               IN VARCHAR2,
  action                IN VARCHAR2)
RETURN CLOB;


DBMS_STATS.CONFIGURE_ADVISOR_OPR_FILTER (
  task_name             IN VARCHAR2,
  stats_adv_opr_type    IN VARCHAR2,
  rule_name             IN VARCHAR2,
  operation_id          IN NUMBER,
  action                IN VARCHAR2)
RETURN CLOB;
```

Parameters

Table 197-8 CONFIGURE_ADVISOR_OPR_FILTER Function Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.

Table 197-8 (Cont.) CONFIGURE_ADVISOR_OPR_FILTER Function Parameters

Parameter	Description
stats_adv_opr_type	The type of operation to configure. Possible values are EXECUTE, REPORT, SCRIPT, and IMPLEMENT. See"CONFIGURE_ADVISOR_RULE_FILTER Function".
rule_name	The name of the rule to configure. If null, the function applies the filter to all operation-level rules.
operation_name	The name of the operation. For example, an operation name could be gather_table_stats. This value cannot be null.
operation_id	The ID of the operation to configure. The filter applies to any operation with the same signature as the specified operation ID. If two operations have the same signature, then they have the same value for every parameter. View the operation ID in DBA_OPSTAT_OPERATIONS.ID. This value cannot be null.
ownname	The owner name of the operation target. This value cannot be null.
tabname	The table name of the operation target.
action	The configuration action to take for the specified rule. See"CONFIGURE_ADVISOR_RULE_FILTER Function".

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Return Values

This function returns an XML CLOB that contains the updated values of the filter.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Example 197-5 Excluding Operations for Gathering Table Statistics

In this example, your goal is to exclude operations that gather table statistics in the hr schema. User account stats has been granted the DBA role, ADVISOR privilege, and SELECT ON DBA_OPTSTAT_OPERATIONS privilege. You perform the following steps:

1. Log in to the database as stats.
2. Drop any existing task named opt_adv_task1.

```
DECLARE
  v_tname VARCHAR2(32767);
BEGIN
  v_tname := 'opt_adv_task1';
  DBMS_STATS.DROP_ADVISOR_TASK(v_tname);
```

```
END;
/
```

3. Create a procedure named `opr_filter` that configures a task to advise on all operations *except* those that gather statistics for tables in the `hr` schema.

```
CREATE OR REPLACE PROCEDURE opr_filter(p_tname IN VARCHAR2) IS
   v_retc CLOB;
BEGIN
   -- For all rules, prevent the advisor from operating
   -- on the operations selected in the following query
   FOR rec IN
     (SELECT ID FROM DBA_OPTSTAT_OPERATIONS WHERE OPERATION =
'gather_table_stats' AND TARGET LIKE 'HR.%')
   LOOP
     v_retc := DBMS_STATS.CONFIGURE_ADVISOR_OPR_FILTER(
                 task_name          => p_tname
               , stats_adv_opr_type => NULL
               , rule_name          => NULL
               , operation_id       => rec.id
               , action             => 'DISABLE');
   END LOOP;
END;
/
SHOW ERRORS
```

4. Create a task named `opt_adv_task1`, and then execute the `opr_filter` procedure for this task.

```
DECLARE
  v_tname VARCHAR2(32767);
  v_ret VARCHAR2(32767);
BEGIN
  v_tname := 'opt_adv_task1';
  v_ret   := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);
  opr_filter(v_tname);
END;
/
```

5. Execute the task `opt_adv_task1`.

```
DECLARE
  v_tname VARCHAR2(32767);
  v_ret   VARCHAR2(32767);
begin
  v_tname := 'opt_adv_task1';
  v_ret   := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
END;
/
```

6. Print the report.

```
SPOOL /tmp/rep.txt
SET LONG 1000000
COLUMN report FORMAT A200
```

```
SET LINESIZE 250
SET PAGESIZE 1000

SELECT DBMS_STATS.REPORT_ADVISOR_TASK(
         task_name      => 'opt_adv_task1'
       , execution_name => NULL
       , type           => 'TEXT'
       , section        => 'ALL'
       ) AS report
FROM   DUAL;
SPOOL OFF
```

See Also:

- *Oracle Database Reference* to learn more about `DBA_OPTSTAT_OPERATIONS`
- *Oracle Database SQL Tuning Guide* to learn how to manage Optimizer Statistics Advisor

CONFIGURE_ADVISOR_RULE_FILTER Function

This function configures a rule filter for an Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.CONFIGURE_ADVISOR_RULE_FILTER (
  task_name            IN   VARCHAR2,
  stats_adv_opr_Type   IN   VARCHAR2,
  rule_name            IN   VARCHAR2,
  action               IN   VARCHAR2)
RETURN CLOB;
```

Parameters

Table 197-9 SCRIPT_ADVISOR_TASK Function Parameters

Parameter	Description
`task_name`	The name of the Optimizer Statistics Advisor task.
`stats_adv_opr_type`	The type of operation to configure. Possible values are `EXECUTE`, `REPORT`, `SCRIPT`, and `IMPLEMENT`. You can specify a combination of operation types, for example, `EXECUTE +REPORT`. If this parameter is null, then the filter applies to all types of Optimizer Statistics Advisor operations.
`rule_name`	The name of the rule to configure. If null, the function applies the filter to all rules.
`action`	The configuration action to take for the specified rule. Possible values are: • `ENABLE`: Enables the filter • `DISABLE`: Disables the filter • `DELETE`: Deletes the filter • `SHOW`: Shows the current filter value

Security Model

Note the following:

- To execute this subprogram, you must have the `ADVISOR` privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Return Values

This function returns an XML CLOB that contains the updated values of the filter.

Exceptions

- `ORA-20000`: Insufficient privileges
- `ORA-20001`: Invalid input values
- `ORA-20012`: Optimizer Statistics Advisor errors

CONVERT_RAW_VALUE Procedures

This procedure converts the internal representation of a minimum value, maximum value, or histogram endpoint actual value into a datatype-specific value.

The `minval`, `maxval`, and `eavals` fields of the `StatRec` structure as filled in by `GET_COLUMN_STATS` or `PREPARE_COLUMN_VALUES` are appropriate values for input.

Syntax

```
DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT BINARY_FLOAT);

DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT BINARY_DOUBLE);

DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT DATE);

DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT NUMBER);

DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT VARCHAR2);
```

Parameters

Table 197-10 CONVERT_RAW_VALUE Procedure Parameters

Parameter	Description
`rawval`	Raw representation of a column minimum, maximum, histogram end point actual value

Table 197-10 (Cont.) CONVERT_RAW_VALUE Procedure Parameters

Parameter	Description
resval	Converted, type-specific value

Usage Notes

No special privilege or role is needed to invoke this procedure.

CONVERT_RAW_VALUE_NVARCHAR Procedure

This procedure converts the internal representation of a a minimum value, maximum value, or histogram end point actual value.

The minval, maxval and eavals fields of the StatRec structure as filled in by GET_COLUMN_STATS or PREPARE_COLUMN_VALUES are appropriate values for input.

Syntax

```
DBMS_STATS.CONVERT_RAW_VALUE_NVARCHAR (
   rawval     RAW,
   resval OUT NVARCHAR2);
```

Parameters

Table 197-11 CONVERT_RAW_VALUE_NVARCHAR Procedure Parameters

Parameter	Description
rawval	The raw representation of a column minimum or maximum datatype-specific output parameters
resval	The converted, type-specific value

Usage Notes

No special privilege or role is needed to invoke this procedure.

CONVERT_RAW_VALUE_ROWID Procedure

This procedure converts the internal representation of a a minimum value, maximum value, or histogram end point actual value.

The minval, maxval and eavals fields of the StatRec structure as filled in by GET_COLUMN_STATS or PREPARE_COLUMN_VALUES are appropriate values for input.

Syntax

```
DBMS_STATS.CONVERT_RAW_VALUE_ROWID (
   rawval     RAW,
   resval OUT ROWID);
```

Pragmas

```
pragma restrict_references(convert_raw_value_rowid, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 197-12 CONVERT_RAW_VALUE_ROWID Procedure Parameters

Parameter	Description
rawval	The raw representation of a column minimum or maximum datatype-specific output parameters
resval	The converted, type-specific value

Usage Notes

No special privilege or role is needed to invoke this procedure.

COPY_TABLE_STATS Procedure

This procedure copies statistics of all dependent object such as columns and local indexes. If the statistics for source are not available then nothing is copied. It can optionally scale the statistics (such as the number of blks, or number of rows) based on the given scale_factor.

Syntax

```
DBMS_STATS.COPY_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   srcpartname      VARCHAR2,
   dstpartname      VARCHAR2,
   scale_factor     VARCHAR2 DEFAULT 1,
   flags            NUMBER DEFAULT 0,
   force            BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-13 COPY_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	Schema of the table of source and destination [sub] partitions
tabname	Table name of source and destination [sub] partitions
srcpartname	Source [sub] partition
dtspartname	Destination [sub] partition
scale_factor	Scale factor to scale nblks, nrows etc. in dstpartname
flags	For internal Oracle use.
force	When value of this argument is TRUE copy statistics even if the destination [sub]partition is locked

Security Model

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

`ORA-20000`: Invalid [sub]partition name

`ORA-20001`: Bad input value

Usage Notes

This procedure updates the minimum and maximum values of destination partition for the first partitioning column as follows:

- If the partitioning type is `HASH`, then the minimum and maximum values of the destination partition are same as that of the source partition.
- If the partitioning type is `LIST`, then the behavior depends on the setting of the destination partition:
  - If the destination partition is a `NOT DEFAULT` partition, then the following statements are true:
    * The minimum value of the destination partition is set to the minimum value of the value list that describes the destination partition.
    * The maximum value of the destination partition is set to the maximum value of the value list that describes the destination partition.
  - Alternatively, if the destination partition is a `DEFAULT` partition, then the following statements are true:
    * The minimum value of the destination partition is set to the minimum value of the source partition.
    * The maximum value of the destination partition is set to the maximum value of the source partition.
- If the partitioning type is `RANGE`, then the following statements are true:
  - The minimum value of the destination partition is set to the high bound of previous partition unless the destination partition is the first partition. For the first partition, the minimum value is set to the high bound of the destination partition.
  - The maximum value of the destination partition is set to the high bound of the destination partition unless the high bound of the destination partition is `MAXVALUE`, in which case the maximum value of the destination partition is set to the high bound of the previous partition.
  - If the source partition column's minimum value is equal to its maximum value, and if both are equal to the source partition's lower bound, and if it has a single distinct value, then the destination partition column's minimum and maximum values are both set to the destination partition's lower bound. This is done for all partitioning columns.

    If the above condition does not apply, second and subsequent partitioning columns are updated as follows. The destination partition column's maximum value is set to the greater of the destination partition upper bound and the source partition column's maximum value, with one exception. If the destination partition is `D` and its preceding partition is `D-1` and the key column to be adjusted is `Cn`, the maximum value for `Cn` is set to the upper bound of `D` (ignoring the maximum value of the source partition column) provided that the upper bounds of the previous key column `Cn-1` are the same in partitions `D` and `D-1`.

- If the minimum and maximum values are different for a column after modifications, and if the number of distinct values is less than 1, then the number of distinct values is updated as 2.
- If the source or destination is a partition of a composite partitioned table, then this procedure does not copy statistics of the underlying subpartitions.

CREATE_ADVISOR_TASK Function

This function creates an advisor task for the Optimizer Statistics Advisor.

Syntax

```
DBMS_STATS.CREATE_ADVISOR_TASK (
   task_name    IN   VARCHAR2   := NULL)
 RETURN VARCHAR2;
```

Parameters

Table 197-14 CREATE_ADVISOR_TASK Function Parameters

Parameter	Description
`task_name`	Name of the task. If the task name is already specified, then the function uses the specified task name. Otherwise, the function generates a new task name automatically.

Security Model

Note the following:

- To execute this subprogram, you must have the `ADVISOR` privilege.
- This subprogram executes using invoker's rights.

Return Values

This function returns the unique name of the Optimizer Statistics Advisor task.

Exceptions

`ORA-20000`: Insufficient privileges / creating extension is not supported

`ORA-20001`: Error when processing extension

`ORA-20012`: Optimizer Statistics Advisor errors

Example 197-6 Creating and Executing a Task

This example creates an Optimizer Statistics Advisor task named `my_task`, and then executes it.

```
DECLARE
  v_tname   VARCHAR2(128) := 'my_task';
BEGIN
  -- create a task
  v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);
  -- execute the task
```

```
  v_tname := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
END;
```

> **Note:**
>
> *Oracle Database SQL Tuning Guide* to learn how manage Optimizer Statistics Advisor

CREATE_EXTENDED_STATS Function

This function creates a column statistics entry in the system for a user-specified column group or an expression in a table.

The database gathers statistics for this extension when a user-initiated or automatic statistics gathering job gathers statistics for the table. Statistics for such an extension are called **extended statistics**. This function returns the name of this newly created entry for the extension.
This second form of this function creates statistics extensions based on the column group usage recorded by the SEED_COL_USAGE Procedure. This function returns a report of extensions created.

Syntax

```
DBMS_STATS.CREATE_EXTENDED_STATS (
   ownname    VARCHAR2,
   tabname    VARCHAR2,
   extension  VARCHAR2)
 RETURN VARCHAR2;

DBMS_STATS.CREATE_EXTENDED_STATS (
   ownname    VARCHAR2,
   tabname    VARCHAR2)
 RETURN CLOB;
```

Parameters

Table 197-15 CREATE_EXTENDED_STATS Function Parameters

Parameter	Description
`ownname`	Owner name of a table
`tabname`	Name of the table
`extension`	Can be either a column group or an expression. Suppose the specified table has columns `c1`, `c2`. An example column group is "(`c1`, `c2`)". An example expression is "(`c1` + `c2`)".

Return Values

This function returns the name of this newly created entry for the extension.

Exceptions

ORA-20000: Insufficient privileges / creating extension is not supported

ORA-20001: Error when processing extension

ORA-20007: Extension already exists

ORA-20008: Reached the upper limit on number of extensions

Usage Notes

To invoke this procedure you must be owner of the table, or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table, or have either the ANALYZE ANY DICTIONARY or SYSDBA privilege.

The extension has the following restrictions:

- The extension cannot contain a virtual column.
- Extensions cannot be created on tables owned by SYS.
- Extensions cannot be created on cluster tables, index organized tables, temporary tables, or external tables.
- The total number of extensions in a table cannot be greater than a maximum of (20, 10% of number of non-virtual columns in the table).
- The number of columns in a column group must be in the range [2, 32].
- A column cannot appear more than once in a column group.
- The extension can contain an expression only if a corresponding virtual column has been created.
- An expression must contain at least one column.
- An expression cannot contain a subquery.
- The COMPATIBLE parameter must be 11.0.0.0.0 or greater.

CREATE_STAT_TABLE Procedure

This procedure creates a table with name stattab in ownname's schema which is capable of holding statistics. The columns and types that compose this table are not relevant as it should be accessed solely through the procedures in this package.

Syntax

```
DBMS_STATS.CREATE_STAT_TABLE (
   ownname             VARCHAR2,
   stattab             VARCHAR2,
   tblspace            VARCHAR2 DEFAULT NULL,
   global_temporary    BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-16 CREATE_STAT_TABLE Procedure Parameters

Parameter	Description
ownname	Name of the schema
stattab	Name of the table to create. This value should be passed as the stattab parameter to other procedures when the user does not want to modify the dictionary statistics directly.
tblspace	Tablespace in which to create the statistics tables. If none is specified, then they are created in the user's default tablespace.
global_temporary	Whether or not the table should be created as a global temporary table

Security Model

To invoke this procedure you need whichever privileges are required for creating a table in the specified schema.

Exceptions

ORA-20000: Table already exists or insufficient privileges

ORA-20001: Tablespace does not exist

DELETE_COLUMN_STATS Procedure

This procedure deletes column-related statistics.

Syntax

```
DBMS_STATS.DELETE_COLUMN_STATS (
   ownname         VARCHAR2,
   tabname         VARCHAR2,
   colname         VARCHAR2,
   partname        VARCHAR2 DEFAULT NULL,
   stattab         VARCHAR2 DEFAULT NULL,
   statid          VARCHAR2 DEFAULT NULL,
   cascade_parts   BOOLEAN  DEFAULT TRUE,
   statown         VARCHAR2 DEFAULT NULL,
   no_invalidate   BOOLEAN  DEFAULT to_no_invalidate_type (
                                      get_param('NO_INVALIDATE')),
   force           BOOLEAN  DEFAULT FALSE,
   col_stat_type   VARCHAR2 DEFAULT 'ALL');
```

Parameters

Table 197-17 DELETE_COLUMN_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
tabname	Name of the table to which this column belongs
colname	Name of the column or extension

Table 197-17 (Cont.) DELETE_COLUMN_STATS Procedure Parameters

Parameter	Description
partname	Name of the table partition for which to delete the statistics. If the table is partitioned and if partname is NULL, then global column statistics are deleted.
stattab	User statistics table identifier describing from where to delete the statistics. If stattab is NULL, then the statistics are deleted directly from the dictionary.
statid	Identifier (optional) to associate with these statistics within stattab (Only pertinent if stattab is not NULL).
cascade_parts	If the table is partitioned and if partname is NULL, then setting this to true causes the deletion of statistics for this column for all underlying partitions as well.
statown	Schema containing stattab (if different than ownname)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	When value of this argument is TRUE, deletes column statistics even if locked
col_stat_type	Type of column statistics to be deleted. This argument takes the following values: • HISTOGRAM - delete column histogram only • ALL - delete base column statistics and histogram

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20005: Object statistics are locked

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

DELETE_DATABASE_PREFS Procedure

This procedure deletes the statistics preferences set for all non-system tables. You can include system tables by passing `TRUE` for the `add_sys` parameter.

Syntax

```
DBMS_STATS.DELETE_DATABASE_PREFS (
    pname            IN   VARCHAR2,
    add_sys          IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-18 DELETE_DATABASE_PREFS Procedure Parameters

Parameter	Description
`pname`	Preference name. The existing value for following preferences can be deleted and default preference values will be used: • `APPROXIMATE_NDV_ALGORITHM` • `AUTO_STAT_EXTENSIONS` • `CASCADE` • `DEGREE` • `ESTIMATE_PERCENT` • `GLOBAL_TEMP_TABLE_STATS` • `GRANULARITY` • `INCREMENTAL` • `INCREMENTAL_LEVEL` • `INCREMENTAL_STALENESS` • `METHOD_OPT` • `NO_INVALIDATE` • `OPTIONS` • `PREFERENCE_OVERRIDES_PARAMETER` • `PUBLISH` • `STALE_PERCENT` • `STAT_CATEGORY` • `TABLE_CACHED_BLOCKS`
`add_sys`	Determines whether `SYS` tables will be included.

Table 197-19 Statistics Preferences

Preference	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. Consider a synopsis as an internal management structure that samples distinct values. You can specify the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
CASCADE	Determines whether index statistics are collected as part of gathering table statistics.
DEGREE	Determines degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	Determines the percentage of rows to estimate.
GLOBAL_TEMP_TABLE_STATS	Controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics.
GRANULARITY	Determines the granularity of statistics to collect. This value is only relevant for partitioned tables.
INCREMENTAL	Determines whether the global statistics of a partitioned table will be maintained without doing a full table scan.
INCREMENTAL_LEVEL	Controls which synopses to collect when INCREMENTAL preference is set to TRUE.

Table 197-19 (Cont.) Statistics Preferences

Preference	Description

INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'.
	
	The parameter accepts the following values:
	• USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference.
	For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics.
	Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics.
	• USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes.
	For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics.
	• ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist.
	• NULL—A partition or subpartition is considered stale when it has any DML changes.
	For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.
	Note that the following two executions are different:
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	
	The first execution uses single quotes to set the preference to the value NULL, whereas the second sets the preference to the default, which is ALLOW_MIXED_FORMAT.

METHOD_OPT	Controls column statistics collection and histogram creation. When setting preference on global, schema, database or dictionary level, only'FOR ALL' syntax is allowed.

Table 197-19 (Cont.) Statistics Preferences

Preference	Description
NO_INVALIDATE	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS Procedure.
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether the database publishes newly gathered statistics after the gathering job completes. You can gather statistics without publishing them immediately. This technique enables you to test new statistics before publishing them.
STALE_PERCENT	Determines the percentage of rows in a table that have to change before the statistics on that table are deemed stale and should be regathered.
STAT_CATEGORY	Specifies which statistics to import or export, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics (default) • SYNOPSES — information to support incremental statistics • REALTIME_STATS — Specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. The value 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.

Security Model

To run this procedure, you must have the `SYSDBA` role or both `ANALYZE ANY DICTIONARY` and `ANALYZE ANY` system privileges.

Exceptions

`ORA-20000`: Insufficient privileges

`ORA-20001`: Invalid or Illegal input values

Usage Notes

All `pname` arguments are of type `VARCHAR2` and values are enclosed in quotes, even when they represent numbers.

Example 197-7 Examples

```
DBMS_STATS.DELETE_DATABASE_PREFS('CASCADE', FALSE);
DBMS_STATS.DELETE_DATABASE_PREFS('ESTIMATE_PERCENT', TRUE);
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to manage optimizer statistics preferences

DELETE_DATABASE_STATS Procedure

This procedure deletes statistics for all the tables in a database.

Syntax

```
DBMS_STATS.DELETE_DATABASE_STATS (
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   force            BOOLEAN  DEFAULT FALSE,
   stat_category    VARCHAR2 DEFAULT DEFAULT_DEL_STAT_CATEGORY);
```

Parameters

Table 197-20 DELETE_DATABASE_STATS Procedure Parameters

Parameter	Description
`stattab`	User statistics table identifier describing from where to delete the statistics. If `stattab` is `NULL`, then the statistics are deleted directly in the dictionary.
`statid`	Identifier (optional) to associate with these statistics within `stattab` (Only pertinent if `stattab` is not `NULL`)
`statown`	Schema containing `stattab` (if different from current schema)

Table 197-20 (Cont.) DELETE_DATABASE_STATS Procedure Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	When the value of this argument is TRUE, deletes statistics of tables in a database even if they are locked
stat_category	Statistics to delete. It accepts multiple values separated by comma: • OBJECT_STATS — table statistics, column statistics and index statistics • SYNOPSES — information to support incremental statistics • REALTIME_STATS — specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. The default is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

To run this procedure, you need to have the SYSDBA role or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.

DELETE_DICTIONARY_STATS Procedure

This procedure deletes statistics for all dictionary schemas ('SYS', 'SYSTEM' and RDBMS component schemas).

Syntax

```
DBMS_STATS.DELETE_DICTIONARY_STATS (
   stattab        VARCHAR2 DEFAULT NULL,
   statid         VARCHAR2 DEFAULT NULL,
   statown        VARCHAR2 DEFAULT NULL,
   no_invalidate  BOOLEAN  DEFAULT to_no_invalidate_type (
                                 get_param('NO_INVALIDATE')),
   stattype       VARCHAR2 DEFAULT 'ALL',
   force          BOOLEAN  DEFAULT FALSE,
   stat_category  VARCHAR2 DEFAULT DEFAULT_DEL_STAT_CATEGORY);
```

Parameters

Table 197-21 DELETE_DICTIONARY_STATS Procedure Parameters

Parameter	Description
stattab	User statistics table identifier describing from where to delete the statistics. If stattab is NULL, then the statistics are deleted directly in the dictionary.
statid	Identifier (optional) to associate with these statistics within stattab (Only pertinent if stattab is not NULL)
statown	Schema containing stattab (if different from current schema)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattype	Statistics type
force	When the value of this argument is TRUE, deletes statistics of tables in a database even if they are locked
stat_category	Statistics to delete. It accepts multiple values separated by comma: • OBJECT_STATS — table statistics, column statistics and index statistics • SYNOPSES — information to support incremental statistics • REALTIME_STATS — Specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. The default is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'

Usage Notes

You must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege to execute this procedure.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20002: Bad user statistics table, may need to upgrade it

DELETE_FIXED_OBJECTS_STATS Procedure

This procedure deletes statistics of all fixed tables.

Syntax

```
DBMS_STATS.DELETE_FIXED_OBJECTS_STATS (
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   force            BOOLEAN  DEFAULT FALSE);
```

Parameters

Table 197-22 DELETE_FIXED_OBJECTS_STATS Procedure Parameters

Parameter	Description
stattab	The user statistics table identifier describing from where to delete the current statistics. If stattab is NULL, the statistics will be deleted directly in the dictionary.
statid	The (optional) identifier to associate with these statistics within stattab. This only applies if stattab is not NULL.
statown	Schema containing stattab (if different from current schema)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Ignores the statistics lock on objects and deletes the statistics if set to TRUE

Usage Notes

You must have the SYSDBA or ANALYZE ANY DICTIONARY system privilege to execute this procedure.

Exceptions

ORA-20000: Insufficient privileges

ORA-20002: Bad user statistics table, may need to upgrade it

DELETE_INDEX_STATS Procedure

This procedure deletes index-related statistics.

Syntax

```
DBMS_STATS.DELETE_INDEX_STATS (
   ownname          VARCHAR2,
   indname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   cascade_parts    BOOLEAN  DEFAULT TRUE,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   stattype         VARCHAR2 DEFAULT 'ALL',
   force            BOOLEAN  DEFAULT FALSE);
   stat_category    VARCHAR2 DEFAULT DEFAULT_DEL_STAT_CATEGORY);
```

Parameters

Table 197-23 DELETE_INDEX_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
indname	Name of the index
partname	Name of the index partition for which to delete the statistics. If the index is partitioned and if partname is NULL, then index statistics are deleted at the global level.
stattab	User statistics table identifier describing from where to delete the statistics. If stattab is NULL, then the statistics are deleted directly from the dictionary.
statid	Identifier (optional) to associate with these statistics within stattab (Only pertinent if stattab is not NULL)
cascade_parts	If the index is partitioned and if partname is NULL, then setting this to TRUE causes the deletion of statistics for this index for all underlying partitions as well
statown	Schema containing stattab (if different than ownname)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattype	Statistics type
force	When value of this argument is TRUE, deletes index statistics even if locked

Table 197-23 (Cont.) DELETE_INDEX_STATS Procedure Parameters

Parameter	Description
stat_category	Statistics to delete. It accepts multiple values separated by comma: • OBJECT_STATS — table statistics, column statistics and index statistics • SYNOPSES — information to support incremental statistics • REALTIME_STATS — Specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. The default is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20005: Object statistics are locked

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

DELETE_PENDING_STATS Procedure

This procedure is used to delete the pending statistics that have been collected but have not been published.

Syntax

```
DBMS_STATS.DELETE_PENDING_STATS (
    ownname     IN  VARCHAR2  DEFAULT USER,
    tabname     IN  VARCHAR2);
```

Parameters

Table 197-24 DELETE_PENDING_STATS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name

Security Model

To run this procedure, you need to have the same privilege for gathering statistics on the tables that will be affected by this procedure. The default owner is the user who runs the procedure.

Exceptions

ORA-20000: Insufficient privileges

Usage Notes

If the parameter `tabname` is `NULL` delete applies to all tables of the specified schema.

Examples

```
DBMS_STATS.DELETE_PENDING_STATS('SH', 'SALES');
```

DELETE_PLSQL_PREFS Procedure

Deletes the named preference from the named PL/SQL function or from all functions in the same package.

Syntax

```
DBMS_STATS.DELETE_PLSQL_PREFS(
    ownname        IN VARCHAR2,
    package_name   IN VARCHAR2,
    function_name  IN VARCHAR2,
    pname          IN VARCHAR2
);
```

Parameters

Table 197-25 DELETE_PLSQL_PREFS Procedure Parameters

Parameter	Description
`ownname`	Owner of the function. If NULL, the current schema is used.
`package_name`	The PL/SQL package name. If NULL, this specifies a stand-alone, top-level function.
`function_name`	Name of the PL/SQL function. The PL/SQL preference will be deleted on this function. If NULL, the preference is deleted on all functions in the package.
`pname`	The preference that you want to delete. Currently only `'dynamic_stats'` is accepted.

Examples

In this example the owner is the current schema and so the `ownname` parameter is set to `NULL`. The dynamic statistics preference for the named function is deleted and the global preference is used for this function.

```
EXEC DELETE_PLSQL_PREFS(NULL, '<PACKAGE_NAME>', '<FUNCTION_NAME>',
'dynamic_stats')
```

In this example, the `function_name` parameter is set to NULL. Therefore the dynamic statistics preference is deleted on all functions within the named PL/SQL package.

```
EXEC DELETE_PLSQL_PREFS('<ownname>', '<PACKAGE_NAME>', NULL,'dynamic_stats')
```

DELETE_PROCESSING_RATE Procedure

This procedure deletes the processing rate of a given statistics source. If the source is not specified, it deletes the statistics of all the sources.

Syntax

```
DBMS_STATS.DELETE_PROCESSING_RATE (
   stat_source      IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-26 DELETE_PROCESSING_RATE Procedure Parameters

Parameter	Description
stat_source	Source of processing rates: • 'MANUAL': values set by the user manually using the SET_PROCESSING_RATE Procedure • 'CALIBRATION' : values collected by the calibration GATHER_PROCESSING_RATE Procedure run explicitly by the user • 'FEEDBACK': values obtained by time feedback

Usage Notes

You require the OPTIMIZER_PROCESSING_RATE role to run this procedure since AUTO DOP uses processing rates to determine the optimal degree of parallelism for a SQL statement.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or illegal input value

DELETE_SCHEMA_PREFS Procedure

This procedure is used to delete the statistics preferences of all the tables owned by the specified owner name.

Syntax

```
DBMS_STATS.DELETE_SCHEMA_PREFS (
    ownname    IN   VARCHAR2,
    pname      IN   VARCHAR2);
```

Parameters

Table 197-27 DELETE_SCHEMA_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name

Table 197-27 (Cont.) DELETE_SCHEMA_PREFS Procedure Parameters

Parameter	Description
pname	Preference name. The existing value for following preferences can be deleted and default preference values will be used: • APPROXIMATE_NDV_ALGORITHM • AUTO_STAT_EXTENSIONS • CASCADE • DEGREE • ESTIMATE_PERCENT • GLOBAL_TEMP_TABLE_STATS • GRANULARITY • INCREMENTAL • INCREMENTAL_LEVEL • INCREMENTAL_STALENESS • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • TABLE_CACHED_BLOCKS

Table 197-28 Statistics Preferences

Preference	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. You can consider a synopsis as an internal management structure that samples distinct values. You can set the following values: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses.

Table 197-28 (Cont.) Statistics Preferences

Preference	Description
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
CASCADE	Determines whether index statistics are collected as part of gathering table statistics.
COORDINATOR_TRIGGER_SHARD	User of each shard uses this preference to determine whether to allow shard coordinator to interact with the statistics gathering in each shards. While gathering the statistics in shard coordinator, if the statistics in one of the shards are not up to date, the shard coordinator will try to trigger the statistics gathering in that shard. By using this preference, user can execute or ignore that command from the shard coordinator. You can set the following values: • TRUE—Allows the shard coordinator trigger the statistics gathering on sharded table in local shard if the statistics on local shard are stale. • FALSE—Ignores the statistics gathering command triggered from the shard coordinator. The default value is FALSE.
DEGREE	Determines the degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	The value determines the percentage of rows to estimate.
METHOD_OPT	Controls column statistics collection and histogram creation. When setting preferences at the global, schema, database, or dictionary level, onlyFOR ALL syntax is allowed.
NO_INVALIDATE	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
GRANULARITY	The value determines granularity of statistics to collect (only pertinent if the table is partitioned)

Table 197-28 (Cont.) Statistics Preferences

Preference	Description
PUBLISH	This value determines whether or not newly gathered statistics will be published once the gather job has completed.
INCREMENTAL	This value determines whether or not the global statistics of a partitioned table will be maintained without doing a full table scan.
INCREMENTAL_LEVEL	This value controls what synopses to collect when INCREMENTAL preference is set to TRUE.

Table 197-28 (Cont.) Statistics Preferences

Preference	Description

INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'.
	
	The parameter accepts the following values:
	• USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference.
	For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics.
	Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics.
	• USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes.
	For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics.
	• ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist.
	• NULL—A partition or subpartition is considered stale when it has any DML changes.
	For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.
	Note that the following two executions are different:
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	
	The first execution uses single quotes to set the preference to the value NULL, whereas the second sets the preference to the default, which is ALLOW_MIXED_FORMAT.

Table 197-28 (Cont.) Statistics Preferences

Preference	Description
ROOT_TRIGGER_PDB	The application PDB user, uses this preference to determine whether to allow the application root to interact with the statics gathering in PDB. During the statistics gathering of a metadata linked table in the application root, if the statistics in a PDB are in stale state, the application root triggers the statistics gathering for the particular PDB. Using this preference, the user can either execute or ignore the command from the application root. You can set the following values: • TRUE—Allows the application root trigger the statistics gathering on metadata linked table in application PDB if the statistics on PDB are stale. • FALSE—Ignores the statistics gathering command triggered from application root. The default value is FALSE. **Note:** CDB root, different from application root, never triggers statistics gathering on the PDBs and it is not controlled by this preference.
STALE_PERCENT	This value determines the percentage of rows in a table that have to change before the statistics on that table are deemed stale and should be regathered.
GLOBAL_TEMP_TABLE_STATS	This controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS Procedure.

Security Model

To run this procedure, you must be the object owner, or have the SYSDBA privilege, or have the ANALYZE ANY system privilege.

Exceptions

ORA-20000: Insufficient privileges / Schema "<schema>" does not exist

ORA-20001: Invalid or Illegal input values

Usage Notes

All arguments are of type `VARCHAR2` and values are enclosed in quotes, even when they represent numbers.

Examples

```
DBMS_STATS.DELETE_SCHEMA_PREFS('SH', 'CASCADE');
DBMS_STATS.DELETE_SCHEMA_PREFS('SH', 'ESTIMATE_PERCENT');
DBMS_STATS.DELETE_SCHEMA_PREFS('SH', 'DEGREE');
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to manage optimizer statistics preferences

DELETE_SCHEMA_STATS Procedure

This procedure deletes statistics for an entire schema.

Syntax

```
DBMS_STATS.DELETE_SCHEMA_STATS (
   ownname          VARCHAR2,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN DEFAULT to_no_invalidate_type (
                                    get_param('NO_INVALIDATE')),
   force            BOOLEAN DEFAULT FALSE,
   stat_category   VARCHAR2 DEFAULT DEFAULT_DEL_STAT_CATEGORY);
```

Parameters

Table 197-29 DELETE_SCHEMA_STATS Procedure Parameters

Parameter	Description
`ownname`	Specifies the name of the schema.
`stattab`	Identifies the table where statistics are stored. If `stattab` is `NULL`, then the procedure deletes statistics directly from the data dictionary.
`statid`	Specifies the identifier (optional) associated with these statistics within `stattab`. This parameter is only relevant if `stattab` is not `NULL`.
`statown`	Specifies the schema containing `stattab` (if different than `ownname`).

Table 197-29 (Cont.) DELETE_SCHEMA_STATS Procedure Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Indicates whether to force the deletion for locked statistics. When the value is TRUE, this procedure deletes table statistics even if locked.
stat_category	Specifies which statistics to process. The following values are supported: • OBJECT_STATS — Specifies table statistics, column statistics, and index statistics. • SYNOPSES — Specifies metadata for incremental statistics. • REALTIME_STATS — Specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. You can specify a list of comma-delimited values. For example, 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

DELETE_SYSTEM_STATS Procedure

This procedure deletes workload statistics (collected using the 'INTERVAL' or 'START' and 'STOP' options) and resets the default to noworkload statistics (collected using 'NOWORKLOAD' option), if stattab is not specified. If stattab is specified, the subprogram deletes all system statistics with the associated statid from the stattab.

Syntax

```
DBMS_STATS.DELETE_SYSTEM_STATS (
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-30 DELETE_SYSTEM_STATS Procedure Parameters

Parameter	Description
stattab	Identifier of the user statistics table where the statistics will be saved
statid	Optional identifier associated with the statistics saved in the stattab
statown	Schema containing stattab (if different from current schema)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20002: Bad user statistics table; may need to be upgraded

Usage Notes

To run this procedure, you need the GATHER_SYSTEM_STATISTICS role.

DELETE_TABLE_PREFS Procedure

This procedure deletes the optimizer statistics preferences of the specified table in the specified schema.

Syntax

```
DBMS_STATS.DELETE_TABLE_PREFS (
    ownname     IN  VARCHAR2,
    tabname     IN  VARCHAR2,
    pname       IN  VARCHAR2);
```

Parameters

Table 197-31 DELETE_TABLE_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name

Table 197-31 (Cont.) DELETE_TABLE_PREFS Procedure Parameters

Parameter	Description
pname	Preference name. The existing value for following preferences can be deleted and default preference values will be used: • APPROXIMATE_NDV_ALGORITHM • AUTO_STAT_EXTENSIONS • CASCADE • DEGREE • ESTIMATE_PERCENT • GRANULARITY • GLOBAL_TEMP_STATS • INCREMENTAL • INCREMENTAL_LEVEL • INCREMENTAL_STALENESS • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • STAT_CATEGORY • TABLE_CACHED_BLOCKS

Table 197-32 Statistics Preferences

Preference	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. Consider a synopsis as an internal management structure that samples distinct values. You can specify the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
CASCADE	Determines whether index statistics are collected as part of gathering table statistics.
DEGREE	Determines the degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	Determines the percentage of rows to estimate.
GRANULARITY	Determines granularity of statistics to collect. This value is only relevant for partitioned tables.
GLOBAL_TEMP_TABLE_STATS	Controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics.

Table 197-32 (Cont.) Statistics Preferences

Preference	Description
INCREMENTAL	This value determines whether or not the global statistics of a partitioned table will be maintained without doing a full table scan.
INCREMENTAL_LEVEL	This value controls what synopses to collect when INCREMENTAL preference is set to TRUE.

Table 197-32 (Cont.) Statistics Preferences

Preference	Description

INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'.
	The parameter accepts the following values:
	• USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference.
	For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics.
	Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics.
	• USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes.
	For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics.
	• ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist.
	• NULL—A partition or subpartition is considered stale when it has any DML changes.
	For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.
	Note that the following two executions are different:
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	The first execution uses single quotes to set the preference to the value NULL, whereas the second sets the preference to the default, which is ALLOW_MIXED_FORMAT.

Table 197-32 (Cont.) Statistics Preferences

Preference	Description
METHOD_OPT	Controls column statistics collection and histogram creation. When setting preference at the global, schema, database, or dictionary level, only 'FOR ALL' syntax is allowed.
NO_INVALIDATE	The value controls the invalidation of dependent cursors of the tables for which statistics are being gathered.
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS Procedure.
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether newly gathered statistics will be published after the statistics gathering job has completed.
STALE_PERCENT	Determines the percentage of rows in a table that have to change before the statistics on that table are deemed stale and should be regathered.
STAT_CATEGORY	Specifies which statistics to import or export, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics (default) • SYNOPSES — information to support incremental statistics • REALTIME_STATS — specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. The value 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Invalid or Illegal input values

Usage Notes

- To run this procedure, you need to connect as owner of the table, be granted ANALYZE privilege on the table, or ANALYZE ANY system privilege.

- All arguments are of type `VARCHAR2` and values are enclosed in quotes, even when they represent numbers.

Examples

```
DBMS_STATS.DELETE_TABLE_PREFS('SH', 'SALES', 'CASCADE');
DBMS_STATS.DELETE_TABLE_PREFS('SH', 'SALES', 'DEGREE');
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to manage optimizer statistics preferences

DELETE_TABLE_STATS Procedure

This procedure deletes table-related statistics.

Syntax

```
DBMS_STATS.DELETE_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   cascade_parts    BOOLEAN  DEFAULT TRUE,
   cascade_columns  BOOLEAN  DEFAULT TRUE,
   cascade_indexes  BOOLEAN  DEFAULT TRUE,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   force            BOOLEAN  DEFAULT FALSE,
   stat_category    VARCHAR2 DEFAULT DEFAULT_DEL_STAT_CATEGORY);
```

Parameters

Table 197-33 DELETE_TABLE_STATS Procedure Parameters

Parameter	Description
`ownname`	Specifies the name of the schema.
`tabname`	Specifies the name of the table to which this column belongs.
`partname`	Specifies the name of the table partition or subpartition from which to get the statistics. If the table is partitioned and if `partname` is `NULL`, then the statistics are retrieved from the global table level.
`stattab`	Identifies the user statistics table where statistics will be retrieved. If `stattab` is `NULL`, then the procedure retrieves statistics directly from the dictionary.
`statid`	Specifies the identifier (optional) associated with these statistics within `stattab`. This parameter is only relevant if `stattab` is not `NULL`.

Table 197-33 (Cont.) DELETE_TABLE_STATS Procedure Parameters

Parameter	Description
cascade_parts	Specifies whether the procedure should operate on underlying partitions. If the table is partitioned, and if partname is NULL, then specifying TRUE deletes statistics for underlying partitions.
cascade_columns	Indicates whether to invoke the DELETE_COLUMN_STATS procedure. If TRUE, then this procedure calls DELETE_COLUMN_STATS for all underlying columns.
cascade_indexes	Indicates whether to invoke the DELETE_INDEX_STATS procedure. If TRUE, then this procedure calls DELETE_INDEX_STATS for all underlying columns.
statown	Specifies the schema containing stattab (if different than ownname).
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Indicates whether to force the deletion for locked statistics. When the value is TRUE, this procedure deletes table statistics even if locked.
stat_category	Specifies which statistics to process. The following values are supported: • OBJECT_STATS — specifies table statistics, column statistics, and index statistics. • SYNOPSES — specifies metadata for incremental statistics. • REALTIME_STATS — specifies only real-time statistics. • MODELS — supports import, export, and deletion for regression models in real-time stats. You can specify a list of comma-delimited values. For example, 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS, MODELS'.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20002: Bad user statistics table, may need to upgrade it

ORA-20005: Object statistics are locked

DIFF_TABLE_STATS_IN_HISTORY Function

This function compares statistics for a table as of two specified timestamps.

Syntax

```
DBMS_STATS.DIFF_TABLE_STATS_IN_HISTORY(
   ownname        IN  VARCHAR2,
   tabname        IN  VARCHAR2,
   time1          IN  TIMESTAMP WITH TIME ZONE,
   time2          IN  TIMESTAMP WITH TIME ZONE DEFAULT NULL,
   pctthreshold   IN  NUMBER                   DEFAULT 10)
RETURN DiffRepTab pipelined;
```

Parameters

Table 197-34 DIFF_TABLE_STATS_IN_HISTORY Function Parameters

Parameter	Description
ownname	Specifies the owner of the table. Specify NULL for current schema.
tabname	Specifies the table for which statistics are to be compared.
time1	Specifies the first timestamp for comparison.
time2	Specifies the second timestamp for comparison.
pctthreshold	Specifies the threshold limit. The function reports differences in statistics only if the change percentage exceeds this limit. The default value is 10.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Usage Notes

If the second timestamp is NULL, then the function compares the current statistics in the data dictionary with the statistics as of the first timestamp.

DIFF_TABLE_STATS_IN_PENDING Function

This function compares pending statistics to either the current statistics in the data dictionary, or user-specified historical statistics.

Syntax

```
DBMS_STATS.DIFF_TABLE_STATS_IN_PENDING(
      ownname        IN  VARCHAR2,
      tabname        IN  VARCHAR2,
      timestamp      IN  TIMESTAMP WITH TIME ZONE,
      pctthreshold   IN  NUMBER  DEFAULT 10)
RETURN DiffRepTab pipelined;
```

Parameters

Table 197-35 DIFF_TABLE_STATS_IN_PENDING Function Parameters

Parameter	Description
ownname	Owner of the table. Specify NULL for the current schema.
tabname	Table for which statistics are to be compared.
timestamp	Timestamp in the statistics history that corresponds to the desired statistics. If the timestamp is NULL, then this function compares the current statistics in the dictionary with the pending statistics (default).
pctthreshold	Limit for reporting. The function reports difference in statistics only if it exceeds the specified limit. The default value is 10.

Security Model

To invoke this procedure you must be owner of the table, or you must have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table, or you must have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

DIFF_TABLE_STATS_IN_STATTAB Function

This function compares table statistics from two sources.

The function can obtain statistics from the following sources:

- Two user statistics tables
- A single user statistics table containing two sets of statistics that can be identified using statids
- A user statistics table and dictionary

The function also compares the statistics of the dependent objects: indexes, columns, and partitions. It displays statistics of the objects from both sources when the difference between those statistics exceeds a certain threshold (%). You can specify this threshold as an argument to the function. The function uses the statistics corresponding to the first source (stattab1 or time1) as the basis for computing the difference percentage.

Syntax

```
DBMS_STATS.DIFF_TABLE_STATS_IN_STATTAB(
   ownname         IN  VARCHAR2,
   tabname         IN  VARCHAR2,
   stattab1        IN  VARCHAR2,
   stattab2        IN  VARCHAR2 DEFAULT NULL,
   pctthreshold    IN  NUMBER   DEFAULT 10,
   statid1         IN  VARCHAR2 DEFAULT NULL,
   statid2         IN  VARCHAR2 DEFAULT NULL,
   stattab1own     IN  VARCHAR2 DEFAULT NULL,
   stattab2own     IN  VARCHAR2 DEFAULT NULL)
RETURN DiffRepTab pipelined;
```

Parameters

Table 197-36 DIFF_TABLE_STATS_IN_STATTAB Function Parameters

Parameter	Description
ownname	Specifies the owner of the table. Specify NULL for current schema.
tabname	Specifies the table for which statistics are to be compared.
stattab1	Specifies the user statistics table 1.
stattab2	Specifies the user statistics table 2. If NULL, the function compares statistics in stattab1 with current statistics in the data dictionary. This is the default. To compare two sets within the statistics table, specify the same table as stattab1 (see statid below).
pctthreshold	Specifies the percent thresholds for comparison. The function reports difference in statistics only if it exceeds this limit. The default value is 10.
stadid1	(optional) Identifies statistics set within stattab1.
stadid2	(optional) Identifies statistics set within stattab2.
stattab1own	Specifies the schema containing stattab1 (if other than ownname).
stattab2own	Specifies the schema containing stattab2 (if other than ownname).

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

DROP_ADVISOR_TASK Procedure

This procedure drops the specified Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.DROP_ADVISOR_TASK (
  task_name IN VARCHAR2);
```

Parameters

Table 197-37 DROP_ADVISOR_TASK Procedure Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Example 197-8 Dropping an Optimizer Statistics Advisor Task

This example drops the Optimizer Statistics Advisor task named my_task:

```
EXEC DBMS_STATS.DROP_ADVISOR_TASK('my_task');
```

DROP_EXTENDED_STATS Procedure

This function drops the statistics entry that is created for the user specified extension.

This cancels the effects of the CREATE_EXTENDED_STATS Function.

Syntax

```
DBMS_STATS.DROP_EXTENDED_STATS (
   ownname    VARCHAR2,
   tabname    VARCHAR2,
   extension  VARCHAR2);
```

Parameters

Table 197-38 DROP_EXTENDED_STATS Procedure Parameters

Parameter	Description
ownname	Owner name of a table
tabname	Name of the table
extension	Can be either a column group or an expression. Suppose the specified table has two column c1, c2. An example column group can be "(c1, c2)" and an example expression can be "(c1 + c2)".

Exceptions

- ORA-20000: Insufficient privileges or extension does not exist
- ORA-20001: Error when processing extension

Usage Notes

- To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.
- If no extended statistics set is created for the extension, this function throws an error.

DROP_STAT_TABLE Procedure

This procedure drops a user statistics table.

Syntax

```
DBMS_STATS.DROP_STAT_TABLE (
   ownname VARCHAR2,
   stattab VARCHAR2);
```

Parameters

Table 197-39 DROP_STAT_TABLE Procedure Parameters

Parameter	Description
ownname	Name of the schema
stattab	User statistics table identifier

Exceptions

ORA-20000: Table does not exists or insufficient privileges.

Usage Notes

To invoke this procedure you need the privileges for dropping the specified table.

EXECUTE_ADVISOR_TASK Function

This function executes a previously created Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.EXECUTE_ADVISOR_TASK (
   task_name        IN   VARCHAR2,
   execution_name   IN   VARCHAR2 := NULL)
 RETURN VARCHAR2;
```

Parameters

Table 197-40 EXECUTE_ADVISOR_TASK Parameters

Parameter	Description
task_name	Name of the Optimizer Statistics Advisor task.
execution_name	A name that qualifies and identifies an advisor execution. If not specified, then the advisor automatically generates it. If the specified execution conflicts with the name of an existing execution, then the function returns an error.

Security Model

Note the following:

- To execute this subprogram, you must have the `ADVISOR` privilege.
- You must be the owner of the task.
- You can execute this subprogram for `AUTO_STATS_ADVISOR_TASK`, which is predefined.
- This subprogram executes using invoker's rights.

The results of performing this task depend on the privileges of the executing user:

- `SYSTEM` level

  Only users with both the `ANALYZE ANY` and `ANALYZE ANY DICTIONARY` privileges can perform this task on system-level rules.
- Operation level

  The results depend on the following privileges:

  – Users with both the `ANALYZE ANY` and `ANALYZE ANY DICTIONARY` privileges can perform this task for all statistics operations.

  – Users with the `ANALYZE ANY` privilege but *not* the `ANALYZE ANY DICTIONARY` privilege can perform this task for statistics operations related to any schema except `SYS`.

  – Users with the `ANALYZE ANY DICTIONARY` privilege but *not* the `ANALYZE ANY` privilege can perform this task for statistics operations related to their own schema and the `SYS` schema.

  – Users with neither the `ANALYZE ANY` nor the `ANALYZE ANY DICTIONARY` privilege can only perform this operation for statistics operations relating to their own schema.
- Object level

  Users can perform this task for any object for which they have statistics collection privileges.

Exceptions

- `ORA-20000`: Insufficient privileges
- `ORA-20001`: Invalid input values
- `ORA-20012`: Optimizer Statistics Advisor errors

Returns

This function returns the name of the new execution.

Usage Notes

The results of the execution depend on user privileges and the type of rules:

- System

  To perform the operation on system-level rules, you must have both the `ANALYZE ANY` and `ANALYZE ANY DICTIONARY` privileges.
- Operation

If you have the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges, then you can execute this function for all operations. If you have only the ANALYZE ANY privilege, then you can execute this function for operations related to any schemas except SYS. If you have only the ANALYZE ANY DICTIONARY privilege, then you can execute this function for operations related to any schemas, including SYS. If you have neither the ANALYZE ANY nor the ANALYZE ANY DICTIONARY privilege, then you can execute this function only for operations in your own schema.

- Object

  If you have the privilege to collect statistics for an object, then you can execute this function for the object.

Example 197-9 Creating and Executing a Task

This example creates an Optimizer Statistics Advisor task named my_task, and then executes it.

```
DECLARE
  v_tname   VARCHAR2(128) := 'my_task';
BEGIN
  -- create a task
  v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);
  -- execute the task
  v_tname := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
END;
```

EXPORT_COLUMN_STATS Procedure

This procedure exports statistics for a specified column and stores them in the user statistics table identified by stattab.

Syntax

```
DBMS_STATS.EXPORT_COLUMN_STATS (
   ownname  VARCHAR2,
   tabname  VARCHAR2,
   colname  VARCHAR2,
   partname VARCHAR2 DEFAULT NULL,
   stattab  VARCHAR2,
   statid   VARCHAR2 DEFAULT NULL,
   statown  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-41 EXPORT_COLUMN_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
tabname	Name of the table to which this column belongs
colname	Name of the column or extension

Table 197-41 (Cont.) EXPORT_COLUMN_STATS Procedure Parameters

Parameter	Description
partname	Name of the table partition. If the table is partitioned and if partname is NULL, then global and partition column statistics are exported.
stattab	User statistics table identifier describing where to store the statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

EXPORT_DATABASE_PREFS Procedure

This procedure is used to export the statistics preferences of all the tables, excluding the tables owned by Oracle. These tables can by included by passing TRUE for the add_sys parameter.

Syntax

```
DBMS_STATS.EXPORT_DATABASE_PREFS (
    stattab    IN  VARCHAR2,
    statid     IN  VARCHAR2 DEFAULT NULL,
    statown    IN  VARCHAR2 DEFAULT NULL
    add_sys    IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-42 EXPORT_DATABASE_PREFS Procedure Parameters

Parameter	Description
stattab	Statistics table name to where statistics should be exported
statid	(Optional) Identifier to associate with these statistics within stattab
statown	Schema containing stattab (if other than ownname)
add_sys	Value TRUE will include the Oracle-owned tables

Exceptions

ORA-20000: Insufficient privileges

Usage Notes

- To run this procedure, you need to have the SYSDBA role, or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.
- All arguments are of type VARCHAR2 and values are enclosed in quotes.
- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.EXPORT_DATABASE_PREFS('STATTAB', statown=>'SH');
```

EXPORT_DATABASE_STATS Procedure

This procedure exports statistics for all objects in the database and stores them in the user statistics tables identified by statown.stattab.

Syntax

```
DBMS_STATS.EXPORT_DATABASE_STATS (
   stattab         VARCHAR2,
   statid          VARCHAR2 DEFAULT NULL,
   statown         VARCHAR2 DEFAULT NULL,
   stat_category   VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-43 EXPORT_DATABASE_STATS Procedure Parameters

Parameter	Description
stattab	User statistics table identifier describing where to store the statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different from current schema)
stat_category	Specifies what statistics to import, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics (default). • SYNOPSES — information to support incremental statistics. • REALTIME_STATS — specifies only real-time statistics. If 'OBJECT_STATS, SYNOPSES' is specified, table statistics, column statistics, index statistics and synopses are deleted.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

EXPORT_DICTIONARY_STATS Procedure

This procedure exports statistics for all data dictionary schemas (`SYS`, `SYSTEM`, and RDBMS component schemas) and stores them in the user statistics table identified by `stattab`.

Syntax

```
DBMS_STATS.EXPORT_DICTIONARY_STATS (
   stattab         VARCHAR2,
   statid          VARCHAR2 DEFAULT NULL,
   statown         VARCHAR2 DEFAULT NULL,
   stat_category   VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-44 EXPORT_DICTIONARY_STATS Procedure Parameters

Parameter	Description
`stattab`	User statistics table identifier describing where to store the statistics
`statid`	Identifier (optional) to associate with these statistics within `stattab`
`statown`	Schema containing `stattab` (if different from current schema)
`stat_category`	Specifies what statistics to import, accepting multiple values separated by a comma. Values supported: • `OBJECT_STATS` — table statistics, column statistics and index statistics (default) • `SYNOPSES` — information to support incremental statistics • `REALTIME_STATS` — specifies only real-time statistics. If `'OBJECT_STATS, SYNOPSES'` is specified, table statistics, column statistics, index statistics and synopses are deleted.

Security Model

To invoke this procedure you must be owner of the table or have the `ANALYZE ANY` privilege. For objects owned by `SYS`, you must be either the owner of the table or have either the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

`ORA-20002`: Bad user statistics table, may need to upgrade it

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

EXPORT_FIXED_OBJECTS_STATS Procedure

This procedure exports statistics for fixed tables and stores them in the user statistics table identified by `stattab`.

Syntax

```
DBMS_STATS.EXPORT_FIXED_OBJECTS_STATS (
   stattab  VARCHAR2,
   statid   VARCHAR2 DEFAULT NULL,
   statown  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-45 EXPORT_FIXED_OBJECTS_STATS Procedure Parameters

Parameter	Description
`stattab`	User statistics table identifier describing where to store the statistics
`statid`	Identifier (optional) to associate with these statistics within `stattab`
`statown`	Schema containing `stattab` (if different from current schema)

Security Model

To invoke this procedure you must be owner of the table or have the `ANALYZE ANY` privilege. For objects owned by `SYS`, you must be either the owner of the table or have either the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

`ORA-20002`: Bad user statistics table, may need to upgrade it

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

EXPORT_INDEX_STATS Procedure

This procedure retrieves statistics for a particular index and stores them in the user statistics table identified by `stattab`.

Syntax

```
DBMS_STATS.EXPORT_INDEX_STATS (
   ownname  VARCHAR2,
   indname  VARCHAR2,
   partname VARCHAR2 DEFAULT NULL,
   stattab  VARCHAR2,
   statid   VARCHAR2 DEFAULT NULL,
   statown  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-46 EXPORT_INDEX_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
indname	Name of the index
partname	Name of the index partition. If the index is partitioned and if partname is NULL, then global and partition index statistics are exported.
stattab	User statistics table identifier describing where to store the statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

- To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.
- Oracle does not support export or import of statistics across databases of different character sets.

EXPORT_PENDING_STATS Procedure

This procedure is used to export the statistics gathered and stored as pending.

Syntax

```
DBMS_STATS.EXPORT_PENDING_STATS (
    ownname    IN  VARCHAR2  DEFAULT USER,
    tabname    IN  VARCHAR2,
    stattab    IN  VARCHAR2,
    statid     IN  VARCHAR2 DEFAULT NULL,
    statown    IN  VARCHAR2 DEFAULT USER);
```

Parameters

Table 197-47 EXPORT_PENDING_STATS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name
stattab	Statistics table name to where to export the statistics
statid	(Optional) Identifier to associate with these statistics within stattab
statown	Schema containing stattab (if other than ownname)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

- If the parameter tabname is NULL then export applies to all tables of the specified schema.
- The default owner/schema is the user who runs the procedure.
- To run this procedure, you need to have the same privilege for gathering statistics on the tables that will be touched by this procedure.
- All arguments are of type VARCHAR2 and values are enclosed in quotes.
- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.EXPORT_PENDING_STATS(NULL, NULL, 'MY_STAT_TABLE');
```

EXPORT_SCHEMA_PREFS Procedure

This procedure is used to export the statistics preferences of all the tables owned by the specified owner name.

Syntax

```
DBMS_STATS.EXPORT_SCHEMA_PREFS (
    ownname    IN  VARCHAR2,
    stattab    IN  VARCHAR2,
    statid     IN  VARCHAR2 DEFAULT NULL,
    statown    IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-48 EXPORT_SCHEMA_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
stattab	Statistics table name to where to export the statistics
statid	(Optional) Identifier to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

- To run this procedure, you need to connect as owner, or have the SYSDBA privilege, or have the ANALYZE ANY system privilege.
- All arguments are of type VARCHAR2 and values are enclosed in quotes.

- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.EXPORT_SCHEMA_PREFS('SH', 'STAT');
```

EXPORT_SCHEMA_STATS Procedure

This procedure exports statistics for all objects in the schema identified by `ownname` and stores them in the user statistics tables identified by `stattab`.

Syntax

```
DBMS_STATS.EXPORT_SCHEMA_STATS (
   ownname         VARCHAR2,
   stattab         VARCHAR2,
   statid          VARCHAR2 DEFAULT NULL,
   statown         VARCHAR2 DEFAULT NULL,
   stat_category   VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-49 EXPORT_SCHEMA_STATS Procedure Parameters

Parameter	Description
`ownname`	Specifies the name of the schema.
`stattab`	Identifies the user statistics table in which to store the exported statistics.
`statid`	Specifies the identifier (optional) associated with these statistics within `stattab`.
`statown`	Specifies the schema containing `stattab` (if different than `ownname`).
`stat_category`	Specifies which statistics to process. The following values are supported: • `OBJECT_STATS` — specifies table statistics, column statistics, and index statistics • `SYNOPSES` — specifies metadata for incremental statistics • `REALTIME_STATS` — specifies only real-time statistics You can specify a list of comma-delimited values. For example, '`OBJECT_STATS, SYNOPSES`' specifies table statistics, column statistics, index statistics, and synopses. The default value is '`OBJECT_STATS, SYNOPSES, REALTIME_STATS`'.

Security Model

To invoke this procedure you must be owner of the table or have the `ANALYZE ANY` privilege. For objects owned by `SYS`, you must be either the owner of the table or have either the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

EXPORT_SYSTEM_STATS Procedure

This procedure retrieves system statistics and stores them in the user statistics table, identified by `stattab`.

Syntax

```
DBMS_STATS.EXPORT_SYSTEM_STATS (
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-50 EXPORT_SYSTEM_STATS Procedure Parameters

Parameter	Description
`stattab`	Identifier of the user statistics table that describes where the statistics will be stored
`statid`	Optional identifier associated with the statistics stored from the `stattab`
`statown`	Schema containing `stattab` (if different from current schema)

Security Model

To run this procedure, you must have the `GATHER_SYSTEM_STATISTICS` role.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

`ORA-20002`: Bad user statistics table; may need to be upgraded

`ORA-20003`: Unable to export system statistics

Usage Notes

Oracle Database does not support the export or import of statistics across databases of different character sets.

EXPORT_TABLE_PREFS Procedure

This procedure is used to export the statistics preferences of the specified table in the specified schema into the specified statistics table.

Syntax

```
DBMS_STATS.EXPORT_TABLE_PREFS (
    ownname    IN  VARCHAR2,
    tabname    IN  VARCHAR2,
    stattab    IN  VARCHAR2,
```

```
    statid     IN  VARCHAR2 DEFAULT NULL,
    statown    IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-51 EXPORT_TABLE_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name
stattab	Statistics table name where to export the statistics
statid	Optional identifier to associate with these statistics within stattab
statown	Schema containing stattab (if other than ownname)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

- To run this procedure, you need to connect as owner of the table, or have the ANALYZE ANY system privilege.
- All arguments are of type VARCHAR2 and values are enclosed in quotes.
- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.EXPORT_TABLE_PREFS('SH', 'SALES', 'STAT');
```

EXPORT_TABLE_STATS Procedure

This procedure exports statistics for a specified table (including associated index statistics) and stores them in the user statistics table identified by stattab.

Syntax

```
DBMS_STATS.EXPORT_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2,
   statid           VARCHAR2 DEFAULT NULL,
   cascade          BOOLEAN  DEFAULT TRUE,
   statown          VARCHAR2 DEFAULT NULL,
   stat_category    VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-52 EXPORT_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	Specifies the name of the schema.
tabname	Specifies the name of the table.
partname	Specifies the name of the table partition. If the table is partitioned, and if partname is NULL, then the procedure exports global and partition statistics.
stattab	Specifies the identifier (optional) associated with these statistics within stattab.
statid	Specifies the identifier (optional) associated with these statistics within stattab.
cascade	Indicates whether to export column and index statistics. If TRUE, then the procedure exports column and index statistics for the specified table. This is the default.
statown	Specifies the schema containing stattab (if different than ownname).
stat_category	Specifies which statistics to process. The following values are supported: • OBJECT_STATS — specifies table statistics, column statistics, and index statistics • SYNOPSES — specifies metadata for incremental statistics • REALTIME_STATS — specifies only real-time statistics You can specify a list of comma-delimited values. For example, 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS'.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

FLUSH_DATABASE_MONITORING_INFO Procedure

This procedure saves monitoring information for all tables in the dictionary. The database immediately updates corresponding entries in the `*_TAB_MODIFICATIONS`, `*_TAB_STATISTICS` and `*_IND_STATISTICS` views.

Syntax

```
DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO;
```

Security Model

The `ANALYZE_ANY` system privilege is required to run this procedure.

Exceptions

`ORA-20000`: Insufficient privileges

Usage Notes

Starting in Oracle Database 12c Release 2 (12.2), you do not need to call `FLUSH_DATABASE_MONITORING_INFO` to view the latest information in `*_TAB_STATISTICS` and `*_IND_STATISTICS` because these views show statistics cached in the SGA and stored on disk. Because the `GATHER_*_STATS` procedures internally save monitoring information to disk, it is not necessary to run this procedure before gathering statistics.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set optimizer statistics preferences

GATHER_DATABASE_STATS Procedures

This procedure gathers statistics for all objects in the database.

Syntax

```
DBMS_STATS.GATHER_DATABASE_STATS (
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                      (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   objlist          OUT      ObjectTab,
   statown          VARCHAR2 DEFAULT NULL,
   gather_sys       BOOLEAN  DEFAULT TRUE,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   obj_filter_list ObjectTab DEFAULT NULL);
```

```
DBMS_STATS.GATHER_DATABASE_STATS (
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                        (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   statown          VARCHAR2 DEFAULT NULL,
   gather_sys       BOOLEAN  DEFAULT TRUE,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   obj_filter_list ObjectTab DEFAULT NULL);
```

Parameters

Table 197-53 GATHER_DATABASE_STATS Procedure Parameters

Parameter	Description
estimate_percent	Determines the percentage of rows to sample. The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for optimal statistics. This is the default. You can change the default value using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Determines whether the database uses random block sampling (TRUE) or random row sampling (FALSE). The default is FALSE. Random block sampling is more efficient, but if the data is not randomly distributed on disk, then sample values may be somewhat correlated. This parameter is only relevant when estimating statistics.
method_opt	When setting preference on global, schema, database or dictionary level, only'FOR ALL' syntax is allowed.: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms. – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO.The value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Table 197-53 (Cont.) GATHER_DATABASE_STATS Procedure Parameters

Parameter	Description
degree	Determines the degree of parallelism used for gathering statistics. The default for degree is NULL. NULL means to use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Change the default using the SET_DATABASE_PREFS, SET_GLOBAL_PREFS, SET_SCHEMA_PREFS, and SET_TABLE_PREFS procedures. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. The degree is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters), according to the size of the object. When using DEGREE=>NULL, DEGREE=>*n*, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution when the size of the object does not warrant parallel execution.
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics
cascade	Determines whether to collect index statistics as part of gathering table statistics. Specifying this option is equivalent to running the GATHER_INDEX_STATS procedure on each index of the table. Use the constant DBMS_STATS.AUTO_CASCADE to enable the database to determine whether index statistics need to be collected. This is the default. You can change the default using the SET_DATABASE_PREFS, SET_GLOBAL_PREFS, SET_SCHEMA_PREFS, and SET_TABLE_PREFS procedures.
stattab	User statistics table identifier describing where to save the current statistics. The statistics table is assumed to reside in the same schema as the object being analyzed, so there must be one such table in each schema to use this option.
statid	Identifier (optional) to associate with these statistics within stattab.

Table 197-53 (Cont.) GATHER_DATABASE_STATS Procedure Parameters

Parameter	Description
options	Specifies which objects require statistics to be gathered. Valid values are as follows: • GATHER — Gathers statistics on all objects in the database. This is the default. • GATHER AUTO — Gathers all necessary statistics automatically. The database implicitly determines which objects need new statistics and determines how to gather those statistics. When GATHER AUTO is specified, the only additional valid parameters are comp_id, no_invalidate, stattab, statid, and statown; all other parameter settings will be ignored. Also, the database returns a list of objects processed. • GATHER STALE — Gathers statistics on stale objects by querying the *_TAB_MODIFICATIONS views. Also, the database returns a list of objects found to be stale. • GATHER EMPTY — Gathers statistics on objects that currently have no statistics. Also, the database returns a list of objects found to have no statistics. • LIST AUTO — Returns a list of objects to be processed with GATHER AUTO. • LIST STALE — Returns list of stale objects as determined by looking at the *_TAB_MODIFICATIONS views. • LIST EMPTY — Returns a list of objects that currently have no statistics.
objlist	List of objects found to be stale or empty
statown	Schema containing stattab (if different from current schema)
gather_sys	Gathers statistics on the objects owned by the SYS user.
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. If set to TRUE, then the database not invalidate dependent cursors. If set to FALSE, then the procedure invalidates dependent cursors immediately.
obj_filter_list	A list of object filters. When provided, GATHER_DATABASE_STATS will gather statistics only on objects which satisfy at least one object filter in the list as needed. In a single object filter, we can specify the constraints on the object attributes. The attribute values specified in the object filter are case- insensitive unless double-quoted. Wildcard is allowed in the attribute values. Suppose non-NULL values s1, s2, ... are specified for attributes a1, a2, ... in one object filter. An object o is said to satisfy this object filter if (o.a1 like s1) or (o.a2 like s2) or ... is true.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Bad input value

Usage Notes

To run this procedure, you need to have the SYSDBA role or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.

If the GATHER AUTO option is used then frequency histograms will be created using a sample rather than a full scan. The following scenario shows where GATHER AUTO is used:

- A table is created like this: CREATE TABLE NEWTAB as SELECT * FROM .....

  This will create statistics on NEWTAB but no histograms.
- Next, the DBA creates the histograms using GATHER AUTO on gather_table_stats.
- The FREQUENCY histograms on NEWTAB will be created using a sample rather than a full table scan.

GATHER_DICTIONARY_STATS Procedure

This procedure gathers statistics for dictionary schemas SYS, SYSTEM and schemas of RDBMS components.

Syntax

```
DBMS_STATS.GATHER_DICTIONARY_STATS (
   comp_id          VARCHAR2 DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                        (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER AUTO',
   objlist    OUT   ObjectTab,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   obj_filter_list ObjectTab DEFAULT NULL);

DBMS_STATS.GATHER_DICTIONARY_STATS (
   comp_id           VARCHAR2 DEFAULT NULL,
   estimate_percent  NUMBER DEFAULT
                        to_estimate_percent_type(GET_PARAM('ESTIMATE_PERCENT')),
   block_sample      BOOLEAN DEFAULT FALSE,
   method_opt        VARCHAR2 DEFAULT GET_PARAM('METHOD_OPT'),
   degree            NUMBER DEFAULT to_degree_type(GET_PARAM('DEGREE')),
   granularity       VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade           BOOLEAN DEFAULT to_cascade_type(GET_PARAM('CASCADE')),
   stattab           VARCHAR2 DEFAULT NULL,
   statid            VARCHAR2 DEFAULT NULL,
   options           VARCHAR2 DEFAULT 'GATHER AUTO',
   statown           VARCHAR2 DEFAULT NULL,
   no_invalidate     BOOLEAN DEFAULT
                             to_no_invalidate_type(get_param('NO_INVALIDATE')),
   obj_filter_list   ObjectTab DEFAULT NULL);
```

Parameters

Table 197-54 GATHER_DICTIONARY_STATS Procedure Parameters

Parameter	Description
comp_id	Component id of the schema to analyze. NULL results in the analysis of schemas for all RDBMS components. Refer to the COMP_ID column of DBA_REGISTRY view. The procedure always gather statistics on SYS and SYSTEM schemas regardless of this argument.
estimate_percent	Percentage of rows to estimate (NULL means compute). The valid range is [0.000001,100]. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to have Oracle determine the appropriate sample size for good statistics. This is the default.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Determines whether or not to use random block sampling instead of random row sampling. Random block sampling is more efficient, but if the data is not randomly distributed on disk then the sample values may be somewhat correlated. Only pertinent when performing estimate statistics.
method_opt	The method options. This parameter accepts the following values: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms. – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.

Table 197-54 (Cont.) GATHER_DICTIONARY_STATS Procedure Parameters

Parameter	Description
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - gathers subpartition-level statistics
cascade	Gathers statistics on indexes also. Index statistics gathering will not be parallelized. Using this option is equivalent to running the GATHER_INDEX_STATS Procedure on each of the indexes in the schema in addition to gathering table and column statistics. Use the constant DBMS_STATS.AUTO_CASCADE to have Oracle determine whether index statistics to be collected or not. This is the default.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics
statid	The (optional) identifier to associate with these statistics within stattab

Table 197-54 (Cont.) GATHER_DICTIONARY_STATS Procedure Parameters

Parameter	Description
options	Specifies which objects require statistics to be gathered. Valid values are as follows: • GATHER AUTO — Gathers all necessary statistics automatically. This is the default. The database implicitly determines which objects need new statistics and determines how to gather those statistics. When GATHER AUTO is specified, the only additional valid parameters are comp_id, no_invalidate, stattab, statid, and statown; all other parameter settings will be ignored. Also, the database returns a list of objects processed. • GATHER — Gathers statistics on all objects in the relevant schema. • GATHER STALE — Gathers statistics on stale objects by querying the *_TAB_MODIFICATIONS views. Also, the database returns a list of objects found to be stale. • GATHER EMPTY — Gathers statistics on objects that currently have no statistics. Also, the database returns a list of objects found to have no statistics. • LIST AUTO — Returns a list of objects to be processed with GATHER AUTO. • LIST STALE — Returns list of stale objects as determined by looking at the *_TAB_MODIFICATIONS views. • LIST EMPTY — Returns a list of objects that currently have no statistics.
objlist	The list of objects found to be stale or empty.
statown	Schema containing stattab, if different from the current schema.
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
obj_filter_list	A list of object filters. When provided, this will gather statistics only on objects which satisfy at least one object filter in the list as needed. In a single object filter, we can specify the constraints on the object attributes. The attribute values specified in the object filter are case-insensitive unless double-quoted. Wildcard is allowed in the attribute values. Suppose non-NULL values s1, s2, ... are specified for attributes a1, a2, ... in one object filter. An object o is said to satisfy this object filter if (o.a1 like s1) and (o.a2 like s2) and ... is true. See Applying an Object Filter List.

Usage Notes

You must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege to execute this procedure.

If the GATHER AUTO option is used then frequency histograms will be created using a sample rather than a full scan. The following scenario shows where GATHER AUTO is used:

- A table is created like this: CREATE TABLE NEWTAB as SELECT * FROM .....

  This will create statistics on NEWTAB but no histograms.
- Next, the DBA creates the histograms using GATHER AUTO on gather_table_stats.
- The FREQUENCY histograms on NEWTAB will be created using a sample rather than a full table scan.

Exceptions

ORA-20000: Index does not exist or insufficient privileges

ORA-20001: Bad input value

ORA-20002: Bad user statistics table, may need to upgrade it

GATHER_FIXED_OBJECTS_STATS Procedure

This procedure gathers statistics for all fixed objects (dynamic performance tables).

Syntax

```
DBMS_STATS.GATHER_FIXED_OBJECTS_STATS (
   stattab        VARCHAR2 DEFAULT NULL,
   statid         VARCHAR2 DEFAULT NULL,
   statown        VARCHAR2 DEFAULT NULL,
   no_invalidate  BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')));
```

Parameters

Table 197-55 GATHER_FIXED_OBJECTS_STATS Procedure Parameters

Parameter	Description
stattab	User statistics table identifier describing where to save the current statistics
statid	Identifier to associate with these statistics within stattab (optional)
statown	Schema containing stattab (if different from current schema)
no_invalidate	Does not invalidate the dependent cursors if set to TRUE. The procedure invalidates the dependent cursors immediately if set to FALSE. Use DBMS_STATS.AUTO_INVALIDATE. to have Oracle decide when to invalidate dependent cursors. This is the default. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Usage Notes

You must have the SYSDBA or ANALYZE ANY DICTIONARY system privilege to execute this procedure.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Bad input value

ORA-20002: Bad user statistics table, may need to upgrade it

GATHER_INDEX_STATS Procedure

This procedure gathers index statistics. It attempts to parallelize as much of the work as possible.

Restrictions are described in the individual parameters. This operation will not parallelize with certain types of indexes, including cluster indexes, domain indexes, and bitmap join indexes. The granularity and no_invalidate arguments are not relevant to these types of indexes.

Syntax

```
DBMS_STATS.GATHER_INDEX_STATS (
   ownname                 VARCHAR2,
   indname                 VARCHAR2,
   partname                VARCHAR2 DEFAULT NULL,
   estimate_percent        NUMBER   DEFAULT to_estimate_percent_type
                                        (GET_PARAM('ESTIMATE_PERCENT')),
   stattab                 VARCHAR2 DEFAULT NULL,
   statid                  VARCHAR2 DEFAULT NULL,
   statown                 VARCHAR2 DEFAULT NULL,
   degree                  NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity             VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   no_invalidate           BOOLEAN  DEFAULT to_no_invalidate_type
                                       (GET_PARAM('NO_INVALIDATE')),
   force                   BOOLEAN  DEFAULT FALSE,
   vector_index_options    JSON     DEFAULT NULL
);
```

Parameters

Table 197-56 GATHER_INDEX_STATS Procedure Parameters

Parameter	Description
ownname	Schema of index to analyze
indname	Name of index
partname	Name of partition
estimate_percent	Percentage of rows to estimate (NULL means compute). The valid range is [0.000001,100]. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to have Oracle determine the appropriate sample size for good statistics. This is the default.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics

Table 197-56 (Cont.) GATHER_INDEX_STATS Procedure Parameters

Parameter	Description
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics.
no_invalidate	Does not invalidate the dependent cursors if set to TRUE. The procedure invalidates the dependent cursors immediately if set to FALSE. Use DBMS_STATS.AUTO_INVALIDATE. to have Oracle decide when to invalidate dependent cursors. This is the default. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Gather statistics on object even if it is locked
vector_index_options	Vector index control options for gather stats on vector indexes.

Exceptions

ORA-20000: Index does not exist or insufficient privileges

ORA-20001: Bad input value

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

GATHER_PROCESSING_RATE Procedure

This procedure starts the job of gathering the processing rates which end after an interval defined in minutes.

Syntax

```
DBMS_STATS.GATHER_PROCESSING_RATE (
   gathering_mode      IN    VARCHAR2  DEFAULT 'START',
   interval            IN    NUMBER    DEFAULT  NULL);
```

Parameters

Table 197-57 GATHER_PROCESSING_RATE Procedure Parameters

Parameter	Description
gathering_mode	Mode: 'START' or 'END'. The mode is based on the Active Session History (ASH) data when invoked with 'START' option. It stops gathering when invoked with 'END' option. When invoked with 'START', 'interval' option can be specified optionally. If interval is not specified, its default value is set to 60 minutes.
interval	Time interval (number of minutes) for which the processing must be gathered

Usage Notes

- You require the OPTIMIZER_PROCESSING_RATE role to run this procedure.
- AUTO DOP uses processing rates to determine the optimal degree of parallelism for a SQL statement.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or illegal input value

GATHER_SCHEMA_STATS Procedures

This procedure gathers statistics for all objects in a schema.

Syntax

```
DBMS_STATS.GATHER_SCHEMA_STATS (
   ownname          VARCHAR2,
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                        (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
```

```
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   objlist          OUT      ObjectTab,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
  force             BOOLEAN DEFAULT FALSE,
  obj_filter_list  ObjectTab DEFAULT NULL);

DBMS_STATS.GATHER_SCHEMA_STATS (
   ownname          VARCHAR2,
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                          (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type (
                                     get_param('NO_INVALIDATE')),
   force            BOOLEAN DEFAULT FALSE,
   obj_filter_list  ObjectTab DEFAULT NULL);
```

Parameters

Table 197-58 GATHER_SCHEMA_STATS Procedure Parameters

Parameter	Description
ownname	Schema to analyze (NULL means current schema)
estimate_percent	Determines the percentage of rows to sample. The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for optimal statistics. This is the default. You can change the default value using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Whether or not to use random block sampling instead of random row sampling. Random block sampling is more efficient, but if the data is not randomly distributed on disk, then the sample values may be somewhat correlated. Only pertinent when doing an estimate statistics.

Table 197-58 (Cont.) GATHER_SCHEMA_STATS Procedure Parameters

Parameter	Description
method_opt	Accepts: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics.

Table 197-58 (Cont.) GATHER_SCHEMA_STATS Procedure Parameters

Parameter	Description
cascade	Gather statistics on the indexes as well. Using this option is equivalent to running the GATHER_INDEX_STATS Procedure on each of the indexes in the schema in addition to gathering table and column statistics. Use the constant DBMS_STATS.AUTO_CASCADE to have Oracle determine whether index statistics to be collected or not. This is the default. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics
statid	Identifier (optional) to associate with these statistics within stattab
options	Specifies which objects require statistics to be gathered. Valid values are as follows: • GATHER — Gathers statistics on all objects in the schema. This is the default. • GATHER AUTO — Gathers all necessary statistics automatically. The database implicitly determines which objects need new statistics and determines how to gather those statistics. When GATHER AUTO is specified, the only additional valid parameters are comp_id, no_invalidate, stattab, statid, and statown; all other parameter settings will be ignored. Also, the database returns a list of objects processed. • GATHER STALE — Gathers statistics on stale objects by querying the *_TAB_MODIFICATIONS views. Also, the database returns a list of objects found to be stale. • GATHER EMPTY — Gathers statistics on objects that currently have no statistics. Also, the database returns a list of objects found to have no statistics. • LIST AUTO — Returns a list of objects to be processed with GATHER AUTO. • LIST STALE — Returns list of stale objects as determined by looking at the *_TAB_MODIFICATIONS views. • LIST EMPTY — Returns a list of objects that currently have no statistics.
objlist	List of objects found to be stale or empty
statown	Schema containing stattab (if different than ownname)
no_invalidate	Does not invalidate the dependent cursors if set to TRUE. The procedure invalidates the dependent cursors immediately if set to FALSE. Use DBMS_STATS.AUTO_INVALIDATE. to have Oracle decide when to invalidate dependent cursors. This is the default. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Gather statistics on objects even if they are locked

Table 197-58 (Cont.) GATHER_SCHEMA_STATS Procedure Parameters

Parameter	Description
obj_filter_list	A list of object filters. When provided, GATHER_SCHEMA_STATS will gather statistics only on objects which satisfy at least one object filter in the list as needed. In a single object filter, we can specify the constraints on the object attributes. The attribute values specified in the object filter are case- insensitive unless double-quoted. Wildcard is allowed in the attribute values. Suppose non-NULL values s1, s2, ... are specified for attributes a1, a2, ... in one object filter. An object o is said to satisfy this object filter if (o.a1 like s1) and (o.a2 like s2) and ... is true.

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

If the GATHER AUTO option is used then frequency histograms will be created using a sample rather than a full scan. The following scenario shows where GATHER AUTO is used:

- A table is created like this: CREATE TABLE NEWTAB as SELECT * FROM .....

  This will create statistics on NEWTAB but no histograms.
- Next, the DBA creates the histograms using GATHER AUTO on gather_table_stats.
- The FREQUENCY histograms on NEWTAB will be created using a sample rather than a full table scan.

Exceptions

ORA-20000: Schema does not exist or insufficient privileges

ORA-20001: Bad input value

Examples

Applying an Object Filter List

The following example specifies that the tables SH.SALES and SH.COSTS, if stale, will have statistics gathered upon them.

```
DECLARE
   filter_lst  DBMS_STATS.OBJECTTAB := DBMS_STATS.OBJECTTAB();
BEGIN
   filter_lst.extend(2);
   filter_lst(1).ownname := 'SH';
   filter_lst(1).objname := 'SALES';
   filter_lst(2).ownname := 'SH';
   filter_lst(2).objname := 'COSTS';
   DBMS_STATS.GATHER_SCHEMA_STATS(ownname=>'SH',obj_filter_list=>filter_lst);
 END;
```

GATHER_SYSTEM_STATS Procedure

This procedure gathers system statistics.

Syntax

```
DBMS_STATS.GATHER_SYSTEM_STATS (
   gathering_mode   VARCHAR2 DEFAULT 'NOWORKLOAD',
   interval         INTEGER  DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-59 GATHER_SYSTEM_STATS Procedure Parameters

Parameter	Description
gathering_mode	Specifies the mode in which the database gathers system statistics. Possible values are: • NOWORKLOAD The database captures performance characteristics of the I/O system. Gathering may take a few minutes and depends on the size of the database. During this period, the database estimates the average read seek time and transfer speed for the I/O system. This mode is suitable for the all workloads. To fine tune system statistics for the workload, use START and STOP or the INTERVAL option. If you gather both NOWORKLOAD and workload-specific statistics (statistics collected using INTERVAL or START and STOP), the optimizer uses the workload statistics. Collected components include cpuspeednw, ioseektim, and iotfrspeed. • INTERVAL The database captures system activity during a specified interval in minutes. This parameter works in combination with the interval parameter. The database creates or updates system statistics in the dictionary or stattab. You can use GATHER_SYSTEM_STATS (gathering_mode=>'STOP') to stop gathering earlier than scheduled. Collected components include maxthr, slavethr, cpuspeed, sreadtim, mreadtim, and mbrc. • START | STOP The database captures system activity during specified start and stop times and refreshes the dictionary or stattab with statistics for the elapsed period. The database ignores the INTERVAL value. Collected components include maxthr, slavethr, cpuspeed, sreadtim, mreadtim, and mbrc. • EXADATA In this mode, gathered system statistics take into account the unique capabilities of Oracle Exadata, such as large I/O size and high I/O throughput. The database sets multiblock read count and I/O throughput statistics along with CPU speed.
interval	Specifies the number of minutes in which to gather system statistics. This parameter applies only when gathering_mode='INTERVAL'.

Table 197-59 (Cont.) GATHER_SYSTEM_STATS Procedure Parameters

Parameter	Description
stattab	Specifies the table in which the database stores the statistics.
statid	Specifies an optional identifier associated with the statistics saved in stattab.
statown	Specifies the schema containing stattab, if different from the current schema.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid input value

ORA-20002: Bad user statistics table; may need to be upgraded

ORA-20003: Unable to gather system statistics

ORA-20004: Error in the INTERVAL mode: system parameter job_queue_processes must be >0

Usage Notes

To run this procedure, you must have the GATHER_SYSTEM_STATISTICS role.

Examples

Assume that you want to perform database application processing OLTP transactions during the day and run reports at night.

To collect daytime system statistics, gather statistics for 720 minutes. Store the statistics in the MYSTATS table.

```
BEGIN
   DBMS_STATS.GATHER_SYSTEM_STATS (
      interval => 720,
      stattab  => 'mystats',
      statid   => 'OLTP');
END;
```

To collect nighttime system statistics, gather statistics for 720 minutes. Store the statistics in the MYSTATS table.

```
BEGIN
   DBMS_STATS.GATHER_SYSTEM_STATS (
      interval => 720,
      stattab  => 'mystats',
      statid   => 'OLAP');
END;
```

Update the dictionary with the gathered statistics.

```
VARIABLE   jobno   number;
BEGIN
```

```
   DBMS_JOB.SUBMIT (:jobno, 'DBMS_STATS.IMPORT_SYSTEM_STATS
   (''mystats'',''OLTP'');'
   sysdate, 'sysdate + 1');
   COMMIT;
END;

BEGIN
   DBMS_JOB.SUBMIT (:jobno, 'DBMS_STATS.IMPORT_SYSTEM_STATS
   (''mystats'',''OLAP'');'
   sysdate + 0.5, 'sysdate + 1');
   COMMIT;
END;
```

GATHER_TABLE_STATS Procedure

This procedure gathers table, column, and index statistics. It attempts to parallelize as much work as possible, but there are some restrictions, which are described in the individual parameters.

Syntax

```
DBMS_STATS.GATHER_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                          (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT  to_no_invalidate_type (
                                            get_param('NO_INVALIDATE')),
   stattype         VARCHAR2 DEFAULT 'DATA',
   force            BOOLEAN  DEFAULT FALSE,
   context          DBMS_STATS.CCONTEXT DEFAULT NULL, -- non operative
   options          VARCHAR2 DEFAULT get_param('OPTIONS'));
```

Parameters

Table 197-60 GATHER_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	Schema containing the table.
tabname	Name of the table.
partname	Name of the partition.

Table 197-60 (Cont.) GATHER_TABLE_STATS Procedure Parameters

Parameter	Description
estimate_percent	Determines the percentage of rows to sample. The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for optimal statistics. This is the default. You can change the default value using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Determines whether the database uses random block sampling (TRUE) or random row sampling (FALSE). The default is FALSE. Random block sampling is more efficient, but if the data is not randomly distributed on disk, then sample values may be somewhat correlated. This parameter is only relevant when estimating statistics.
method_opt	METHOD_OPT - When setting preference on global, schema, database or dictionary level, only 'FOR ALL' syntax is allowed. Other than that, method_opt accepts either of the following options, or both in combination: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] • FOR COLUMNS [column_clause] [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} column_clause is defined as column_clause := column_name | extension name | extension – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. - column_name : Name of a column - extension : can be either a column group in the format of (column_name, Colume_name [, ...]) or an expression The default is FOR ALL COLUMNS SIZE AUTO. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Table 197-60 (Cont.) GATHER_TABLE_STATS Procedure Parameters

Parameter	Description
degree	Determines the degree of parallelism used for gathering statistics. The default for degree is NULL. NULL means to use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Change the default using the SET_DATABASE_PREFS, SET_GLOBAL_PREFS, SET_SCHEMA_PREFS, and SET_TABLE_PREFS procedures. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. The degree is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters), according to the size of the object. When using DEGREE=>NULL, DEGREE=>*n*, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution when the size of the object does not warrant parallel execution.
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'APPROX_GLOBAL AND PARTITION' - similar to 'GLOBAL AND PARTITION' but in this case the global statistics are aggregated from partition level statistics. This option will aggregate all statistics except the number of distinct values for columns and number of distinct keys of indexes. The existing histograms of the columns at the table level are also aggregated.The aggregation will use only partitions with statistics, so to get accurate global statistics, users should make sure to have statistics for all partitions. Global statistics are gathered if partname is NULL or if the aggregation cannot be performed (for example, if statistics for one of the partitions is missing). 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics.

Table 197-60 (Cont.) GATHER_TABLE_STATS Procedure Parameters

Parameter	Description
cascade	Determines whether to collect index statistics as part of gathering table statistics. Specifying this option is equivalent to running the GATHER_INDEX_STATS procedure on each index of the table. Use the constant DBMS_STATS.AUTO_CASCADE to enable the database to determine whether index statistics need to be collected. This is the default. You can change the default using the SET_DATABASE_PREFS, SET_GLOBAL_PREFS, SET_SCHEMA_PREFS, and SET_TABLE_PREFS procedures.
stattab	User statistics table identifier describing where to save the current statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. If set to TRUE, then the database not invalidate dependent cursors. If set to FALSE, then the procedure invalidates dependent cursors immediately.
stattype	Statistics type. The only value allowed is DATA.
force	Gather statistics of table even if it is locked
context	Not used.
options	Determines the options parameter used in the GATHER_TABLE_STATS procedure. The preference takes the following values: • GATHER — Gathers statistics for all objects in the table. This is the default. • GATHER AUTO — Oracle recommends using GATHER AUTO to gather necessary statistics, such as histograms, after a table has been bulk loaded and acquired online statistics. This is applicable only to tables that are not using INCREMENTAL statistics. For partitioned tables using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO will gather statistics for a table if it is marked stale or has no statistics. In addition, statistics will be gathered for partitions and sub-partitions that are marked stale or have no statistics. For tables not using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO may gather statistics for partitions and sub-partitions, even if they are not marked stale.

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Index statistics collection can be parellelized except for cluster, domain and join indexes.

An explicit call to `dbms_stats.gather_table_stats` is needed for proper bookkeeping of the keys of a local Index.

If the `GATHER AUTO` option is used then frequency histograms will be created using a sample rather than a full scan. The following scenario shows where `GATHER AUTO` is used:

- A table is created like this: `CREATE TABLE NEWTAB as SELECT * FROM .....`

  This will create statistics on `NEWTAB` but no histograms.
- Next, the DBA creates the histograms using `GATHER AUTO` on `gather_table_stats`.
- The `FREQUENCY` histograms on `NEWTAB` will be created using a sample rather than a full table scan.

Exceptions

`ORA-20000`: Table does not exist or insufficient privileges

`ORA-20001`: Bad input value

Examples

An extension can be either a column group (see Example 1) or an expression (see Example 2).

Example 1

```
DBMS_STATS.GATHER_TABLE_STATS(
    'SH', 'SALES', method_opt => 'FOR COLUMNS (empno, deptno)');
```

Example 2

```
DBMS_STATS.GATHER_TABLE_STATS(
    'SH', 'SALES', method_opt => 'FOR COLUMNS (sal+comm)');
```

GENERATE_STATS Procedure

This **deprecated procedure** generates object statistics from previously collected statistics of related objects. The currently supported objects are b-tree and bitmap indexes.

> **Note:**
>
> This subprogram has been deprecated and replaced by improved technology. It is maintained only for purposes of backward compatibility. As an alternative, use the `GATHER_INDEX_STAT` procedure. See "GATHER_INDEX_STATS Procedure".

Syntax

```
DBMS_STATS.GENERATE_STATS (
   ownname     VARCHAR2,
   objname     VARCHAR2,
   organized   NUMBER DEFAULT 7,
   force       BOOLEAN default FALSE);
```

Parameters

Table 197-61 GENERATE_STATS Procedure Parameters

Parameter	Description
ownname	Schema of object
objname	Name of object
organized	Amount of ordering associated between the index and its underlying table. A heavily organized index would have consecutive index keys referring to consecutive rows on disk for the table (the same block). A heavily disorganized index would have consecutive keys referencing different table blocks on disk. This parameter is only used for b-tree indexes. The number can be in the range of 0-10, with 0 representing a completely organized index and 10 a completely disorganized one.
force	If TRUE, generates statistics for the target object even if it is locked

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

For fully populated schemas, the gather procedures should be used instead when more accurate statistics are desired.

Exceptions

ORA-20000: Unsupported object type of object does not exist

ORA-20001: Invalid option or invalid statistics

GET_ADVISOR_OPR_FILTER Procedure

This procedure creates an operation filter for an Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.GET_ADVISOR_OPR_FILTER (
  opr_id        IN             NUMBER,
  opr_filter    IN OUT NOCOPY  StatsAdvOpr);
```

Parameters

Table 197-62 GET_ADVISOR_OPR_FILTER Procedure Parameters

Parameter	Description
opr_id	The ID of the statistics operation stored in the DBA_OPTSTAT_OPERATIONS view.
opr_filter	The Optimizer Statistics Advisor filter that is generated based on the specified statistics operation.

Security Model

Note the following:

- To execute this subprogram, you must have the `ADVISOR` privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Exceptions

- `ORA-20000`: Insufficient privileges
- `ORA-20001`: Invalid input values
- `ORA-20012`: Optimizer Statistics Advisor errors

Usage Notes

You can specify the filter using either the operation ID or the filter ID, but not both at the same time.

GET_ADVISOR_RECS Function

This function generates a recommendation report for the specified item.

Syntax

```
DBMS_STATS.GET_ADVISOR_RECS(
  ownname     IN   VARCHAR2,
  tabname     IN   VARCHAR2,
  rec         IN   VARCHAR2,
  type        IN   VARCHAR2    := 'TEXT')
RETURN CLOB;
```

Parameters

Table 197-63 GET_ADVISOR_RECS Function Parameters

Parameter	Description
`ownname`	The owner of the table.
`tabname`	The name of the table.

Table 197-63 (Cont.) GET_ADVISOR_RECS Function Parameters

Parameter	Description
rec	The Optimizer Statistics Advisor recommendation. • INCREMENTAL When only a small number of range partitions are modified, this option improves the performance of statistics gathering dramatically. However, it requires additional space to store synopses for maintaining incremental statistics. The report analyzes this trade-off. • CONCURRENT The report recommends either setting the CONCURRENT preference, or specifying AUTO_DEGREE for individual tables. If the system resources and usage satisfies the conditions, the advisor always recommends setting CONCURRENT first. The advisor only recommends AUTO_DEGREE when statistics gathering on an individual table take a long time and the CONCURRENT preference is already set.
type	Type of the report: TEXT, HTML, or XML.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must have the privileges to gather statistics for the objects for which recommendations are generated.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Usage Notes

The advisor does not make recommendations for manual statistics gathering. The database only make recommendations for automatic statistics gathering jobs, with the main goal of finishing the job within the maintenance window. As long as the automatic job finishes, the database does not make further recommendations.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

GET_COLUMN_STATS Procedures

These overloaded procedures get column-related statistics. In the user-defined statistics version, the procedure returns the type of statistics stored.

Syntax

```
DBMS_STATS.GET_COLUMN_STATS (
   ownname         VARCHAR2,
```

```
   tabname        VARCHAR2,
   colname        VARCHAR2,
   partname       VARCHAR2 DEFAULT NULL,
   stattab        VARCHAR2 DEFAULT NULL,
   statid         VARCHAR2 DEFAULT NULL,
   distcnt OUT    NUMBER,
   density OUT    NUMBER,
   nullcnt OUT    NUMBER,
   srec    OUT    StatRec,
   avgclen OUT    NUMBER,
   statown        VARCHAR2 DEFAULT NULL,
   realtime_stats BOOLEAN iDEFAULT TRUE);
```

Use the following for user-defined statistics:

```
DBMS_STATS.GET_COLUMN_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   colname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   ext_stats   OUT  RAW,
   stattypown  OUT  VARCHAR2 DEFAULT NULL,
   stattypname OUT  VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-64 GET_COLUMN_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
tabname	Specifies the name of the table to which this column belongs.
colname	Specifies the name of the column or extension.
partname	Specifies the name of the table partition from which to get the statistics. If the table is partitioned, and if partname is NULL, then the procedure retrieves statistics at the global table level.
stattab	Specifies the statistics table ID describing where to retrieve the statistics. If stattab is NULL, then the procedure retrieves statistics directly from the data dictionary.
statid	Specifies an optional identifier associated with these statistics within stattab. This parameter is only relevant when stattab is not NULL.
ext_stats	Specifies the user-defined statistics.
stattypown	Specifies the schema of the statistics type.
stattypname	Specifies the name of the statistics type.
distcnt	Specifies the number of distinct values.
density	Specifies the column density.
nullcnt	Specifies the number of NULL values.

Table 197-64 (Cont.) GET_COLUMN_STATS Procedure Parameters

Parameter	Description
srec	Specifies the structure holding the internal representation of the column minimum, maximum, and histogram values.
avgclen	Specifies the average length of the column (in bytes).
statown	Specifies the schema containing stattab (if different than ownname).
realtime_stats	Specifies whether to include real-time statistics. The default value is TRUE. When realtime_stats is FALSE, the database only includes optimizer statistics that were gathered by the GATHER_*_STATS procedures.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be the owner of the table, or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges or no statistics have been stored for requested object

Usage Notes

Before invoking this procedure, ensure that the table exists.

GET_INDEX_STATS Procedures

This overloaded procedure gets all index-related statistics. In the form of this procedure that deals with user-defined statistics, the statistics type returned is the type stored, in addition to the user-defined statistics.

Syntax

```
DBMS_STATS.GET_INDEX_STATS (
   ownname         VARCHAR2,
   indname         VARCHAR2,
   partname        VARCHAR2 DEFAULT NULL,
   stattab         VARCHAR2 DEFAULT NULL,
   statid          VARCHAR2 DEFAULT NULL,
   numrows     OUT NUMBER,
   numlblks    OUT NUMBER,
   numdist     OUT NUMBER,
   avglblk     OUT NUMBER,
   avgdblk     OUT NUMBER,
   clstfct     OUT NUMBER,
   indlevel    OUT NUMBER,
   statown         VARCHAR2 DEFAULT NULL);

DBMS_STATS.GET_INDEX_STATS (
   ownname         VARCHAR2,
   indname         VARCHAR2,
   partname        VARCHAR2 DEFAULT NULL,
```

```
   stattab        VARCHAR2 DEFAULT NULL,
   statid         VARCHAR2 DEFAULT NULL,
   numrows    OUT NUMBER,
   numlblks   OUT NUMBER,
   numdist    OUT NUMBER,
   avglblk    OUT NUMBER,
   avgdblk    OUT NUMBER,
   clstfct    OUT NUMBER,
   indlevel   OUT NUMBER,
   statown        VARCHAR2 DEFAULT NULL,
   guessq     OUT NUMBER);
```

Use the following form of the procedure for user-defined statistics:

```
DBMS_STATS.GET_INDEX_STATS (
   ownname          VARCHAR2,
   indname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   ext_stats   OUT  RAW,
   stattypown  OUT  VARCHAR2 DEFAULT NULL,
   stattypname OUT  VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-65 GET_INDEX_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
indname	Name of the index
partname	Name of the index partition for which to get the statistics. If the index is partitioned and if partname is NULL, then the statistics are retrieved for the global index level.
stattab	User statistics table identifier describing from where to retrieve the statistics. If stattab is NULL, then the statistics are retrieved directly from the dictionary.
statid	Identifier (optional) to associate with these statistics within stattab (Only pertinent if stattab is not NULL)
ext_stats	User-defined statistics
stattypown	Schema of the statistics type
stattypname	Name of the statistics type
numrows	Number of rows in the index (partition)
numlblks	Number of leaf blocks in the index (partition)
numdist	Number of distinct keys in the index (partition)
avglblk	Average integral number of leaf blocks in which each distinct key appears for this index (partition)
avgdblk	Average integral number of data blocks in the table pointed to by a distinct key for this index (partition)

Table 197-65 (Cont.) GET_INDEX_STATS Procedure Parameters

Parameter	Description
clstfct	Clustering factor for the index (partition)
indlevel	Height of the index (partition)
statown	Schema containing stattab (if different than ownname)
guessq	Guess quality for the index (partition)

Security Model

Before invoking this procedure, ensure that the table exists. To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges or no statistics have been stored for requested object

Usage Notes

- The optimizer uses the cached data to estimate number of cached blocks for index or statistics table access. The database calculates the total cost of the operation by combining the I/O cost of reading not cached blocks from disk, the CPU cost of getting cached blocks from the buffer cache, and the CPU cost of processing the data.
- The database maintains cachedblk and cachehit at all times. However, the database uses the corresponding caching statistics for optimization as part of the table and index statistics only when the user calls the DBMS_STATS.GATHER_[TABLE/INDEX/SCHEMA/DATABASE]_STATS procedure for automatic mode or DBMS_STATS.GATHER_SYSTEM_STATS for manual mode. To prevent the user from utilizing inaccurate and unreliable data, the optimizer computes a “confidence factor” for each cachehit and a cachedblk for each object. If the confidence factor for the value meets confidence criteria, then the database uses this value; otherwise, the database uses defaults.
- The automatic maintenance algorithm for object caching statistics assumes that only one major database workload exists. The algorithm adjusts statistics to this workload, ignoring other "minor" workloads. If this assumption is false, then you must use manual mode for maintaining object caching statistics.
- The object caching statistics maintenance algorithm for automatic mode prevents you from using statistics in the following situations:
  - When not enough data has been analyzed, such as when an object has been recently created
  - When the system does not have one major workload resulting in averages not corresponding to real values

See Also:

Oracle Database SQL Tuning Guide to learn how to manage optimizer statistics

GET_PARAM Function

This function returns the default value of parameters of `DBMS_STATS` procedures.

> **Note:**
>
> This subprogram has been replaced by improved technology and is maintained only for purposes of backward compatibility. In this case, use the GET_PREFS Function.
>
> See also DBMS_STATS Deprecated Subprograms.

Syntax

```
DBMS_STATS.GET_PARAM (
   pname     IN   VARCHAR2)
 RETURN VARCHAR2;
```

Parameters

Table 197-66 GET_PARAM Function Parameters

Parameter	Description
`pname`	Parameter name

Exceptions

`ORA-20001`: Invalid input values

GET_PLSQL_PREFS Procedure

Returns the preference used for the named function. If a function has no preference set, the procedure returns the global preference.

Syntax

```
DBMS_STATS.GET_PLSQL_PREFS(
      pname           IN VARCHAR2,
      ownname         IN VARCHAR2,
      package_name    IN VARCHAR2,
      function_name   IN VARCHAR2)
RETURN VARCHAR2;
```

You can get the global preference by passing in NULL for `user`, `package_name`, and `function_name`.

Parameters

Table 197-67 GET_PLSQL_PREFS Parameters

Parameter	Description
pname	The preference that you want to get. Currently only 'dynamic_stats' is accepted.
user	The owner of the function. If NULL, the current user is used.
package_name	The name of the PL/SQL package. If NULL, this specifies a standalone, top-level function.
function_name	The name of the PL/SQL function.

Example

Get the preference value that is set for an individual PL/SQL function within a package. The parameter `user` is NULL in order to specify the current user:

```
SELECT DBMS_STATS.GET_PLSQL_PREFS('dynamic_stats',NULL,'<PL/SQL_PACKAGE
NAME>','<PL/SQL FUNCTION
      NAME>') pref_value FROM dual;
```

GET_PREFS Function

This function returns the default value of the specified preference.

Syntax

```
DBMS_STATS.GET_PREFS (
   pname     IN   VARCHAR2,
   ownname   IN   VARCHAR2 DEFAULT NULL,
   tabname   IN   VARCHAR2 DEFAULT NULL)
 RETURN VARCHAR2;
```

Parameters

Table 197-68 GET_PREFS Function Parameters

Parameter	Description
pname	Preference name. The possible values are: • APPROXIMATE_NDV_ALGORITHM • AUTO_STAT_EXTENSIONS • AUTO_TASK_STATUS • AUTO_TASK_MAX_RUN_TIME • AUTO_TASK_INTERVAL • CASCADE • CONCURRENT • DEGREE • ESTIMATE_PERCENT • GLOBAL_TEMP_TABLE_STATS • GRANULARITY • INCREMENTAL • INCREMENTAL_STALENESS • INCREMENTAL_LEVEL • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • STAT_CATEGORY • TABLE_CACHED_BLOCKS • WAIT_TIME_TO_UPDATE_STATUS
ownname	Owner name
tabname	Table name

Table 197-69 Preference Descriptions

Preference Name	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. Consider a synopsis as an internal management structure that samples distinct values. You can specify the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
AUTO_TASK_STATUS	Enables or disables the high-frequency automatic optimizer statistics collection. Values are: • ON — Enables high-frequency automatic optimizer statistics collection. • OFF — Disables high-frequency automatic optimizer statistics collection. This is the default.

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description
AUTO_TASK_MAX_RUN_TIME	Configures the maximum run time in seconds of an execution of high-frequency automatic optimizer statistics collection. The maximum value is 86400 (equal to 24 hour), which is the default.
AUTO_TASK_INTERVAL	Specifies the interval in seconds between executions of high-frequency automatic optimizer statistics collection. The default value is 3600 but Oracle sets it differently for some platforms and Oracle Database releases.
CASCADE	Determines whether index statistics are collected as part of gathering table statistics.
CONCURRENT	Determines whether statistics are gathered concurrently on multiple objects, or serially, one object at a time. Valid values are: • MANUAL — Concurrency is enabled only for manual statistics gathering. • AUTOMATIC — Concurrency is enabled only for the automatic statistics gathering. • ALL — Concurrency is enabled for both manual and automatic statistics gathering. • OFF — Concurrency is disabled for both manual and automatic statistics.
DEGREE	Determines degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	Determines the percentage of rows to sample. The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for optimal statistics. This is the default.
GLOBAL_TEMP_TABLE_STATS	Controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics. This preference takes two values: • SHARED — All sessions see the same set of statistics • SESSION — Statistics gathered by the GATHER_TABLE_STATS procedure on a global temporary table are session-specific. Thus, the database only uses them for queries issued in the same session as the statistics gathering process. The database deletes session-specific statistics when a session terminates.

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description
GRANULARITY	Determines the granularity of statistics to collect. This preference is only relevant for partitioned tables. The following values are valid: • ALL — Gathers all statistics: subpartition, partition, and global. • AUTO — Determines the granularity based on the partitioning type. This is the default value. • DEFAULT — Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. Use GLOBAL AND PARTITION for this functionality. • GLOBAL — Gathers global statistics. • GLOBAL AND PARTITION — Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. • PARTITION — Gathers partition-level statistics. • SUBPARTITION — Gathers subpartition-level statistics.
INCREMENTAL	Determines whether the global statistics for a partitioned table are maintained without performing a full table scan. When a table is partitioned, an application typically loads data into a new partition. As new partitions are added and data is loaded, global table statistics must be kept up to date. If the following conditions are met, then the database updates the global table statistics by scanning only the changed partitions instead of the entire table: • The INCREMENTAL value for the partitioned table is set to TRUE. • The PUBLISH value for the partitioned table is set to TRUE. • The user specifies AUTO_SAMPLE_SIZE for ESTIMATE_PERCENT and AUTO for GRANULARITY when gathering statistics on the table. If the INCREMENTAL value for the partitioned table was set to FALSE (default value), then the database uses a full table scan to maintain the global statistics. This technique is a much more resource-intensive and time-consuming operation for large tables.

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description
INCREMENTAL_LEVEL	Controls which synopses to collect when INCREMENTAL preference is set to TRUE. It takes the following values: • PARTITION — Gathers partition-level synopses. This is the default value. If PARTITION is set on a nonpartitioned table, then the database does not gather synopses. • TABLE — Gathers table-level synopses. Specify this value when you want to exchange this table with a partition. Before the exchange, you can run GATHER_TABLE_STATS on this table with INCREMENTAL set to TRUE and INCREMENTAL_LEVEL to TABLE. The result is that the database gathers table-level synopses on this table. After the exchange, the partition has synopses that come from the table-level synopses of the table before the exchange. You can only use preference value in the SET_TABLE_PREFS procedure: this value is not allowed in the other SET_*_PREFS procedures.

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description

INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'.
	The parameter accepts the following values:
	• USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference.
	For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics.
	Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics.
	• USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes.
	For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics.
	• ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist.
	• NULL—A partition or subpartition is considered stale when it has any DML changes.
	For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.
	Note that the following two executions are different:
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	The first execution uses single quotes to set the preference to the value NULL, whereas the second sets

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description
	the preference to the default, which is `ALLOW_MIXED_FORMAT`.
`METHOD_OPT`	Controls column statistics collection and histogram creation. When setting preferences at the global, schema, database, or dictionary level, only `FOR ALL` syntax is allowed: • `FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause]` The `size_clause` is defined as `size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY}` `integer` — Specifies the number of histogram buckets. The number must be between 1 and 2048. `REPEAT` — Collects histograms only on the columns that already have histograms. `AUTO` — Determines the columns on which to collect histograms based on data distribution and the workload of the columns. `SKEWONLY` — Determines the columns on which to collect histograms based on the data distribution of the columns. The default is `FOR ALL COLUMNS SIZE AUTO`. You can change the value using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
`NO_INVALIDATE`	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • `TRUE`: Dependent cursors are not invalidated. • `FALSE`: Dependent cursors are marked for immediate invalidation. • `AUTO`: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. If set to `TRUE`, then the database not invalidate dependent cursors. If set to `FALSE`, then the procedure invalidates dependent cursors immediately.

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS procedure. The preference takes the following values: • GATHER — Gathers statistics for all objects in the table. This is the default. • GATHER AUTO — Oracle recommends using GATHER AUTO to gather necessary statistics, such as histograms, after a table has been bulk loaded and acquired online statistics. This is applicable only to tables that are not using INCREMENTAL statistics. For partitioned tables using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO will gather statistics for a table if it is marked stale or has no statistics. In addition, statistics will be gathered for partitions and sub-partitions that are marked stale or have no statistics. For tables not using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO may gather statistics for partitions and sub-partitions, even if they are not marked stale.
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether the database publishes newly gathered statistics after the gathering job completes. You can gather statistics without publishing them immediately. This technique enables you to test new statistics before publishing them.
STALE_PERCENT	Determines the percentage of rows in a table that must change before the statistics on that table are stale and need to be regathered. The valid domain for stale_percent is non-negative numbers. The default value is 10, which means that a table having more than 10% of changes is considered stale.

Table 197-69 (Cont.) Preference Descriptions

Preference Name	Description
STAT_CATEGORY	Specifies which statistics to import or export, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS - table statistics, column statistics and index statistics (default) • SYNOPSES - information to support incremental statistics • REALTIME_STATS — specifies only real-time statistics. The value 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.
WAIT_TIME_TO_UPDATE_STATS	Specifies the number of minutes before timing out for locks and pins required for updating statistics. It accepts values in the range 0 to 65535. The default value is 0.0033 minutes. The value 0 gets the locks and pins in no-wait mode.

Security Model

No special privilege or role is needed to invoke this procedure. To gather statistics concurrently, however, you must either have the DBA role, or have the following privileges in addition to privileges that are required for gathering statistics: CREATE JOB, MANAGE SCHEDULER, and MANAGE ANY QUEUE.

Exceptions

- ORA-20000: Unable to gather statistics concurrently: Resource Manager is not enabled.
- ORA-20001: Invalid input values

Usage Notes

Note the following guidelines:

- The CONCURRENT preference determines whether statistics are gathered concurrently when the user issues GATHER_*_STATS procedures. DBMS_STATS can collect statistics for a single object in parallel based on the value of the DEGREE parameter. However, parallelism is limited to one object. The CONCURRENT preference extends the scope of parallelism to multiple database objects. This approach is primarily intended for multi-CPU systems, and may not be suitable for small databases on single-CPU computers.

  To gather statistics concurrently, Resource Manager must be enabled, and the setting for the JOB_QUEUE_PROCESSES initialization parameter must be at least 4.

- If the ownname and tabname are provided, and if a preference has been entered for the table, then the function returns the preference as specified for the table. In all other cases,

it returns the global preference if it has been specified, otherwise it returns the default value.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to get optimizer statistics preferences

GET_STATS_HISTORY_AVAILABILITY Function

This function returns oldest timestamp where statistics history is available. Users cannot restore statistics to a timestamp older than this one.

Syntax

```
DBMS_STATS.GET_STATS_HISTORY_AVAILABILITY
 RETURN TIMESTAMP WITH TIMEZONE;
```

Usage Notes

No special privilege or role is needed to invoke this procedure.

GET_STATS_HISTORY_RETENTION Function

This function returns the current statistics history retention value.

Syntax

```
DBMS_STATS.GET_STATS_HISTORY_RETENTION
 RETURN NUMBER;
```

Usage Notes

No special privilege or role is needed to invoke this procedure.

GET_SYSTEM_STATS Procedure

This procedure gets system statistics from `stattab`, or from the dictionary if `stattab` is `NULL`.

Syntax

```
DBMS_STATS.GET_SYSTEM_STATS (
   status    OUT  VARCHAR2,
   dstart    OUT  DATE,
   dstop     OUT  DATE,
   pname     IN   VARCHAR2,
   pvalue    OUT  NUMBER,
   stattab   IN   VARCHAR2 DEFAULT NULL,
   statid    IN   VARCHAR2 DEFAULT NULL,
   statown   IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-70 GET_SYSTEM_STATS Procedure Parameters

Parameter	Description
status	Output is one of the following: • COMPLETED: • AUTOGATHERING: • MANUALGATHERING: • BADSTATS:
dstart	Date when statistics gathering started. If status = MANUALGATHERING, the start date is returned.
dstop	Date when statistics gathering stopped. • If status = COMPLETE, the finish date is returned. • If status = AUTOGATHERING, the future finish date is returned. • If status = BADSTATS, the must-finished-by date is returned.
pname	The parameter name to get, which can have one of the following values: • iotfrspeed - I/O transfer speed in bytes for each millisecond • ioseektim - seek time + latency time + operating system overhead time, in milliseconds • sreadtim - average time to read single block (random read), in milliseconds • mreadtim - average time to read an mbrc block at once (sequential read), in milliseconds • cpuspeed - average number of CPU cycles for each second, in millions, captured for the workload (statistics collected using 'INTERVAL' or 'START' and 'STOP' options) • cpuspeednw - average number of CPU cycles for each second, in millions, captured for the no-workload (statistics collected using 'NOWORKLOAD' option. • mbrc - average multiblock read count for sequential read, in blocks • maxthr - maximum I/O system throughput, in bytes/second • slavethr - average slave I/O throughput, in bytes/second
pvalue	Parameter value to get
stattab	Identifier of the user statistics table where the statistics will be obtained. If stattab is NULL, the statistics will be obtained from the dictionary.
statid	Optional identifier associated with the statistics saved in the stattab
statown	Schema containing stattab (if different from current schema)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20002: Bad user statistics table; may need to be upgraded

ORA-20003: Unable to gather system statistics

ORA-20004: Parameter does not exist

Usage Notes

To run this procedure, you need the GATHER_SYSTEM_STATISTICS role.

GET_TABLE_STATS Procedure

This overloaded procedure gets all table-related statistics.

Syntax

```
DBMS_STATS.GET_TABLE_STATS (
   ownname            VARCHAR2,
   tabname            VARCHAR2,
   partname           VARCHAR2 DEFAULT NULL,
   stattab            VARCHAR2 DEFAULT NULL,
   statid             VARCHAR2 DEFAULT NULL,
   numrows        OUT NUMBER,
   numblks        OUT NUMBER,
   avgrlen        OUT NUMBER,
   statown            VARCHAR2 DEFAULT NULL,
   realtime_stats     BOOLEAN DEFAULT TRUE);


DBMS_STATS.GET_TABLE_STATS (
   ownname            VARCHAR2,
   tabname            VARCHAR2,
   partname           VARCHAR2 DEFAULT NULL,
   stattab            VARCHAR2 DEFAULT NULL,
   statid             VARCHAR2 DEFAULT NULL,
   numrows        OUT NUMBER,
   numblks        OUT NUMBER,
   avgrlen        OUT NUMBER,
   statown            VARCHAR2 DEFAULT NULL,
   im_imcu_count  OUT NUMBER,
   im_block_count OUT NUMBER,
   scanrate       OUT NUMBER,
   realtime_stats     BOOLEAN DEFAULT TRUE);


DBMS_STATS.GET_TABLE_STATS (
   ownname           VARCHAR2,
   tabname           VARCHAR2,
   partname          VARCHAR2 DEFAULT NULL,
   stattab           VARCHAR2 DEFAULT NULL,
   statid            VARCHAR2 DEFAULT NULL,
   numrows       OUT NUMBER,
   numblks       OUT NUMBER,
   avgrlen       OUT NUMBER,
   cachedblk     OUT NUMBER,
   cachehit      OUT NUMBER,
   realtime_stats     BOOLEAN DEFAULT TRUE);
```

Parameters

Table 197-71 GET_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	Specifies the name of the schema.
tabname	Specifies the name of the table to which this column belongs.
partname	Specifies the name of the table partition from which to get the statistics. If the table is partitioned and if partname is NULL, then the statistics are retrieved from the global table level.
stattab	Specifies the user statistics table ID. This ID describes where to retrieve the statistics. If stattab is NULL, then the procedure gathers statistics directly from the data dictionary.
statid	Specifies the optional ID associates with these statistics within stattab. This ID is only relevant when stattab is not NULL.
numrows	Specifies the number of rows in the table or partition.
numblks	Specifies the number of blocks in the table or partition.
avgrlen	Specifies the average row length for the table or partition.
statown	Specifies the schema containing stattab (if different from ownname).
im_imcu_count	Specifies the number of In-Memory Compression Units (IMCUs) in the table or partition.
im_block_count	Specifies the number of In-Memory blocks in the table or partition. An In-Memory block corresponds to a specific data block on disk. If the table is fully populated in the IM column store, then the number of In-Memory blocks equals the number of data blocks.
scanrate	Specifies the rate, in MB/s, at which the database scans external tables. This parameter is relevant only for external tables.
realtime_stats	Specifies whether to include real-time statistics. The default value is TRUE. When realtime_stats is FALSE, the database only includes optimizer statistics that were gathered by the GATHER_*_STATS procedures.
cachedblk	For internal use only.
cachehit	For internal use only.

Security Model

Before invoking this procedure, ensure that the table exists. To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table, or have the ANALYZE ANY DICTIONARY or SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges or no statistics have been stored for requested object

ORA-20002: Bad user statistics table; may need to upgrade it

Usage Notes

- The optimizer uses the cached data to estimate number of cached blocks for index or statistics table access. The database calculates the total cost of the operation by combining the I/O cost of reading not cached blocks from disk, the CPU cost of getting cached blocks from the buffer cache, and the CPU cost of processing the data.
- The database maintains `cachedblk` and `cachehit` at all times. However, the database uses the corresponding caching statistics for optimization as part of the table and index statistics only when the user calls the `DBMS_STATS.GATHER_*_STATS` procedure for automatic mode or `DBMS_STATS.GATHER_SYSTEM_STATS` for manual mode. To prevent the user from utilizing inaccurate and unreliable data, the optimizer computes a “confidence factor” for each `cachehit` and a `cachedblk` for each object. If the confidence factor for the value meets confidence criteria, then the database uses this value; otherwise, the database uses defaults.
- The automatic maintenance algorithm for object caching statistics assumes that only one major database workload exists. The algorithm adjusts statistics to this workload, ignoring other "minor" workloads. If this assumption is false, then you must use manual mode for maintaining object caching statistics.
- The object caching statistics maintenance algorithm for automatic mode prevents you from using statistics in the following situations
  - When not enough data has been analyzed, such as when an object has been recently created
  - When the system does not have one major workload resulting in averages not corresponding to real values
- The database does not support export or import of statistics across databases of different character sets.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to manage optimizer statistics preferences

IMPLEMENT_ADVISOR_TASK Function

This function implements the recommendations made by Optimizer Statistics Advisor.

Syntax

```
DBMS_STATS.IMPLEMENT_ADVISOR_TASK (
  task_name          IN   VARCHAR2,
  execution_name     IN   VARCHAR2    := NULL,
  level              IN   VARCHAR2    := 'TYPICAL')
RETURN CLOB;
```

Parameters

Table 197-72 IMPLEMENT_ADVISOR_TASK Function Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.
execution_name	A name that qualifies and identifies an advisor execution. If not specified, then the advisor automatically generates it. If the specified execution conflicts with the name of an existing execution, then the function returns an error.
level	The level of the implementation. Possible values are • ALL: Ignores the filters and implements all recommendations. • TYPICAL: Implements the recommendations according to the filters in place.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- You can execute this subprogram for AUTO_STATS_ADVISOR_TASK, which is predefined.
- This subprogram executes using invoker's rights.

The results of performing this task depend on the privileges of the executing user:

- SYSTEM level

  Only users with both the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges can perform this task on system-level rules.
- Operation level

  The results depend on the following privileges:

  – Users with both the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges can perform this task for all statistics operations.

  – Users with the ANALYZE ANY privilege but *not* the ANALYZE ANY DICTIONARY privilege can perform this task for statistics operations related to any schema except SYS.

  – Users with the ANALYZE ANY DICTIONARY privilege but *not* the ANALYZE ANY privilege can perform this task for statistics operations related to their own schema and the SYS schema.

  – Users with neither the ANALYZE ANY nor the ANALYZE ANY DICTIONARY privilege can only perform this operation for statistics operations relating to their own schema.
- Object level

  Users can perform this task for any object for which they have statistics collection privileges.

Return Values

This function returns an XML CLOB that indicates which recommendations were successfully implemented.

Exceptions

- `ORA-20000`: Insufficient privileges
- `ORA-20001`: Invalid input values
- `ORA-20012`: Optimizer Statistics Advisor errors

Example 197-10 Implementing Optimizer Statistics Advisor Recommendations

This script illustrates a basic Optimizer Statistics Advisor session. It creates a task, executes it, generates a report, and then implements the recommendations.

```
DECLARE
  v_tname      VARCHAR2(128) := 'my_task';
  v_ename      VARCHAR2(128) := NULL;
  v_report     CLOB := null;
  v_script     CLOB := null;
  v_imp_result CLOB;
BEGIN
  -- create a task
  v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);

  -- execute the task
  v_ename := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);

  -- view the task report
  v_report := DBMS_STATS.REPORT_ADVISOR_TASK(v_tname);
  DBMS_OUTPUT.PUT_LINE(v_report);

  -- implement all recommendations
  v_imp_result := DBMS_STATS.IMPLEMENT_ADVISOR_TASK(v_tname);
END;
```

IMPORT_COLUMN_STATS Procedure

This procedure retrieves statistics for a particular column from the user statistics table identified by `stattab` and stores them in the dictionary.

Syntax

```
DBMS_STATS.IMPORT_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT to_no_invalidate_type (
                                  get_param('NO_INVALIDATE')),
   force         BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-73 IMPORT_COLUMN_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
tabname	Name of the table to which this column belongs
colname	Name of the column or extension
partname	Name of the table partition. If the table is partitioned and if partname is NULL, then global and partition column statistics are imported.
stattab	User statistics table identifier describing from where to retrieve the statistics
statid	Identifier to associate with these statistics within stattab (optional)
statown	Schema containing stattab (if different than ownname)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	If set to TRUE, imports statistics even if statistics are locked

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

ORA-20005: Object statistics are locked

Usage Notes

Oracle does not support export or import of statistics across databases of different character sets.

IMPORT_DATABASE_PREFS Procedure

This procedure is used to import the statistics preferences of all the tables, excluding the tables owned by Oracle. These tables can by included by passing `TRUE` for the `add_sys` parameter.

Syntax

```
DBMS_STATS.IMPORT_DATABASE_PREFS (
    stattab    IN  VARCHAR2,
    statid     IN  VARCHAR2 DEFAULT NULL,
    statown    IN  VARCHAR2 DEFAULT NULL
    add_sys    IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-74 IMPORT_DATABASE_PREFS Procedure Parameters

Parameter	Description
`stattab`	Statistics table name where to import the statistics
`statid`	Optional identifier to associate with these statistics within `stattab`
`statown`	Schema containing `stattab` (if different than `ownname`)
`add_sys`	Value `TRUE` will include the Oracle-owned tables

Exceptions

`ORA-20000`: Insufficient privileges.

Usage Notes

- To run this procedure, you need to have the SYSDBA role, or both `ANALYZE ANY DICTIONARY` and `ANALYZE ANY` system privileges.
- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.IMPORT_DATABASE_PREFS('STATTAB', statown=>'SH');
```

IMPORT_DATABASE_STATS Procedure

This procedure imports statistics for all objects in the database from the user statistics table and stores them in the data dictionary.

Syntax

```
DBMS_STATS.IMPORT_DATABASE_STATS (
   stattab         VARCHAR2,
   statid          VARCHAR2 DEFAULT NULL,
   statown         VARCHAR2 DEFAULT NULL,
   no_invalidate   BOOLEAN DEFAULT to_no_invalidate_type(
                                   get_param('NO_INVALIDATE')),
```

```
force           BOOLEAN DEFAULT FALSE,
stat_category   VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-75 IMPORT_DATABASE_STATS Procedure Parameters

Parameter	Description
stattab	Specifies the statistics table that contains the statistics to be imported.
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different from current schema)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Overrides statistics locked at the object (table) level: • TRUE - Ignores the statistics lock and imports the statistics • FALSE - The statistics will be imported only if they are not locked
stat_category	Specifies what statistics to import, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics • SYNOPSES — information to support incremental statistics • REALTIME_STATS — specifies only real-time statistics. The default value is 'OBJECT_STATS, SYNOPSES, MODELS'.

Security Model

You must have either the SYSDBA privilege or both the ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

IMPORT_DICTIONARY_STATS Procedure

This procedure imports statistics for all data dictionary schemas (SYS, SYSTEM, and RDBMS component schemas) from the user statistics table and stores them in the dictionary.

Syntax

```
DBMS_STATS.IMPORT_DICTIONARY_STATS (
   stattab         VARCHAR2,
   statid          VARCHAR2 DEFAULT NULL,
   statown         VARCHAR2 DEFAULT NULL,
   no_invalidate   BOOLEAN DEFAULT to_no_invalidate_type(
                                 get_param('NO_INVALIDATE')),
   force           BOOLEAN DEFAULT FALSE,
   stat_category   VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-76 IMPORT_DICTIONARY_STATS Procedure Parameters

Parameter	Description
stattab	User statistics table identifier describing from where to retrieve the statistics
statid	The (optional) identifier to associate with these statistics within stattab
statown	Schema containing stattab (if different from current schema)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Overrides statistics lock at the object (table) level: • TRUE - Ignores the statistics lock and imports the statistics. • FALSE - The statistics will be imported only if there is no lock.
stat_category	Specifies what statistics to import, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics • SYNOPSES — information to support incremental statistics • REALTIME_STATS — Specifies only real-time statistics. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS'.

Security Model

You must have either the SYSDBA privilege or both the ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

ORA-20002: Bad user statistics table, may need to upgrade it

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

IMPORT_FIXED_OBJECTS_STATS Procedure

This procedure imports statistics for fixed tables from the user statistics table and stores them in the data dictionary.

Syntax

```
DBMS_STATS.IMPORT_FIXED_OBJECTS_STATS (
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN  DEFAULT to_no_invalidate_type(
                                    get_param('NO_INVALIDATE')),
   force         BOOLEAN  DEFAULT FALSE);
```

Parameters

Table 197-77 IMPORT_FIXED_OBJECTS_STATS Procedure Parameters

Parameter	Description
stattab	User statistics table identifier describing from where to retrieve the statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different from current schema)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Overrides statistics lock: • TRUE - Ignores the statistics lock and imports the statistics • FALSE - The statistics will be imported only if there is no lock

Security Model

You must have the SYSDBA or ANALYZE ANY DICTIONARY system privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

ORA-20002: Bad user statistics table, may need to upgrade it

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

IMPORT_INDEX_STATS Procedure

This procedure retrieves statistics for a particular index from the user statistics table identified by stattab and stores them in the dictionary.

Syntax

```
DBMS_STATS.IMPORT_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT to_no_invalidate_type(
                                 get_param('NO_INVALIDATE')),
   force         BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-78 IMPORT_INDEX_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
indname	Name of the index
partname	Name of the index partition. If the index is partitioned and if partname is NULL, then global and partition index statistics are imported.
stattab	User statistics table identifier describing from where to retrieve the statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)

Table 197-78 (Cont.) IMPORT_INDEX_STATS Procedure Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Imports statistics even if index statistics are locked

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

ORA-20005: Object statistics are locked

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Oracle does not support export or import of statistics across databases of different character sets.

IMPORT_SCHEMA_PREFS Procedure

This procedure is used to import the statistics preferences of all the tables owned by the specified owner name.

Syntax

```
DBMS_STATS.IMPORT_SCHEMA_PREFS (
    ownname    IN  VARCHAR2,
    stattab    IN  VARCHAR2,
    statid     IN  VARCHAR2 DEFAULT NULL,
    statown    IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-79 IMPORT_SCHEMA_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
stattab	Statistics table name from where to import the statistics

Table 197-79 (Cont.) IMPORT_SCHEMA_PREFS Procedure Parameters

Parameter	Description
statid	(Optional) Identifier to associate with these statistics within stattab
statown	Schema containing stattab (if other than ownname)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

- To run this procedure, you need to connect as owner, or have the SYSDBA privilege, or have the ANALYZE ANY system privilege.
- All arguments are of type VARCHAR2 and values are enclosed in quotes.
- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.IMPORT_SCHEMA_PREFS('SH', 'STAT');
```

IMPORT_SCHEMA_STATS Procedure

This procedure imports statistics for all objects in the schema identified by ownname from the user statistics table and stores them in the data dictionary.

Syntax

```
DBMS_STATS.IMPORT_SCHEMA_STATS (
   ownname          VARCHAR2,
   stattab          VARCHAR2,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type(
                                       get_param('NO_INVALIDATE')),
   force            BOOLEAN DEFAULT FALSE,
   stat_category    VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-80 IMPORT_SCHEMA_STATS Procedure Parameters

Parameter	Description
ownname	Specifies the name of the schema.
stattab	Identifies the user table that stores the statistics to be imported.
statid	Specifies the ID associated with these statistics within stattab.
statown	Specifies the schema containing stattab (if different than ownname).

Table 197-80 (Cont.) IMPORT_SCHEMA_STATS Procedure Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Specifies whether to override statistics locked at the object level. The possible values are as follows: • TRUE — Ignores the statistics lock and imports the statistics. • FALSE — Imports the statistics only if there is no lock. This is the default.
stat_category	Specifies which statistics to process. The following values are supported: • OBJECT_STATS — specifies table statistics, column statistics, and index statistics • SYNOPSES — specifies metadata for incremental statistics • REALTIME_STATS — specifies only real-time statistics You can specify a list of comma-delimited values. For example, 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS '.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

IMPORT_SYSTEM_STATS Procedure

This procedure retrieves system statistics from the user statistics table, identified by stattab, and stores the statistics in the dictionary.

Syntax

```
DBMS_STATS.IMPORT_SYSTEM_STATS (
   stattab       VARCHAR2,
```

```
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-81 IMPORT_SYSTEM_STATS Procedure Parameters

Parameter	Description
stattab	Identifier of the user statistics table where the statistics will be retrieved
statid	Optional identifier associated with the statistics retrieved from the stattab
statown	Schema containing stattab (if different from current schema)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

ORA-20002: Bad user statistics table; may need to be upgraded

ORA-20003: Unable to import system statistics

Usage Notes

To run this procedure, you need the GATHER_SYSTEM_STATISTICS role.

Oracle does not support export or import of statistics across databases of different character sets.

IMPORT_TABLE_PREFS Procedure

This procedure is used to set the statistics preferences of the specified table in the specified schema.

Syntax

```
DBMS_STATS.IMPORT_TABLE_PREFS (
    ownname     IN  VARCHAR2,
    tabname     IN  VARCHAR2,
    stattab     IN  VARCHAR2,
    statid      IN  VARCHAR2 DEFAULT NULL,
    statown     IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-82 IMPORT_TABLE_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name
stattab	Statistics table name from where to import the statistics
statid	(Optional) Identifier to associate with these statistics within stattab
statown	Schema containing stattab (if other than ownname)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

Usage Notes

- To run this procedure, you need to connect as owner of the table, or have the ANALYZE ANY system privilege.
- All arguments are of type VARCHAR2 and values are enclosed in quotes.
- Oracle does not support export or import of statistics across databases of different character sets.

Examples

```
DBMS_STATS.IMPORT_TABLE_PREFS('SH', 'SALES', 'STAT');
```

IMPORT_TABLE_STATS Procedure

This procedure import statistics for a specified table from the user statistics table identified by stattab and stores them in the data dictionary.

Syntax

```
DBMS_STATS.IMPORT_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2,
   statid           VARCHAR2 DEFAULT NULL,
   cascade          BOOLEAN  DEFAULT TRUE,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT to_no_invalidate_type(
                                     get_param('NO_INVALIDATE')),
   force            BOOLEAN  DEFAULT FALSE,
   stat_category    VARCHAR2 DEFAULT DEFAULT_STAT_CATEGORY);
```

Parameters

Table 197-83 IMPORT_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	Specifies the name of the schema.
tabname	Specifies the name of the table.
partname	Name of the table partition. If the table is partitioned and if partname is NULL, then global and partition table statistics are imported.
stattab	Identifies the user statistics table that describes where to retrieve the statistics.
statid	Specifies the ID associated with these statistics within stattab.
cascade	Indicates whether to import column and index statistics for this table. The default is TRUE.
statown	Specifies the schema containing stattab (if different than ownname).

Table 197-83 (Cont.) IMPORT_TABLE_STATS Procedure Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure
force	Specifies whether to override statistics locked at the object level. The possible values are as follows: • TRUE — Ignores the statistics lock and imports the statistics. • FALSE — Imports the statistics only if there is no lock. This is the default.
stat_category	Specifies which statistics to process. The following values are supported: • OBJECT_STATS — specifies table statistics, column statistics, and index statistics • SYNOPSES — specifies metadata for incremental statistics • REALTIME_STATS — specifies only real-time statistics You can specify a list of comma-delimited values. For example, 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses. The default value is 'OBJECT_STATS, SYNOPSES, REALTIME_STATS'.

Security Model

To invoke this procedure you must be owner of the table or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table or have either the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values in the user statistics table

Usage Notes

Oracle Database does not support export or import of statistics across databases of different character sets.

INTERRUPT_ADVISOR_TASK Procedure

This procedure interrupts a currently executing Optimizer Statistics Advisor task.

The task ends its operations as it does when at a normal exit, at which point you can access intermediate results. You can also resume the task using the "RESUME_ADVISOR_TASK Procedure".

Syntax

```
DBMS_STATS.INTERRUPT_ADVISOR_TASK (
  task_name IN VARCHAR2);
```

Parameters

Table 197-84 INTERRUPT_ADVISOR_TASK Procedure Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Consider a case in which a task is executed by one user, interrupted, and then resumed by a different user. In this case, Optimizer Statistics Advisor bases its checks of the resumed execution on the privilege of the user who resumed the task.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Example 197-11 Interrupting an Optimizer Statistics Advisor Task

In this example, you start a SQL*Plus session, and then create and execute an advisor task named my_task:

```
DECLARE
  v_tname   VARCHAR2(128) := 'my_task';
  v_ename   VARCHAR2(128) := NULL;
BEGIN
  -- create a task
  v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);

  -- execute the task
  v_ename := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
END;
/
```

In a separate terminal, you start a second SQL*Plus session, and then execute the following program:

```
XEC DBMS_STATS.INTERRUPT_ADVISOR_TASK('my_task');
```

The first session returns an ORA-13632 to indicate the cancelation of the task:

```
ORA-13638: The user interrupted the current operation.
```

LOCK_PARTITION_STATS Procedure

This procedure enables the user to lock statistics for a partition.

Syntax

```
DBMS_STATS.LOCK_PARTITION_STATS (
    ownname    VARCHAR2,
    tabname    VARCHAR2,
    partname   VARCHAR2);
```

Parameters

Table 197-85 LOCK_PARTITION_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema to lock
tabname	Name of the table
partname	[Sub]Partition name

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

LOCK_SCHEMA_STATS Procedure

This procedure locks the statistics of all tables of a schema.

Syntax

```
DBMS_STATS.LOCK_SCHEMA_STATS (
   ownname    VARCHAR2);
```

Parameters

Table 197-86 LOCK_SCHEMA_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema to lock

Usage Notes

- To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

- When statistics on a table are locked, all the statistics depending on the table, including table statistics, column statistics, histograms and statistics on all dependent indexes, are considered to be locked.
- The `SET_*, DELETE_*, IMPORT_*, GATHER_*` procedures that modify statistics in the dictionary of an individual table, index or column will raise an error if statistics of the object is locked.
- Procedures that operates on multiple objects (such as `GATHER_SCHEMA_STATS`) will skip modifying the statistics of an object if it is locked. Many procedures have force argument to override the lock.
- This procedure either freezes the current set of the statistics or keeps the statistics empty (uncollected) to use dynamic statistics.
- The locked or unlocked state is not exported along with the table statistics when using `EXPORT_*_STATS` procedures.
- Neither the UNLOCK_SCHEMA_STATS Procedure nor the UNLOCK_TABLE_STATS Procedure is designed to unlock statistics of corresponding partitions. When you invoke the LOCK_TABLE_STATS Procedure, it sets the statistics lock bit at the table level. In that case, you cannot gather statistics on dependent objects such as partitions and indexes. By the same token, if table statistics are locked, the dependents are locked and you do not need to explicitly invoke the LOCK_PARTITION_STATS Procedure.

LOCK_TABLE_STATS Procedure

This procedure locks the statistics on the table.

Syntax

```
DBMS_STATS.LOCK_TABLE_STATS (
   ownname    VARCHAR2,
   tabname    VARCHAR2);
```

Parameters

Table 197-87 LOCK_TABLE_STATS Procedure Parameters

Parameter	Description
`ownname`	Name of the schema
`tabname`	Name of the table

Usage Notes

- To invoke this procedure you must be owner of the table, or you need the `ANALYZE ANY` privilege. For objects owned by `SYS`, you need to be either the owner of the table, or you need the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.
- When statistics on a table are locked, all the statistics depending on the table, including table statistics, column statistics, histograms and statistics on all dependent indexes, are considered to be locked.
- The `SET_*, DELETE_*, IMPORT_*, GATHER_*` procedures that modify statistics in the dictionary of an individual table, index or column will raise an error if statistics of the object is locked.

- Procedures that operates on multiple objects (such as `GATHER_SCHEMA_STATS`) will skip modifying the statistics of an object if it is locked. Many procedures have force argument to override the lock.
- This procedure either freezes the current set of the statistics or keeps the statistics empty (uncollected) to use dynamic statistics.
- The locked or unlocked state is not exported along with the table statistics when using `EXPORT_*_STATS` procedures.
- Neither the UNLOCK_SCHEMA_STATS Procedure nor the UNLOCK_TABLE_STATS Procedure is designed to unlock statistics of corresponding partitions. When you invoke the LOCK_TABLE_STATS Procedure, it sets the statistics lock bit at the table level. In that case, you cannot gather statistics on dependent objects such as partitions and indexes. By the same token, if table statistics are locked, the dependents are locked and you do not need to explicitly invoke the LOCK_PARTITION_STATS Procedure.

MERGE_COL_USAGE Procedure

This procedure merges column usage information from a source database by means of a `dblink` into the local database.

If column usage information already exists for a given table or column `MERGE_COL_USAGE` will combine both the local and the remote information.

Syntax

```
DBMS_STATS.MERGE_COL_USAGE (
   dblink    IN    VARCHAR2);
```

Parameters

Table 197-88 MERGE_COL_USAGE Procedure Parameters

Parameter	Description
`dblink`	Name of `dblink`

Usage Notes

User must be `SYS` to execute this procedure. addition the user specified during the creation of the `dblink` is expected to have privileges to select from tables in the `SYS` schema.

Exceptions

`ORA-20000`: Insufficient privileges

`ORA-20001`: Parameter `dblink` cannot be `NULL`

`ORA-20002`: Unable to create a `TEMP` table

PREPARE_COLUMN_VALUES Procedures

These procedures convert user-specified minimum, maximum, and histogram endpoint actual values into Oracle's internal representation for future storage using `SET_COLUMN_STATS`.

Syntax

```
DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   charvals        CHARARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec      IN OUT StatRec,
   datevals         DATEARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   dblvals         DBLARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   fltvals         FLTARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   numvals         NUMARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   rawvals         RAWARRAY);
```

Parameters

Table 197-89 PREPARE_COLUMN_VALUES Procedure Parameters

Parameter	Description
`srec.epc`	Number of values specified in `charvals`, `datevals`, `dblvals`, `fltvals`, `numvals`, or `rawvals`. This value must be between 2 and 2050, inclusive, and it should be set to `2` for procedures which do not allow histogram information (`nvarchar` and `rowid`). The first corresponding array entry should hold the minimum value for the column, and the last entry should hold the maximum. If there are more than two entries, then all the others hold the remaining height-balanced or frequency histogram endpoint values (with in-between values ordered from next-smallest to next-largest). This value may be adjusted to account for compression, so the returned value should be left as is for a call to `SET_COLUMN_STATS`.
`srec.bkvals`	If you want a frequency or hybrid histogram, this array contains the number of occurrences of each distinct value specified in `charvals`, `datevals`, `dblvals`, `fltvals`, `numvals`, or `rawvals`. Otherwise, it is merely an output parameter, and it must be set to `NULL` when this procedure is called.

Table 197-89 (Cont.) PREPARE_COLUMN_VALUES Procedure Parameters

Parameter	Description
srec.rpcnts	If you want a hybrid histogram, this array contains the total frequency of values that are less than or equal to each distinct value specified in charvals, datevals, numvals, or rawvals. Otherwise, it is merely an output argument and must be set to NULL when this procedure is called. As an example, for a given array numvals with numvals(i)=4, rpcnts(i)=13 means that there are 13 rows in the column which are less than or equal to 4. Note: • Whenever srec.rpcnts is populated, srec.bkvals must be populated as described above. • Whenever bkvals and/or rpcnts are populated, there should not be any duplicates in charvals, datevals, numvals, or rawvals.

Datatype-specific input parameters (use one) are shown in Table 197-90.

Table 197-90 Datatype-Specific Input Parameters

Type	Description
charvals	The array of values when the column type is character-based. Up to the first 64 bytes of each string should be provided. Arrays must have between 2 and 2050 entries, inclusive. If the datatype is fixed CHAR, the strings must be space-padded to 15 characters for correct normalization.
datevals	Array of values when the column type is date-based
dblvals	Array of values when the column type is double-based
fltvals	Array of values when the column type is float-based
numvals	Array of values when the column type is numeric-based
rawvals	Array of values when the column type is RAW. Up to the first 64 bytes of each value should be provided.
nvmin, nvmax	Minimum and maximum values when the column type is national character set based. No histogram information can be provided for a column of this type. If the datatype is fixed CHAR, the strings must be space-padded to 15 characters for correct normalization.
rwmin, rwmax	Minimum and maximum values when the column type is rowid. No histogram information is provided for a column of this type.

Output Parameters

Table 197-91 PREPARE_COLUMN_VALUES Procedure Output Parameters

Parameter	Description
srec.minval	Internal representation of the minimum suitable for use in a call to SET_COLUMN_STATS
srec.maxval	Internal representation of the maximum suitable for use in a call to SET_COLUMN_STATS
srec.bkvals	Array suitable for use in a call to SET_COLUMN_STATS

Table 197-91 (Cont.) PREPARE_COLUMN_VALUES Procedure Output Parameters

Parameter	Description
srec.novals	Array suitable for use in a call to SET_COLUMN_STATS
srec.eavals	Array suitable for use in a call to SET_COLUMN_STATS
srec.rpcnts	Array suitable for use in a call to SET_COLUMN_STATS

Exceptions

ORA-20001: Invalid or inconsistent input values

Usage Notes

No special privilege or role is needed to invoke this procedure.

PREPARE_COLUMN_VALUES_NVARCHAR Procedure

This procedure converts user-specified minimum, maximum, and histogram endpoint actual values into Oracle's internal representation for future storage using the SET_COLUMN_STATS Procedures.

Syntax

```
DBMS_STATS.PREPARE_COLUMN_VALUES_NVARCHAR (
   srec     IN OUT StatRec,
   nvmin           NVARCHAR2,
   nvmax           NVARCHAR2);
```

Parameters

Table 197-92 PREPARE_COLUMN_VALUES_NVARCHAR Procedure Parameters

Parameter	Description
srec.epc	Number of values specified in charvals, datevals, dblvals, fltvals, numvals, or rawvals. This value must be between 2 and 2050, inclusive, and it should be set to 2 for procedures which do not allow histogram information (nvarchar and rowid). The first corresponding array entry should hold the minimum value for the column, and the last entry should hold the maximum. If there are more than two entries, then all the others hold the remaining height-balanced or frequency histogram endpoint values (with in-between values ordered from next-smallest to next-largest). This value may be adjusted to account for compression, so the returned value should be left as is for a call to SET_COLUMN_STATS.
srec.bkvals	If you want a frequency or hybrid histogram, then this array contains the number of occurrences of each distinct value specified in charvals, datevals, dblvals, fltvals, numvals, or rawvals. Otherwise, it is merely an output parameter, and it must be set to NULL when this procedure is called.

Table 197-92 (Cont.) PREPARE_COLUMN_VALUES_NVARCHAR Procedure Parameters

Parameter	Description
srec.rpcnts	If you want a hybrid histogram, this array contains the total frequency of values that are less than or equal to each distinct value specified in charvals, datevals, numvals, or rawvals. Otherwise, it is merely an output argument and must be set to NULL when this procedure is called. As an example, for a given array numvals with numvals(i)=4, rpcnts(i)=13 means that there are 13 rows in the column which are less than or equal to 4. Note: • Whenever srec.rpcnts is populated, srec.bkvals must be populated as described above. • Whenever bkvals and/or rpcnts are populated, there should not be any duplicates in charvals, datevals, numvals, or rawvals.

Datatype-specific input parameters (use one) are shown in Table 197-90.

Table 197-93 PREPARE_COLUMN_VALUES_NVARCHAR Datatype-Specific Input Parameters

Type	Description
nvmin, nvmax	The minimum and maximum values when the column type is national character set based. No histogram information can be provided for a column of this type. If the datatype is fixed CHAR, the strings must be space-padded to 15 characters for correct normalization.

Output Parameters

Table 197-94 PREPARE_COLUMN_VALUES_NVARCHAR Procedure Output Parameters

Parameter	Description
srec.minval	Internal representation of the minimum suitable for use in a call to SET_COLUMN_STATS
srec.maxval	Internal representation of the maximum suitable for use in a call to SET_COLUMN_STATS
srec.bkvals	Array suitable for use in a call to SET_COLUMN_STATS.
srec.novals	Array suitable for use in a call to SET_COLUMN_STATS
srec.eavals	Array suitable for use in a call to SET_COLUMN_STATS
srec.rpcnts	Array suitable for use in a call to SET_COLUMN_STATS

Exceptions

ORA-20001: Invalid or inconsistent input values

Usage Notes

No special privilege or role is needed to invoke this procedure.

Related Topics

- SET_COLUMN_STATS Procedures
  This procedure sets column-related information.

PREPARE_COLUMN_VALUES_ROWID Procedure

This procedure converts user-specified minimum, maximum, and histogram endpoint datatype-specific values into Oracle's internal representation for future storage using `SET_COLUMN_STATS`.

Syntax

```
DBMS_STATS.PREPARE_COLUMN_VALUES_ROWID (
   srec  IN OUT StatRec,
   rwmin        ROWID,
   rwmax        ROWID);
```

Pragmas

```
pragma restrict_references(prepare_column_values_rowid, WNDS, RNDS, WNPS, RNPS);
```

Parameters

Table 197-95 PREPARE_COLUMN_VALUES_ROWID Procedure Parameters

Parameter	Description
srec	Values (`IN`): • `epc` • `bkvals` • `rpcnts` Values (`OUT`): • `minval` • `maxval` • `bkvals` • `novals` • `eavals` • `rpcnts`
rwmin	Minimum value when the column type is `rowid`. No histogram information is provided for a column of this type.
rwmax	Maximum value when the column type is `rowid`. No histogram information is provided for a column of this type.

Table 197-96 StatRec Record Type Fields

Field	Description
epc (IN)	Number of values specified in charvals, datevals, dblvals, fltvals, numvals, or rawvals. This value must be between 2 and 2050, inclusive, and it should be set to 2 for procedures which do not allow histogram information (nvarchar and rowid). The first corresponding array entry should hold the minimum value for the column, and the last entry should hold the maximum. If there are more than two entries, then all the others hold the remaining height-balanced or frequency histogram endpoint values (with in-between values ordered from next-smallest to next-largest). This value may be adjusted to account for compression, so the returned value should be left as is for a call to SET_COLUMN_STATS.
bkvals (IN)	If you want a frequency or hybrid histogram, this array contains the number of occurrences of each distinct value specified in charvals, datevals, dblvals, fltvals, numvals, or rawvals. Otherwise, it is merely an output parameter, and it must be set to NULL when this procedure is called.
rpcnts (IN)	If you want a hybrid histogram, this array contains the total frequency of values that are less than or equal to each distinct value specified in charvals, datevals, numvals, or rawvals. Otherwise, it is merely an output argument and must be set to NULL when this procedure is called. As an example, for a given array numvals with numvals(i)=4, rpcnts(i)=13 means that there are 13 rows in the column which are less than or equal to 4. Note: • Whenever srec.rpcnts is populated, srec.bkvals must be populated as described above. • Whenever bkvals and/or rpcnts are populated, there should not be any duplicates in charvals, datevals, numvals, or rawvals.
minval (OUT)	Internal representation of the minimum suitable for use in a call to SET_COLUMN_STATS.
maxval (OUT)	Internal representation of the maximum suitable for use in a call to SET_COLUMN_STATS.
bkvals (OUT)	Array suitable for use in a call to SET_COLUMN_STATS.
novals (OUT)	Array suitable for use in a call to SET_COLUMN_STATS.
eavals (OUT)	Array suitable for use in a call to SET_COLUMN_STATS
rpcnts (OUT)	Array suitable for use in a call to SET_COLUMN_STATS

Usage Notes

No special privilege or role is needed to invoke this procedure.

PUBLISH_PENDING_STATS Procedure

This procedure is used to publish the statistics gathered and stored as pending.

Syntax

```
DBMS_STATS.PUBLISH_PENDING_STATS (
    ownname          IN  VARCHAR2 DEFAULT USER,
    tabname          IN  VARCHAR2,
    no_invalidate        BOOLEAN DEFAULT
       TO_NO_INVALIDATE_TYPE(GET_PARAM('NO_INVALIDATE')),
    force            IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-97 PUBLISH_PENDING_STATS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	If TRUE, will override the lock

Security Model

To run this procedure, you must have the same privilege for gathering statistics on the tables that will be touched by this procedure.

Exceptions

ORA-20000: Insufficient privileges

Usage Notes

- If the parameter tabname is NULL then publish applies to all tables of the specified schema.
- The default owner/schema is the user who runs the procedure.

Examples

```
DBMS_STATS.PUBLISH_PENDING_STATS ('SH', null);
```

PURGE_STATS Procedure

This procedure purges old versions of statistics saved in the dictionary.

To run this procedure, you must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege.

Syntax

```
DBMS_STATS.PURGE_STATS(
    before_timestamp       TIMESTAMP WITH TIME ZONE);
```

Parameters

Table 197-98 PURGE_STATS Procedure Parameters

Parameter	Description
before_timestamp	Versions of statistics saved before this timestamp are purged. If NULL, it uses the purging policy used by automatic purge. The automatic purge deletes all history older than the older of (current time - statistics history retention) and (time of recent analyze in the system - 1). The statistics history retention value can be changed using ALTER_STATS_HISTORY_RETENTION Procedure.The default is 31 days.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

Usage Notes

To invoke this procedure you need the ANALYZE ANY privilege and the ANALYZE ANY DICTIONARY privilege.

REMAP_STAT_TABLE Procedure

This procedure remaps the names of objects in the user statistics table. It allows you to import the statistics to objects with same definition but with different names.

Syntax

```
DBMS_STATS.REMAP_STAT_TABLE (
   ownname    IN    VARCHAR2,
   stattab    IN    VARCHAR2,
   src_own    IN    VARCHAR2,
   src_tab    IN    VARCHAR2,
   tgt_own    IN    VARCHAR2,
   tgt_tab    IN    VARCHAR2);
```

Parameters

Table 197-99 REMAP_STAT_TABLE Procedure Parameters

Parameter	Description
ownname	Owner of the statistics table. NULL means the current schema.
stattab	User statistics table identifier
src_own	Owner of the table to be renamed. This argument cannot be NULL.
src_tab	Name of the table to be renamed. If NULL, all tables are owned by src_own.
tgt_own	New name of the owner of the table. The owner name is also updated for the dependent objects such as columns and indexes. Note that an index of src_tab not owned by src_own is not renamed.This argument cannot be NULL.
tgt_tab	New name of the table. This argument is valid only if src_tab is not NULL.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Invalid input

Examples

The following statement remaps all objects of sh to shsave in user statistics table sh.ustat:

```
DBMS_STATS.REMAP_STAT_TABLE ('sh', 'ustat', 'sh', NULL, 'shsave', NULL);
```

The following statement can be used to import statistics into objects of shsave once the preceding remap procedure is completed:

```
DBMS_STATS.IMPORT_SCHEMA_STATS ('shsave', 'ustat', statown => 'sh');
```

The following statement remaps sh.customers to shsave.customers_sav:

```
DBMS_STATS.REMAP_STAT_TABLE ('sh', 'ustat', 'sh', 'customers','shsave', 'customers_sav');
```

REPORT_ADVISOR_TASK Function

This function reports the results of an Optimizer Statistics Advisor task.

Syntax

```
DBMS_STATS.REPORT_ADVISOR_TASK(
  task_name          IN   VARCHAR2,
  execution_name     IN   VARCHAR2   := NULL,
  type               IN   VARCHAR2   := 'TEXT',
  section            IN   VARCHAR2   := 'ALL',
  level              IN   VARCHAR2   := 'TYPICAL')
RETURN CLOB;
```

Parameters

Table 197-100 REPORT_ADVISOR_TASK Function Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.
execution_name	A name that qualifies and identifies an advisor task execution. If not specified, the function uses the latest execution of the specified task.
type	The type of the Optimizer Statistics Advisor report. Possible values are TEXT, HTML, and XML.
section	A section in the report. Possible values are SUMMARY, FINDINGS, ERRORS, and ALL. You can specify combinations of different values can be using the plus (+) and minus (-) operator, as in 'SUMMARY +FINDINGS +ERRORS', and 'ALL -ERRORS'.
level	The format of the report. Possible values are BASIC, TYPICAL, ALL, and SHOW_HIDDEN. You can specify SHOW_HIDDEN together with the other three input values, as in 'BASIC +SHOW_HIDDEN' and 'TYPICAL +SHOW_HIDDEN'.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- You can execute this subprogram for AUTO_STATS_ADVISOR_TASK, which is predefined.
- This subprogram executes using invoker's rights.

The results of performing this task depend on the privileges of the executing user:

- SYSTEM level

  Only users with both the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges can perform this task on system-level rules.
- Operation level

  The results depend on the following privileges:

  – Users with both the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges can perform this task for all statistics operations.

  – Users with the ANALYZE ANY privilege but *not* the ANALYZE ANY DICTIONARY privilege can perform this task for statistics operations related to any schema except SYS.

  – Users with the ANALYZE ANY DICTIONARY privilege but *not* the ANALYZE ANY privilege can perform this task for statistics operations related to their own schema and the SYS schema.

  – Users with neither the ANALYZE ANY nor the ANALYZE ANY DICTIONARY privilege can only perform this operation for statistics operations relating to their own schema.
- Object level

  Users can perform this task for any object for which they have statistics collection privileges.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: User input errors
- ORA-20012: Optimizer Statistics Advisor errors

Returns

This function returns a CLOB that contains the report.

Examples

(Optional) List and briefly describe the examples for using the API or subprogram here.

Example 197-12 Generating an HTML Report

This example creates a procedure named myrep, and then calls this procedure to generate an HTML report.

```
SET ECHO ON
SET FEEDBACK ON
SET SERVEROUTPUT ON
SET TRIMS ON
SET LINESIZE 300

EXECUTE DBMS_OUTPUT.ENABLE (buffer_size => 10000000);

CREATE OR REPLACE PROCEDURE myrep(p_tname VARCHAR2, p_ftype VARCHAR2, which
VARCHAR2)
IS
  v_report CLOB          := null;
  v_script CLOB          := null;
  v_ftype  VARCHAR2(400) := p_ftype;
  v_tname  VARCHAR2(400) := p_tname;
  v_len    NUMBER(10);
  v_ps     NUMBER(10) := 10000;
  v_pn     NUMBER(10) := 1;
  v_ret    VARCHAR2(32767);
BEGIN
   IF which = 'REPORT'
   THEN
      -- generate a report
      v_report := DBMS_STATS.REPORT_ADVISOR_TASK(
                    task_name => v_tname,
                    type      => v_ftype,
                    section   => 'ALL',
                    level     => 'ALL');
      v_len := DBMS_LOB.getlength(v_report);
      WHILE (v_pn < v_len)
      LOOP
        DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBSTR(v_report, v_ps, v_pn));
        v_pn := v_pn + v_ps;
      END LOOP;
   ELSE
      -- generate a script
```

```
      v_script := DBMS_STATS.SCRIPT_ADVISOR_TASK(v_tname);
      v_len := DBMS_LOB.getlength(v_script);
      WHILE (v_pn < v_len)
      LOOP
        DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBSTR( v_script, v_ps, v_pn));
        v_pn := v_pn + v_ps;
      END LOOP;
   END IF;
END;
/

SHOW ERRORS

SPOOL report.txt
EXECUTE myrep('my_task','HTML','REPORT');
SPOOL OFF
```

Example 197-13 Generating a Textual Report for AUTO_STATS_ADVISOR_TASK

The following example invokes the `myrep` procedure created in the preceding example for `AUTO_STATS_ADVISOR_TASK`:

```
EXEC myrep('AUTO_STATS_ADVISOR_TASK','TEXT','REPORT');
```

The following sample output shows part of the report:

```
GENERAL INFORMATION
-------------------------------------------------------------------------------
-

 Task Name       : AUTO_STATS_ADVISOR_TASK
 Execution Name  : EXEC_97
 Created         : 07-08-16 10:18:10
 Last Modified   : 07-11-16 03:02:30

-------------------------------------------------------------------------------
-
SUMMARY
-------------------------------------------------------------------------------
-
 For execution EXEC_97 of task AUTO_STATS_ADVISOR_TASK, the Statistics
Advisor
 has 10 finding(s). The findings are related to the following rules:
 COMPLETEAUTOJOB, MAINTAINSTATSHISTORY, USEDEFAULTPREFERENCE,
 AVOIDSETPROCEDURES, USEDEFAULTPARAMS, USEGATHERSCHEMASTATS, AVOIDSTALESTATS,
 UNLOCKNONVOLATILETABLE, USEINCREMENTAL, AVOIDANALYZETABLE. Please refer to
the
 finding section for detailed information.
-------------------------------------------------------------------------------
-
FINDINGS
-------------------------------------------------------------------------------
-
 Rule Name:         MaintainStatsHistory
 Rule Description:  Maintain Statistics History
```

```
 Finding:  Statistics history tables are too big.

 Recommendation:  Check the other findings of this rule, as well as
the
                  findings for the rules
AvoidFrequentStatsCollection,
                  UseDefaultPreference, UseDefaultParams for possible
causes
                  and
recommendations.
 Rationale:  The size of the statistics history table could be big because
of
             violations of other rules.

--------------------------------------------------
 Rule Name:         UseDefaultPreference
 Rule Description:  Use Default Preference for Stats Collection
 Finding:  Global preference SYS_FLAGS is set to a non-default value '1'.


Recommendation:  Set the value of preference SYS_FLAGS to '0'.
 Example:
 -- Setting preference cascade to default value:
 dbms_stats.set_global_prefs('CASCADE', NULL);

 Rationale:  Preference SYS_FLAGS is for Oracle internal use only, setting
it
             to nondefault value '1' could cause unforeseen consequences.
.
.
.
```

REPORT_COL_USAGE Function

This function reports the recorded column (group) usage information.

Syntax

```
DBMS_STATS.REPORT_COL_USAGE (
   ownname    IN    VARCHAR2,
   tabname    IN    VARCHAR2)
RETURN CLOB;
```

Parameters

Table 197-101 REPORT_COL_USAGE Function Parameters

Parameter	Description
`ownname`	Owner name. If `NULL` it reports column usage information for tables in all schemas in the database.
`tabname`	Table name. If `NULL` it reports column usage information for all tables of `ownname`.

Usage Notes

To run this procedure, you need to have the `SYSDBA` administrative privilege or both `ANALYZE ANY DICTIONARY` and `ANALYZE ANY` system privileges.

REPORT_GATHER_AUTO_STATS Function

This function runs the auto statistics gathering job in reporting mode. That is, statistics are not actually collected, but all the objects that will be affected when auto statistics gathering is invoked are reported.

Syntax

```
DBMS_STATS.REPORT_GATHER_AUTO_STATS (
   detail_level      VARCHAR2  DEFAULT 'TYPICAL',
   format            VARCHAR2  DEFAULT 'TEXT')
 RETURN CLOB;
```

Parameters

Table 197-102 REPORT_GATHER_AUTO_STATS Function Parameters

Parameter	Description
detail_level	Detail level for the content of the report • BASIC: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • TYPICAL: In addition to the information provided at level BASIC, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed-out objects (applies to only auto statistics gathering) With regard to target objects: - owner and name of each target object - target object type (such as: table, index) - start time - end time - completion status • ALL: In addition to the information provided at level TYPICAL, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name - session ID - parameter values - error message if the operation failed With regard to target objects: - job name - batching details - estimated cost - rank in the target list - columns for which histograms were collected - list of collected extended statistics (if any) - reason for including the object in the target list - additional error details if the task has failed. Note that several fields (such as job name, estimated task cost) in the report are populated only when an operation is executed concurrently (CONCURRENT preference is turned on).

Table 197-102 (Cont.) REPORT_GATHER_AUTO_STATS Function Parameters

Parameter	Description
format	Report format: • XML • HTML • TEXT (Default)

Usage Notes

Only user SYS can run the REPORT_GATHER_AUTO_STATS function.

REPORT_GATHER_DATABASE_STATS Functions

This function runs the GATHER_DATABASE_STATS function in reporting mode.

The database does not collect statistics, but reports all objects that would be affected when invoking GATHER_DATABASE_STATS. The input set of parameters is the same as in GATHER_DATABASE_STATS, with two extra parameters.

Syntax

```
DBMS_STATS.REPORT_GATHER_DATABASE_STATS (
   estimate_percent     IN    NUMBER     DEFAULT to_estimate_percent_type (
                                             GET_PARAM('ESTIMATE_PERCENT')),
   block_sample         IN    BOOLEAN    DEFAULT FALSE,
   method_opt           IN    VARCHAR2   DEFAULT GET_PARAM('METHOD_OPT'),
   degree               IN    NUMBER     DEFAULT TO_DEGREE_TYPE(
                                             GET_PARAM('DEGREE')),
   granularity          IN    VARCHAR2   DEFAULT GET_PARAM('GRANULARITY'),
   cascade              IN    BOOLEAN    DEFAULT to_cascade_type (
                                             GET_PARAM('CASCADE')),
   stattab              IN    VARCHAR2   DEFAULT NULL,
   statid               IN    VARCHAR2   DEFAULT NULL,
   options              IN    VARCHAR2   DEFAULT 'GATHER',
   statown              IN    VARCHAR2   DEFAULT NULL,
   gather_sys           IN    BOOLEAN    DEFAULT TRUE,
   no_invalidate        IN    BOOLEAN    DEFAULT TO_NO_INVALIDATE_TYPE (
                                             GET_PARAM('NO_INVALIDATE')),
   gather_temp          IN    BOOLEAN    DEFAULT FALSE,
   gather_fixed         IN    BOOLEAN    DEFAULT FALSE,
   stattype             IN    VARCHAR2   DEFAULT DATA,
   obj_filter_list      IN    ObjectTab  DEFAULT NULL,
   detail_level         IN    VARCHAR2   DEFAULT 'TYPICAL',
   format               IN    VARCHAR2   DEFAULT 'TEXT')
 RETURN CLOB;
```

Parameters

Table 197-103 REPORT_GATHER_DATABASE_STATS Function Parameters

Parameter	Description
estimate_percent	The percentage of rows to use for the sample size. The valid range is between 0.000001 and 100. The null value means to compute. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for good statistics. This is the default. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Whether or not to use random block sampling instead of random row sampling. Random block sampling is more efficient, but if the data is not randomly distributed on disk, then the sample values may be somewhat correlated. This parameter is only relevant when estimating statistics.
method_opt	Method options. This parameter accepts the following values: FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause]. size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} - integer : Number of histogram buckets. Must be in the range [1,2048]. - REPEAT : Collects histograms only on the columns that already have histograms. - AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. - SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO.The value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.

Table 197-103 (Cont.) REPORT_GATHER_DATABASE_STATS Function Parameters

Parameter	Description
granularity	Determines the granularity of statistics to collect. This preference is only relevant for partitioned tables. The following values are valid: • ALL — Gathers all statistics: subpartition, partition, and global. • AUTO — Determines the granularity based on the partitioning type. This is the default value. • DEFAULT — Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. Use GLOBAL AND PARTITION for this functionality. • GLOBAL — Gathers global statistics. • GLOBAL AND PARTITION — Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. • PARTITION — Gathers partition-level statistics. • SUBPARTITION — Gathers subpartition-level statistics.
cascade	Gather statistics on the indexes as well. Using this option is equivalent to running the GATHER_INDEX_STATS Procedure on each of the indexes in the database in addition to gathering table and column statistics. Use the constant DBMS_STATS.AUTO_CASCADE to have Oracle determine whether index statistics to be collected or not. This is the default. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics. The database assumes that the statistics table resides in the same schema as the object being analyzed. Thus, to use this option, one such table must exist in each schema.
statid	Identifier (optional) to associate with these statistics within stattab.
options	Further specification of which objects to gather statistics for: GATHER: Gathers statistics on all objects in the schema. GATHER AUTO: Gathers all necessary statistics automatically. Oracle implicitly determines which objects need new statistics, and determines how to gather those statistics. When GATHER AUTO is specified, the only additional valid parameters are stattab, statid, objlist and statown; all other parameter settings are ignored. Returns a list of processed objects. GATHER STALE: Gathers statistics on stale objects as determined by looking at the *_tab_modifications views. Also, return a list of objects found to be stale. GATHER EMPTY: Gathers statistics on objects which currently have no statistics. Return a list of objects found to have no statistics. LIST AUTO: Returns a list of objects to be processed with GATHER AUTO LIST STALE: Returns a list of stale objects as determined by looking at the *_tab_modifications views LIST EMPTY: Returns a list of objects which currently have no statistics
statown	Schema containing stattab, if different from current schema.

Table 197-103 (Cont.) REPORT_GATHER_DATABASE_STATS Function Parameters

Parameter	Description
gather_sys	Gathers statistics on the objects owned by the SYS user.
no_invalidate	Does not invalidate the dependent cursors if set to TRUE. The procedure invalidates the dependent cursors immediately if set to FALSE. Use DBMS_STATS.AUTO_INVALIDATE. to have Oracle decide when to invalidate dependent cursors. This is the default. The default can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
gather_temp	Gathers statistics on global temporary tables when TRUE. The default is FALSE. The temporary table must be created with the ON COMMIT PRESERVE ROWS clause. Also, the statistics collected are based on the data in the session in which this procedure is run, but they are shared across all sessions.
gather_fixed	Gather statistics on fixed tables when TRUE. The default is FALSE. Only user SYS can collect statistics for fixed tables. The ownname must be SYS or null. When gathering statistics for fixed tables, the database ignores specified values for the following arguments: • estimate_percent • block_sample • stattab • statid • statown The database does not invalidate the dependent cursors on fixed tables on which stats is collected. This option is meant for internal use only.
stattype	The type of statistics: • DATA — Data statistics only • CACHE — Cache statistics only • ALL — All statistics
obj_filter_list	A list of object filters. The attribute values specified in the object filter are case-insensitive unless double-quoted. Wildcards are allowed in the attribute values. When specified, GATHER_DATABASE_STATS gathers statistics only on objects that satisfy at least one object filter in the list as needed. In a single-object filter, the database can specify the constraints on the object attributes. For example, non-NULL values s1 and s2 are specified for attributes a1 and a2 in one object filter. An object o is said to satisfy this object filter when (o.a1 LIKE s1) AND (o.a2 LIKE s2) is true.

Table 197-103 (Cont.) REPORT_GATHER_DATABASE_STATS Function Parameters

Parameter	Description
detail_level	The level of detail for the content of the report. Valid values are as follows: • BASIC: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • TYPICAL: In addition to the information provided at level BASIC, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed-out objects (applies to only auto statistics gathering) With regard to target objects: - owner and name of each target object - target object type (such as: table, index) - start time - end time - completion status • ALL: In addition to the information provided at level TYPICAL, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name - session ID - parameter values - error message if the operation failed With regard to target objects: - job name - batching details - estimated cost - rank in the target list - columns for which histograms were collected - list of collected extended statistics (if any) - additional error details if the task has failed. Note that several fields (such as job name, estimated task cost) in the report are populated only when an operation is executed concurrently (CONCURRENT preference is turned on).
format	The format of the report. Valid values are: • XML • HTML • TEXT (Default)

Return Values

A CLOB object that contains the report

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Bad input value

Usage Notes

To run this procedure, you need to have the SYSDBA role or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.

REPORT_GATHER_DICTIONARY_STATS Functions

This function runs the GATHER_DICTIONARY_STATS procedure in reporting mode.

The database does not collect statistics, but reports all objects affected when invoking GATHER_DICTIONARY_STATS. The detail level for the report is defined by the detail_level input parameter. See the descriptions of detail_level and format in REPORT_GATHER_DICTIONARY_STATS Functions. For all other input parameters, see GATHER_DICTIONARY_STATS Procedure.

Syntax

```
DBMS_STATS.REPORT_GATHER_DICTIONARY_STATS (
   comp_id               IN    VARCHAR2   DEFAULT NULL,
   estimate_percent      IN    NUMBER     DEFAULT TO_ESTIMATE_PERCENT_TYPE

(GET_PARAM('ESTIMATE_PERCENT')),
   block_sample          IN    BOOLEAN    DEFAULT FALSE,
   method_opt            IN    VARCHAR2   DEFAULT GET_PARAM('METHOD_OPT'),
   degree                IN    NUMBER     DEFAULT TO_DEGREE_TYPE
                                              (GET_PARAM('DEGREE')),
   granularity           IN    VARCHAR2   DEFAULT GET_PARAM('GRANULARITY'),
   cascade               IN    BOOLEAN    DEFAULT TO_CASCADE_TYPE
                                              (GET_PARAM('CASCADE')),
   stattab               IN    VARCHAR2   DEFAULT NULL,
   options               IN    VARCHAR2   DEFAULT 'GATHER AUTO',
   no_invalidate         IN    BOOLEAN    DEFAULT TO_NO_INVALIDATE_TYPE
                                              (GET_PARAM('NO_INVALIDATE')),
   stattype              IN    VARCHAR2   DEFAULT 'DATA',
   obj_filter_list       IN    ObjectTab  DEFAULT NULL,
   detail_level          IN    VARCHAR2   DEFAULT 'TYPICAL',
   format                IN    VARCHAR2   DEFAULT 'TEXT')
 RETURN CLOB;
```

Parameters

Table 197-104 REPORT_GATHER_DICTIONARY_STATS Function Parameters

Parameter	Description
comp_id	Component ID of the schema to analyze. NULL results in analyzing schemas of all RDBMS components. Refer refer to the COMP_ID column of the DBA_REGISTRY view. The procedure always gather statistics on SYS and SYSTEM schemas regardless of this argument.
estimate_percent	Percentage of rows to sample (NULL means compute). The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to let the database determine the appropriate sample size for good statistics. This is the default. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Determines whether to use random block sampling instead of random row sampling. Random block sampling is more efficient, but if the data is not randomly distributed on disk then the sample values may be somewhat correlated. Only pertinent when performing estimate statistics.
method_opt	The method options. This parameter accepts the following values: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms. – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.

Table 197-104 (Cont.) REPORT_GATHER_DICTIONARY_STATS Function Parameters

Parameter	Description
granularity	Determines the granularity of statistics to collect. This preference is only relevant for partitioned tables. The following values are valid: • ALL — Gathers all statistics: subpartition, partition, and global. • AUTO — Determines the granularity based on the partitioning type. This is the default value. • DEFAULT — Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. Use GLOBAL AND PARTITION for this functionality. • GLOBAL — Gathers global statistics. • GLOBAL AND PARTITION — Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. • PARTITION — Gathers partition-level statistics. • SUBPARTITION — Gathers subpartition-level statistics.
cascade	Gathers statistics on indexes also. Index statistics gathering will not be parallelized. Using this option is equivalent to running the GATHER_INDEX_STATS Procedure on each of the indexes in the schema in addition to gathering table and column statistics. Use the constant DBMS_STATS.AUTO_CASCADE to have Oracle determine whether index statistics to be collected or not. This is the default.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics.
options	Further specification of objects for which to gather statistics: • 'GATHER' - Gathers statistics on all objects in the schema • 'GATHER AUTO' - Gathers all necessary statistics automatically. Oracle implicitly determines which objects need new statistics and determines how to gather those statistics. When 'GATHER AUTO' is specified, the only additional valid parameters are comp_id, stattab, statid and statown; all other parameter settings will be ignored. Also, returns a list of objects processed. • 'GATHER STALE' - Gathers statistics on stale objects as determined by looking at the *_tab_modifications views. Also, returns a list of objects found to be stale. • 'GATHER EMPTY' - Gathers statistics on objects which currently have no statistics. Also, returns a list of objects found to have no statistics. • 'LIST AUTO' - Returns list of objects to be processed with 'GATHER AUTO' • 'LIST STALE' - Returns list of stale objects as determined by looking at the *_tab_modifications views • 'LIST EMPTY' - Returns list of objects which currently have no statistics

Table 197-104 (Cont.) REPORT_GATHER_DICTIONARY_STATS Function Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattype	The type of statistics: • DATA — Data statistics only • CACHE — Cache statistics only • ALL — All statistics
obj_filter_list	A list of object filters. When provided, this will gather statistics only on objects which satisfy at least one object filter in the list as needed. In a single object filter, we can specify the constraints on the object attributes. The attribute values specified in the object filter are case-insensitive unless double-quoted. Wildcard is allowed in the attribute values. Suppose non-NULL values s1, s2, ... are specified for attributes a1, a2, ... in one object filter. An object o is said to satisfy this object filter if (o.a1 like s1) and (o.a2 like s2) and ... is true.
detail_level	See the description in REPORT_GATHER_DICTIONARY_STATS Functions.
format	See the description in REPORT_GATHER_DICTIONARY_STATS Functions.

Return Values

A CLOB object that contains the report

Usage Notes

You must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege to execute this procedure.

Exceptions

ORA-20000: Index does not exist or insufficient privileges

ORA-20001: Bad input value

ORA-20002: Bad user statistics table, may need to upgrade it

REPORT_GATHER_FIXED_OBJ_STATS Function

This function runs the GATHER_FIXED_OBJECTS_STATS Procedure in reporting mode.

That is, statistics are not actually collected, but all the objects that will be affected when GATHER_FIXED_OBJ_STATS is invoked are reported. The input set of parameters are exactly the same as in GATHER_FIXED_OBJ_STATS with two extra parameters.

Syntax

```
DBMS_STATS.REPORT_GATHER_FIXED_OBJ_STATS (
   stattab          IN  VARCHAR2 DEFAULT NULL,
   statid           IN  VARCHAR2 DEFAULT NULL,
   statown          IN  VARCHAR2 DEFAULT NULL,
   no_invalidate    IN  BOOLEAN  DEFAULT TO_NO_INVALIDATE_TYPE (
                                    GET_PARAM('NO_INVALIDATE')),
   detail_level     IN  VARCHAR2   DEFAULT 'TYPICAL',
   format           IN  VARCHAR2   DEFAULT 'TEXT')
 RETURN CLOB;
```

Parameters

Table 197-105 REPORT_GATHER_FIXED_OBJ_STATS Procedure Parameters

Parameter	Description
`stattab`	User statistics table identifier describing where to save the current statistics
`statid`	Identifier to associate with these statistics within `stattab` (optional)
`statown`	Schema containing `stattab` (if different from current schema)
`no_invalidate`	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • `TRUE`: Dependent cursors are not invalidated. • `FALSE`: Dependent cursors are marked for immediate invalidation. • `AUTO`: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Table 197-105 (Cont.) REPORT_GATHER_FIXED_OBJ_STATS Procedure Parameters

Parameter	Description
`detail_level`	Detail level for the content of the report • `BASIC`: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • `TYPICAL`: In addition to the information provided at level `BASIC`, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed-out objects (applies to only auto statistics gathering) With regard to target objects: - owner and name of each target object - target object type (such as: table, index) - start time - end time - completion status • `ALL`: In addition to the information provided at level `TYPICAL`, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name - session ID - parameter values - error message if the operation failed With regard to target objects: - job name - batching details - estimated cost - rank in the target list - columns for which histograms were collected - list of collected extended statistics (if any) - additional error details if the task has failed. Note that several fields (such as job name, estimated task cost) in the report are populated only when an operation is executed concurrently (`CONCURRENT` preference is turned on).
`format`	Report format: • XML • HTML • TEXT (Default)

Return Values

A CLOB object that contains the report

Usage Notes

You must have the SYSDBA or ANALYZE ANY DICTIONARY system privilege to execute this procedure.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Bad input value

ORA-20002: Bad user statistics table, may need to upgrade it

Related Topics

- GATHER_FIXED_OBJECTS_STATS Procedure
  This procedure gathers statistics for all fixed objects (dynamic performance tables).

REPORT_GATHER_SCHEMA_STATS Functions

This function runs the GATHER_SCHEMA_STATS procedure in reporting mode.

The database does not actually gather statistics, but reports all objects that would be affected when invoking GATHER_SCHEMA_STATS. The input set of parameters is exactly the same as in GATHER_SCHEMA_STATS, with two extra parameters.

Syntax

```
DBMS_STATS.REPORT_GATHER_SCHEMA_STATS (
   ownname           IN   VARCHAR2,
   estimate_percent  IN   NUMBER      DEFAULT TO_ESTIMATE_PERCENT_TYPE (
                                        GET_PARAM ('ESTIMATE_PERCENT')),
   block_sample      IN   BOOLEAN     DEFAULT FALSE,
   method_opt        IN   VARCHAR2    DEFAULT GET_PARAM ('METHOD_OPT'),
   degree            IN   NUMBER      DEFAULT TO_DEGREE_TYPE (
                                        GET_PARAM('DEGREE')),
   granularity       IN   VARCHAR2    DEFAULT GET_PARAM('GRANULARITY'),
   cascade           IN   BOOLEAN     DEFAULT TO_CASCADE_TYPE (
                                        GET_PARAM ('CASCADE')),
   stattab           IN   VARCHAR2    DEFAULT NULL,
   statid            IN   VARCHAR2    DEFAULT NULL,
   options           IN   VARCHAR2    DEFAULT 'GATHER',
   statown           IN   VARCHAR2    DEFAULT NULL,
   no_invalidate     IN   BOOLEAN     DEFAULT TO_NO_INVALIDATE_TYPE (
                                        GET_PARAM ('NO_INVALIDATE')),
   force             IN   BOOLEAN     DEFAULT FALSE,
   obj_filter_list   IN   ObjectTab   DEFAULT NULL,
   detail_level      IN   VARCHAR2    DEFAULT 'TYPICAL',
   format            IN   VARCHAR2    DEFAULT 'TEXT')
 RETURN CLOB;
```

Parameters

Table 197-106 REPORT_GATHER_SCHEMA_STATS Function Parameters

Parameter	Description
ownname	Schema to analyze (NULL means current schema)
estimate_percent	Percentage of rows to estimate (NULL means compute): The valid range is [0.000001,100]. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to have Oracle determine the appropriate sample size for good statistics. This is the default.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Whether or not to use random block sampling instead of random row sampling. Random block sampling is more efficient, but if the data is not randomly distributed on disk, then the sample values may be somewhat correlated. Only pertinent when doing an estimate statistics.
method_opt	Accepts: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.

Table 197-106 (Cont.) REPORT_GATHER_SCHEMA_STATS Function Parameters

Parameter	Description
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics.
cascade	Gather statistics on the indexes as well. Using this option is equivalent to running the GATHER_INDEX_STATS Procedure on each of the indexes in the schema in addition to gathering table and column statistics. Use the constant DBMS_STATS.AUTO_CASCADE to have Oracle determine whether index statistics to be collected or not. This is the default. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics
statid	Identifier (optional) to associate with these statistics within stattab
options	Further specification of which objects to gather statistics for: GATHER: Gathers statistics on all objects in the schema. GATHER AUTO: Gathers all necessary statistics automatically. Oracle implicitly determines which objects need new statistics, and determines how to gather those statistics. When GATHER AUTO is specified, the only additional valid parameters are ownname, stattab, statid, objlist and statown; all other parameter settings are ignored. Returns a list of processed objects. GATHER STALE: Gathers statistics on stale objects as determined by looking at the *_tab_modifications views. Also, return a list of objects found to be stale. GATHER EMPTY: Gathers statistics on objects which currently have no statistics. also, return a list of objects found to have no statistics. LIST AUTO: Returns a list of objects to be processed with GATHER AUTO. LIST STALE: Returns list of stale objects as determined by looking at the *_tab_modifications views. LIST EMPTY: Returns list of objects which currently have no statistics.
objlist	List of objects found to be stale or empty
statown	Schema containing stattab (if different than ownname)

Table 197-106 (Cont.) REPORT_GATHER_SCHEMA_STATS Function Parameters

Parameter	Description
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Gather statistics on objects even if they are locked
obj_filter_list	A list of object filters. When provided, GATHER_SCHEMA_STATS will gather statistics only on objects which satisfy at least one object filter in the list as needed. In a single object filter, we can specify the constraints on the object attributes. The attribute values specified in the object filter are case- insensitive unless double-quoted. Wildcard is allowed in the attribute values. Suppose non-NULL values s1, s2, ... are specified for attributes a1, a2, ... in one object filter. An object o is said to satisfy this object filter if (o.a1 like s1) and (o.a2 like s2) and ... is true. See Applying an Object Filter List.

Table 197-106 (Cont.) REPORT_GATHER_SCHEMA_STATS Function Parameters

Parameter	Description
`detail_level`	Detail level for the content of the report • `BASIC`: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • `TYPICAL`: In addition to the information provided at level `BASIC`, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed-out objects (applies to only auto statistics gathering) With regard to target objects: - owner and name of each target object - target object type (such as: table, index) - start time - end time - completion status • `ALL`: In addition to the information provided at level `TYPICAL`, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name - session ID - parameter values - error message if the operation failed With regard to target objects: - job name - batching details - estimated cost - rank in the target list - columns for which histograms were collected - list of collected extended statistics (if any) - additional error details if the task has failed. Note that several fields (such as job name, estimated task cost) in the report are populated only when an operation is executed concurrently (`CONCURRENT` preference is turned on).
`format`	Report format: • XML • HTML • TEXT (Default)

Usage Notes

To invoke this procedure you must be owner of the table, or you need the `ANALYZE ANY` privilege. For objects owned by `SYS`, you need to be either the owner of the table, or you need the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

Exceptions

`ORA-20000`: Schema does not exist or insufficient privileges

`ORA-20001`: Bad input value

Examples

Applying an Object Filter List

The following example specifies that any table with a "`T`" prefix in the `SAMPLE` schema or any table in the `HR` schema, if stale, will have statistics gathered upon it.

```
DECLARE
  filter_lst  DBMS_STATS.OBJECTTAB := DBMS_STATS.OBJECTTAB();
BEGIN
  filter_lst.extend(2);
  filter_lst(1).ownname := 'SAMPLE';
  filter_lst(1).objname := 'T%';
  filter_lst(2).ownname := 'HR';
  DBMS_STATS.GATHER_SCHEMA_STATS(NULL, obj_filter_list => filter_lst,
                                 options => 'GATHER STALE');
END;
```

REPORT_GATHER_TABLE_STATS Function

This procedure runs the GATHER_TABLE_STATS Procedure in reporting mode.

That is, statistics are not actually collected, but all the objects that will be affected when `GATHER_TABLE_STATS` is invoked are reported.

Syntax

```
DBMS_STATS.REPORT_GATHER_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT to_estimate_percent_type
                                        (get_param('ESTIMATE_PERCENT')),
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT get_param('METHOD_OPT'),
   degree           NUMBER   DEFAULT to_degree_type(get_param('DEGREE')),
   granularity      VARCHAR2 DEFAULT GET_PARAM('GRANULARITY'),
   cascade          BOOLEAN  DEFAULT to_cascade_type(get_param('CASCADE')),
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT  to_no_invalidate_type (
                                 get_param('NO_INVALIDATE')),
   stattype         VARCHAR2 DEFAULT 'DATA',
   force            BOOLEAN  DEFAULT FALSE)
   detail_level     VARCHAR2 DEFAULT 'TYPICAL',   format          VARCHAR2 DEFAULT
'TEXT')
 RETURN CLOB;
```

Parameters

Table 197-107 REPORT_GATHER_TABLE_STATS Function Parameters

Parameter	Description
ownname	Schema of table to analyze
tabname	Name of table
partname	Name of partition
estimate_percent	Percentage of rows to estimate (NULL means compute) The valid range is [0.000001,100]. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to have Oracle determine the appropriate sample size for good statistics. This is the default.The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
block_sample	Whether or not to use random block sampling instead of random row sampling. Random block sampling is more efficient, but if the data is not randomly distributed on disk, then the sample values may be somewhat correlated. Only pertinent when doing an estimate statistics.
method_opt	Accepts either of the following options, or both in combination: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] • FOR COLUMNS [column_clause] [size_clause] size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} column_clause is defined as column_clause := column_name | extension name | extension – integer : Number of histogram buckets. Must be in the range [1,2048]. – REPEAT : Collects histograms only on the columns that already have histograms – AUTO : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns. – SKEWONLY : Oracle determines the columns on which to collect histograms based on the data distribution of the columns. - column_name : Name of a column - extension : can be either a column group in the format of (column_name, Colume_name [, ...]) or an expression The default is FOR ALL COLUMNS SIZE AUTO. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Table 197-107 (Cont.) REPORT_GATHER_TABLE_STATS Function Parameters

Parameter	Description
degree	Degree of parallelism. The default for degree is NULL. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure. NULL means use the table default value specified by the DEGREE clause in the CREATE TABLE or ALTER TABLE statement. Use the constant DBMS_STATS.DEFAULT_DEGREE to specify the default value based on the initialization parameters. The AUTO_DEGREE value determines the degree of parallelism automatically. This is between 1 (serial execution) and DEFAULT_DEGREE (the system default value based on number of CPUs and initialization parameters) according to the size of the object. When using DEGREE=>NULL, DEGREE=>n, or DEGREE=>DBMS_STATS.DEFAULT_DEGREE, the current implementation of DBMS_STATS may use serial execution if the size of the object does not warrant parallel execution.
granularity	Granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'APPROX_GLOBAL AND PARTITION' - similar to 'GLOBAL AND PARTITION' but in this case the global statistics are aggregated from partition level statistics. This option will aggregate all statistics except the number of distinct values for columns and number of distinct keys of indexes. The existing histograms of the columns at the table level are also aggregated.The aggregation will use only partitions with statistics, so to get accurate global statistics, users should make sure to have statistics for all partitions. Global statistics are gathered if partname is NULL or if the aggregation cannot be performed (for example, if statistics for one of the partitions is missing). 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION '- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics.
cascade	Gathers statistics on the indexes for this table. Using this option is equivalent to running the GATHER_INDEX_STATS Procedure on each of the table's indexes. Use the constant DBMS_STATS.AUTO_CASCADE to have Oracle determine whether index statistics are to be collected or not. This is the default. The default value can be changed using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
stattab	User statistics table identifier describing where to save the current statistics
statid	Identifier (optional) to associate with these statistics within stattab
statown	Schema containing stattab (if different than ownname)

Table 197-107 (Cont.) REPORT_GATHER_TABLE_STATS Function Parameters

Parameter	Description
`no_invalidate`	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • `TRUE`: Dependent cursors are not invalidated. • `FALSE`: Dependent cursors are marked for immediate invalidation. • `AUTO`: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
`stattype`	Statistics type. The only value allowed is `DATA`.
`force`	Gather statistics of table even if it is locked

Table 197-107 (Cont.) REPORT_GATHER_TABLE_STATS Function Parameters

Parameter	Description
`detail_level`	Detail level for the content of the report • `BASIC`: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • `TYPICAL`: In addition to the information provided at level `BASIC`, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed-out objects (applies to only auto statistics gathering) With regard to target objects: - owner and name of each target object - target object type (such as: table, index) - start time - end time - completion status • `ALL`: In addition to the information provided at level `TYPICAL`, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name - session ID - parameter values - error message if the operation failed With regard to target objects: - job name - batching details - estimated cost - rank in the target list - columns for which histograms were collected - list of collected extended statistics (if any) - additional error details if the task has failed. Note that several fields (such as job name, estimated task cost) in the report are populated only when an operation is executed concurrently (`CONCURRENT` preference is turned on).
`format`	Report format: • XML • HTML • TEXT (Default)

Return Values

A `CLOB` object that contains the report

Usage Notes

To invoke this procedure you must be owner of the table, or you need the `ANALYZE ANY` privilege. For objects owned by `SYS`, you need to be either the owner of the table, or you need the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

Related Topics

- GATHER_TABLE_STATS Procedure
  This procedure gathers table, column, and index statistics. It attempts to parallelize as much work as possible, but there are some restrictions, which are described in the individual parameters.

REPORT_SINGLE_STATS_OPERATION Function

This function generates a report for the provided operation optionally in a particular pluggable database (PDB) in a multitenant environment.

> **Note:**
>
> A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

Syntax

```
DBMS_STATS.REPORT_SINGLE_STATS_OPERATIONS (
   opid              NUMBER,
   detail_level      VARCHAR2  DEFAULT 'TYPICAL',
   format            VARCHAR2  DEFAULT 'TEXT'
   container_id      NUMBER    DEFAULT NULL)
 RETURN CLOB;
```

Parameters

Table 197-108 REPORT_SINGLE_STATS_OPERATION Function Parameters

Parameter	Description
`opid`	Operation ID

Table 197-108 (Cont.) REPORT_SINGLE_STATS_OPERATION Function Parameters

Parameter	Description
detail_level	Detail level for the content of the report • BASIC: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • TYPICAL: In addition to the information provided at level BASIC, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed-out objects (applies to only auto statistics gathering) With regard to target objects: - owner and name of each target object - target object type (such as: table, index) - start time - end time - completion status • ALL: In addition to the information provided at level TYPICAL, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name - session ID - parameter values - error message if the operation failed With regard to target objects: - job name - batching details - estimated cost - rank in the target list - columns for which histograms were collected - list of collected extended statistics (if any) - reason for including the object in the target list (applies to only automatic statistics gathering operation tasks) - additional error details if the task has failed. Note that several fields (such as job name, estimated task cost) in the report are populated only when an operation is executed concurrently (CONCURRENT preference is turned on).

Table 197-108 (Cont.) REPORT_SINGLE_STATS_OPERATION Function Parameters

Parameter	Description
format	Report format: • XML • HTML • TEXT (Default)
container_id	ID of the pluggable database (PDB) on which this operation was performed. Note that in a multitenant environment, operation ID does not uniquely identify an operation. That is, different operations from distinct PDBs may have the same operation ID. Hence, in a multitenant environment, if a PDB ID is not provided, then the report may contain multiple operations. In a typical (non-CDB) database environment, operation ID is unique to each operation.

Usage Notes

To invoke this procedure you need the ANALYZE ANY privilege and the ANALYZE ANY DICTIONARY privilege.

REPORT_STATS_OPERATIONS Function

This function generates a report of all statistics operations that take place between two timestamps which may or may not have been provided.

It allows the scope of the report to be narrowed down so that report will include only auto statistics gathering runs. Furthermore, in a multitenant environment, users may optionally provide a set of pluggable database (PDB) IDs so that only statistics operations from the specified pluggable databases will be reported.

Syntax

```
DBMS_STATS.REPORT_STATS_OPERATIONS (
   detail_level       VARCHAR2 DEFAULT 'TYPICAL',
   format             VARCHAR2 DEFAULT 'TEXT',
   latestN            NUMBER DEFAULT NULL,
   since              TIMESTAMP WITH TIME ZONE DEFAULT NULL,
   until              TIMESTAMP WITH TIME ZONE DEFAULT NULl,
   auto_only          BOOLEAN DEFAULT FALSE,
   container_ids      DBMS_UTILITY.NUMBER_ARRAY DEFAULT DBMS_STATS.NULL_NUMTAB)
 RETURN CLOB;
```

Parameters

Table 197-109 REPORT_STATS_OPERATIONS Function Parameters

Parameter	Description
detail_level	Detail level for the content of the report • BASIC: The report includes - operation ID - operation name - operation target object - start time - end time - completion status (such as: succeeded, failed) • TYPICAL: In addition to the information provided at level BASIC, the report includes individual target objects for which statistics are gathered in this operation. Specifically, with regard to operation related details: - total number of target objects - total number of successfully completed objects - total number of failed objects - total number of timed -out objects (applies to only auto statistics gathering) • ALL: In addition to the information provided at level TYPICAL, the report includes further information on each target object. Specifically, with regard to operation-related details: - job name (if the operation was run in a job) - session ID - parameter values - additional error details if the operation has failed
format	Report format: • XML • HTML • TEXT (Default)
latestN	Restricts the report to contain only the latest N operations that took place between the provided time points (since and until). The default value is NULL, meaning that all qualifying operations will be reported.
since	The report will include only statistics operations that started after this timestamp.
until	The report will include only statistics operations that before after this timestamp.
auto_only	When TRUE, the report will contain only auto statistics gathering job runs.
container_ids	A multitenant environment contains one or more pluggable databases (PDBs). container_ids represents a set of PDB IDs so that only statistics operations from the specified PDBs are reported (applies to only multitenant environments).

Usage Notes

To invoke this procedure you need the ANALYZE ANY privilege and the ANALYZE ANY DICTIONARY privilege.

Examples

Note that the type for `container_ids` input parameter is `DBMS_UTILITY.NUMBER_ARRAY` which is an associative PL/SQL array collection. Although associative array type allows for more flexible `harvals` table-like organization of entries, this function treats `container_ids` as a regular table collection with the first ID located at index 1 and the last id located at index `container_ids.count` without any empty array slot left between any two IDs. An example for 3 container ids is provided.

```
DECLARE
    conid_tab  DBMS_UTILITY.NUMBER_ARRAY;
    report clob;
BEGIN
    conid_tab(1) := 124;
    conid_tab(2) := 63;
    conid_tab(3) := 98;
    report := DBMS_STATS.REPORT_STATS_OPERATIONS (container_ids => conid_tab);
END;
```

RESET_ADVISOR_TASK Procedure

This procedure resets an Optimizer Statistics Advisor task execution to its initial state. Only reset a task that is not currently executing.

Syntax

```
DBMS_STATS.RESET_ADVISOR_TASK (
  task_name IN VARCHAR2);
```

Parameters

Table 197-110 RESET_ADVISOR_TASK Procedure Parameters

Parameter	Description
`task_name`	The name of the Optimizer Statistics Advisor task.

Security Model

Note the following:

- To execute this subprogram, you must have the `ADVISOR` privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Exceptions

- `ORA-20000`: Insufficient privileges
- `ORA-20001`: Invalid input values
- `ORA-20012`: Optimizer Statistics Advisor errors

RESET_COL_USAGE Procedure

This procedure deletes the recorded column (group) usage information.

This procedure should only be used in very rare cases when the seed column usage needs to be initialized.

Syntax

```
DBMS_STATS.RESET_COL_USAGE (
   ownname     IN    VARCHAR2,
   tabname     IN    VARCHAR2);
```

Parameters

Table 197-111 RESET_COL_USAGE Procedure Parameters

Parameter	Description
ownname	Owner name. If NULL it deletes column usage information for tables in all schemas in the database.
tabname	Table name. If NULL it deletes column usage information for all tables of ownname. If both the owner and tabname is NULL, the seed column usage is stopped if applicable. See :SEED_COL_USAGE Procedure for more information.

Usage Notes

To run this procedure, you need to have the SYSDBA administrative privilege, or both the ANALYZE ANY DICTIONARY and the ANALYZE ANY system privileges.

RESET_GLOBAL_PLSQL_PREFS_DEF Procedure

Resets PL/SQL global preferences to their default values.

Syntax

```
EXEC DBMS_STATS.RESET_GLOBAL_PLSQL_PREF_DEFAULTS
```

Parameters

None.

RESET_PARAM_DEFAULTS Procedure

This deprecated procedure resets the default values of all parameters to Oracle recommended values.

> **Note:**
>
> This subprogram has been replaced by improved technology and is maintained only for purposes of backward compatibility.
>
> See also . DBMS_STATS Deprecated Subprograms

Syntax

```
DBMS_STATS.RESET_PARAM_DEFAULTS;
```

RESTORE_DATABASE_STATS Procedure

This procedure restores statistics of all tables of the database as of a specified timestamp (`as_of_timestamp`).

Syntax

```
DBMS_STATS.RESTORE_DATABSE_STATS(
   as_of_timestamp        TIMESTAMP WITH TIME ZONE,
   force                  BOOLEAN DEFAULT FALSE,
   no_invalidate          BOOLEAN DEFAULT to_no_invalidate_type
                                              (GET_PARAM('NO_INVALIDATE')));
```

Parameters

Table 197-112 RESTORE_DATABASE_STATS Procedure Parameters

Parameter	Description
`as_of_timestamp`	The timestamp to which to restore statistics
`force`	Restores statistics even if their statistics are locked
`no_invalidate`	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • `TRUE`: Dependent cursors are not invalidated. • `FALSE`: Dependent cursors are marked for immediate invalidation. • `AUTO`: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

ORA-20006: Unable to restore statistics, statistics history not available

RESTORE_DICTIONARY_STATS Procedure

This procedure restores statistics of all dictionary tables (tables of 'SYS', 'SYSTEM' and RDBMS component schemas) as of a specified timestamp (as_of_timestamp).

Syntax

```
DBMS_STATS.RESTORE_DICTIONARY_STATS(
   as_of_timestamp        TIMESTAMP WITH TIME ZONE,
   force                  BOOLEAN DEFAULT FALSE,
   no_invalidate          BOOLEAN DEFAULT to_no_invalidate_type
                                              (GET_PARAM('NO_INVALIDATE')));
```

Parameters

Table 197-113 RESTORE_DICTIONARY_STATS Procedure Parameters

Parameter	Description
as_of_timestamp	Timestamp to which to restore statistics
force	Restores statistics even if their statistics are locked
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Usage Notes

To run this procedure, you must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

ORA-20006: Unable to restore statistics, statistics history not available

RESTORE_FIXED_OBJECTS_STATS Procedure

This procedure restores statistics of all fixed tables as of a specified timestamp (as_of_timestamp).

Syntax

```
DBMS_STATS.RESTORE_FIXED_OBJECTS_STATS(
   as_of_timestamp        TIMESTAMP WITH TIME ZONE,
   force                  BOOLEAN DEFAULT FALSE,
   no_invalidate          BOOLEAN DEFAULT to_no_invalidate_type
                                            (GET_PARAM('NO_INVALIDATE')));
```

Parameters

Table 197-114 RESTORE_FIXED_OBJECTS_STATS Procedure Parameters

Parameter	Description
as_of_timestamp	The timestamp to which to restore statistics
force	Restores statistics even if their statistics are locked
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Usage Notes

To run this procedure, you must have the SYSDBA or ANALYZE ANY DICTIONARY system privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

ORA-20006: Unable to restore statistics, statistics history not available

RESTORE_SCHEMA_STATS Procedure

This procedure restores statistics of all tables of a schema as of a specified timestamp (as_of_timestamp).

Syntax

```
DBMS_STATS.RESTORE_SCHEMA_STATS(
   ownname                VARCHAR2,
   as_of_timestamp        TIMESTAMP WITH TIME ZONE,
   force                  BOOLEAN DEFAULT FALSE,
```

```
   no_invalidate          BOOLEAN DEFAULT to_no_invalidate_type
                                                 (GET_PARAM('NO_INVALIDATE')));
```

Parameters

Table 197-115 RESTORE_SCHEMA_STATS Procedure Parameters

Parameter	Description
ownname	Schema of the tables for which the statistics are to be restored
as_of_timestamp	The timestamp to which to restore statistics
force	Restores statistics even if their statistics are locked
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

ORA-20006: Unable to restore statistics, statistics history not available

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

RESTORE_SYSTEM_STATS Procedure

This procedure restores system statistics as of a specified timestamp (as_of_timestamp).

Syntax

```
DBMS_STATS.RESTORE_SCHEMA_STATS(
   as_of_timestamp        TIMESTAMP WITH TIME ZONE);
```

Parameters

Table 197-116 RESTORE_SYSTEM_STATS Procedure Parameters

Parameter	Description
as_of_timestamp	The timestamp to which to restore statistics

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

ORA-20006: Unable to restore statistics, statistics history not available

Usage Notes

To run this procedure, you need the GATHER_SYSTEM_STATISTICS role.

RESTORE_TABLE_STATS Procedure

This procedure restores statistics of a table as of a specified timestamp (as_of_timestamp). It also restores statistics of associated indexes and columns.

If the table statistics were locked at the specified timestamp the procedure will lock the statistics. The procedure will not restore user defined statistics.

Syntax

```
DBMS_STATS.RESTORE_TABLE_STATS (
   ownname                  VARCHAR2,
   tabname                  VARCHAR2,
   as_of_timestamp          TIMESTAMP WITH TIME ZONE,
   restore_cluster_index    BOOLEAN DEFAULT FALSE,
   force                    BOOLEAN DEFAULT FALSE,
   no_invalidate            BOOLEAN DEFAULT to_no_invalidate_type
                                          (GET_PARAM('NO_INVALIDATE')));
```

Parameters

Table 197-117 RESTORE_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	The schema of the table for which the statistics are to be restored
tabname	The table name
as_of_timestamp	The timestamp to which to restore statistics
restore_cluster_index	If the table is part of a cluster, restore statistics of the cluster index if set to TRUE
force	Restores statistics even if the table statistics are locked. If the table statistics were not locked at the specified timestamp, it unlocks the statistics.
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent values

ORA-20006: Unable to restore statistics, statistics history not available

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

RESUME_ADVISOR_TASK Procedure

This procedure resumes an interrupted task. It only resumes the execution that was most recently interrupted.

Syntax

```
DBMS_STATS.RESUME_ADVISOR_TASK (
  task_name IN VARCHAR2);
```

Parameters

Table 197-118 RESUME_ADVISOR_TASK Procedure Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Consider a case in which a task is executed by one user, interrupted, and then resumed by a different user. In this case, Optimizer Statistics Advisor bases its checks of the resumed execution on the privilege of the user who resumed the task.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Example 197-14 Resuming an Interrupted Task

In this example, you start a SQL*Plus session, and then create and execute an advisor task named `my_task`:

```
DECLARE
  v_tname   VARCHAR2(128) := 'my_task';
  v_ename   VARCHAR2(128) := NULL;
BEGIN
  -- create a task
  v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);

  -- execute the task
  v_ename := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
END;
/
```

In a separate terminal, you start a second SQL*Plus session, and then execute the following program:

```
XEC DBMS_STATS.INTERRUPT_ADVISOR_TASK('my_task');
```

The first session returns an `ORA-13632` to indicate the cancelation of the task:

```
ORA-13638: The user interrupted the current operation.
```

In the second SQL*Plus session, you resume the task execution as follows:

```
XEC DBMS_STATS.RESUME_ADVISOR_TASK('my_task');
```

SCRIPT_ADVISOR_TASK Function

Retrieves the script that implements the recommended actions for the problems found by Optimizer Statistics Advisor.

The generated script contains PL/SQL statements that you can choose to execute. Preceding the commands for each action are comments that list the potential side effects. You can review the comments, and choose to execute only the desired sections.

Syntax

```
DBMS_STATS.SCRIPT_ADVISOR_TASK (
  task_name          IN   VARCHAR2,
  execution_name     IN   VARCHAR2    := NULL,
  dir_name           IN   VARCHAR2    := NULL,
  level              IN   VARCHAR2    := 'TYPICAL')
RETURN CLOB;
```

Parameters

Table 197-119 SCRIPT_ADVISOR_TASK Function Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.
execution_name	A name that qualifies and identifies an advisor execution. If not specified, then the advisor automatically generates it. If the specified execution conflicts with the name of an existing execution, then the function returns an error.
dir_name	Directory name to which to write the generated script. If the name is not specified (NULL), then the function includes the script in the returned CLOB. If the name is specified, then the function returns the script as a CLOB and as a new file in the specified directory.
level	The level of the script to generate. Possible values are • ALL: Ignores the filter and generates a script for all findings • TYPICAL: Generates a script according to the filters in place

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- You can execute this subprogram for AUTO_STATS_ADVISOR_TASK, which is predefined.
- This subprogram executes using invoker's rights.

The results of performing this task depend on the privileges of the executing user:

- SYSTEM level

  Only users with both the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges can perform this task on system-level rules.

- Operation level

  The results depend on the following privileges:

  – Users with both the ANALYZE ANY and ANALYZE ANY DICTIONARY privileges can perform this task for all statistics operations.

  – Users with the ANALYZE ANY privilege but *not* the ANALYZE ANY DICTIONARY privilege can perform this task for statistics operations related to any schema except SYS.

  – Users with the ANALYZE ANY DICTIONARY privilege but *not* the ANALYZE ANY privilege can perform this task for statistics operations related to their own schema and the SYS schema.

  – Users with neither the ANALYZE ANY nor the ANALYZE ANY DICTIONARY privilege can only perform this operation for statistics operations relating to their own schema.

- Object level

  Users can perform this task for any object for which they have statistics collection privileges.

Return Values

This function returns a CLOB that contains the script.

Exceptions

- ORA-20000: Insufficient privileges
- ORA-20001: Invalid input values
- ORA-20012: Optimizer Statistics Advisor errors

Example 197-15 Creating an Optimizer Statistics Advisor Script

This example creates a procedure named myrep, and then calls this procedure to print the script the implements the recommendations.

```
SET ECHO ON
SET FEEDBACK ON
SET SERVEROUTPUT ON
SET TRIMS ON
SET LINESIZE 300

EXECUTE DBMS_OUTPUT.ENABLE (buffer_size => 10000000);

CREATE OR REPLACE PROCEDURE myrep(p_tname VARCHAR2, p_ftype VARCHAR2, which
VARCHAR2)
IS
  v_report CLOB          := null;
  v_script CLOB          := null;
  v_ftype  VARCHAR2(400) := p_ftype;
  v_tname  VARCHAR2(400) := p_tname;
  v_len    NUMBER(10);
  v_ps     NUMBER(10)    := 10000;
  v_pn     NUMBER(10)    := 1;
  v_ret    VARCHAR2(32767);
BEGIN
   IF which = 'REPORT'
   THEN
      -- generate a report
      v_report := DBMS_STATS.REPORT_ADVISOR_TASK(
                    task_name => v_tname,
                    type      => v_ftype,
                    section   => 'ALL',
                    level     => 'ALL');
      v_len := DBMS_LOB.getlength(v_report);
      WHILE (v_pn < v_len)
      LOOP
        DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBSTR(v_report, v_ps, v_pn));
        v_pn := v_pn + v_ps;
      END LOOP;
   ELSE
      -- generate a script
      v_script := DBMS_STATS.SCRIPT_ADVISOR_TASK(v_tname);
      v_len := DBMS_LOB.getlength(v_script);
      WHILE (v_pn < v_len)
      LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBSTR(v_script, v_ps, v_pn));
        v_pn := v_pn + v_ps;
      END LOOP;
   END IF;
END;
/

SHOW ERRORS

SPOOL report.txt
EXECUTE myrep('my_task','-','SCRIPT');
SPOOL OFF
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide*

SEED_COL_USAGE Procedure

This procedure seeds column usage information from a statements in the specified SQL tuning set, or in the database.

The procedure iterates over the SQL statements, compiles them, and then seeds column usage information for the columns that appear in these statements. You can monitor the workload on the system for given amount of time and seed the and seed the column usage information based on the columns that appear in statements executed during the monitoring window.

Syntax

```
DBMS_STATS.SEED_COL_USAGE (
   sqlset_name     IN    VARCHAR2,
   owner_name      IN    VARCHAR2,
   time_limit      IN    POSITIVE DEFAULT NULL);
```

Parameters

Table 197-120 SEED_COL_USAGE Procedure Parameters

Parameter	Description
`sqlset_name`	Name of the SQL tuning set that contains the statements to be monitored. If this parameter and `owner_name` are both null, then the procedure monitors all statements in the database for the specified time limit.
`owner_name`	Owner of the SQL tuning set that contains the statements to be monitored. If this parameter and `sqlset_name` are both null, then the procedure monitors all statements in the database for the specified time limit.
`time_limit`	Time limit (in seconds).

Security Model

To invoke this procedure you must have the `ANALYZE ANY` privilege and the `ANALYZE ANY DICTIONARY` privilege.

Exceptions

ORA-20000: Insufficient privileges

Usage Notes

This procedure also records group of columns. You can create extensions for the recorded group of columns using the CREATE_EXTENDED_STATS Function procedure. If sqlset_name and owner_name are NULL, then the procedure records the column (group) usage information for the statements executed in the system in next time_limit seconds.

This monitoring procedure records different information from the traditional column usage information that is visible in SYS.COL_USAGE$. The procedure stores information in SYS.COL_GROUP_USAGE$.

Examples

The following example turns on monitoring for 5 minutes or 300 seconds.

```
BEGIN
   DBMS_STATS.SEED_COL_USAGE (null,null,300);
END;
```

SET_ADVISOR_TASK_PARAMETER Procedure

This procedure updates the value of an Optimizer Statistics Advisor task parameter.

Syntax

```
DBMS_STATS.SET_ADVISOR_TASK_PARAMETER (
  task_name     IN   VARCHAR2,
  parameter     IN   VARCHAR2,
  value         IN   VARCHAR2);
```

Parameters

Table 197-121 SET_ADVISOR_TASK_PARAMETER Procedure Parameters

Parameter	Description
task_name	The name of the Optimizer Statistics Advisor task.
parameter	The name of the parameter to set. The function returns an error if the specified parameter does not exist.
value	The new value of the parameter.

Security Model

Note the following:

- To execute this subprogram, you must have the ADVISOR privilege.
- You must be the owner of the task.
- This subprogram executes using invoker's rights.

Exceptions

- ORA-20000: Insufficient privileges

- `ORA-20001`: Invalid input values
- `ORA-20012`: Optimizer Statistics Advisor errors

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to manage Optimizer Statistics Advisor

SET_COLUMN_STATS Procedures

This procedure sets column-related information.

In the version of this procedure that deals with user-defined statistics, the statistics type specified is the type to store in the dictionary, in addition to the actual user-defined statistics. If this statistics type is `NULL`, the statistics type associated with the index or column is stored.

Syntax

```
DBMS_STATS.SET_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   distcnt       NUMBER DEFAULT NULL,
   density       NUMBER DEFAULT NULL,
   nullcnt       NUMBER DEFAULT NULL,
   srec          StatRec DEFAULT NULL,
   avgclen       NUMBER DEFAULT NULL,
   flags         NUMBER DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT to_no_invalidate_type(
                                 get_param('NO_INVALIDATE')),
   force         BOOLEAN DEFAULT FALSE);
```

Use the following for user-defined statistics:

```
DBMS_STATS.SET_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   ext_stats     RAW,
   stattypown    VARCHAR2 DEFAULT NULL,
   stattypname   VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT to_no_invalidate_type(
                                 get_param('NO_INVALIDATE')),
   force         BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-122 SET_COLUMN_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema.
tabname	Name of the table to which this column belongs.
colname	Name of the column or extension
partname	Name of the table partition in which to store the statistics. If the table is partitioned and partname is NULL, then the statistics are stored at the global table level.
stattab	User statistics table identifier describing where to store the statistics. If stattab is NULL, then the statistics are stored directly in the dictionary.
statid	Identifier (optional) to associate with these statistics within stattab (Only pertinent if stattab is not NULL)
ext_stats	User-defined statistics
stattypown	Schema of the statistics type
stattypname	Name of the statistics type
distcnt	Number of distinct values
density	Column density. If this value is NULL and if distcnt is not NULL, then density is derived from distcnt.
nullcnt	Number of NULLs
srec	StatRec structure filled in by a call to PREPARE_COLUMN_VALUES or GET_COLUMN_STATS
avgclen	Average length for the column (in bytes)
flags	For internal Oracle use (should be left as NULL)
statown	Schema containing stattab (if different than ownname)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
force	Sets the values even if statistics of the column are locked

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or inconsistent input values

ORA-20005: Object statistics are locked

Usage Notes

To invoke this procedure you must be owner of the table, or you need the `ANALYZE ANY` privilege. For objects owned by `SYS`, you need to be either the owner of the table, or you need the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

SET_DATABASE_PREFS Procedure

This procedure sets the statistics preferences of all the tables, excluding the tables owned by the database. These tables can by included by passing `TRUE` for the `add_sys` parameter.

Syntax

```
DBMS_STATS.SET_DATABASE_PREFS (
    pname           IN   VARCHAR2,
    pvalue          IN   VARCHAR2,
    add_sys         IN   BOOLEAN DEFAULT FALSE);
```

Parameters

Table 197-123 SET_DATABASE_PREFS Procedure Parameters

Parameter	Description
`pname`	Preference name. The existing value for following preferences can be set and default preference values will be used: • `APPROXIMATE_NDV_ALGORITHM` • `AUTO_STAT_EXTENSIONS` • `CASCADE` • `DEGREE` • `ESTIMATE_PERCENT` • `GLOBAL_TEMP_TABLE_STATS` • `GRANULARITY` • `INCREMENTAL` • `INCREMENTAL_STALENESS` • `INCREMENTAL_LEVEL` • `METHOD_OPT` • `NO_INVALIDATE` • `OPTIONS` • `PREFERENCE_OVERRIDES_PARAMETER` • `PUBLISH` • `STALE_PERCENT` • `STAT_CATEGORY` • `TABLE_CACHED_BLOCKS`
`pvalue`	Preference value. If `NULL` is specified, it will set the Oracle default values
`add_sys`	Value `TRUE` will include the Oracle-owned tables

Table 197-124 Statistics Preferences

Preference	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. Consider a synopsis as an internal management structure that samples distinct values. You can specify the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
CASCADE	Determines whether to collect index statistics as part of gathering table statistics.
DEGREE	Determines the degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	Determines the percentage of rows to sample. The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for optimal statistics. This is the default.

Table 197-124 (Cont.) Statistics Preferences

Preference	Description
GLOBAL_TEMP_TABLE_STATS	Controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics. This preference takes two values: • SHARED — All sessions see the same set of statistics • SESSION — Statistics gathered by the GATHER_TABLE_STATS procedure on a global temporary table are session-specific. Thus, the database only uses them for queries issued in the same session as the statistics gathering process. The database deletes session-specific statistics when a session terminates.
GRANULARITY	Determines the granularity of statistics to collect. This preference is only relevant for partitioned tables. The following values are valid: • ALL — Gathers all statistics: subpartition, partition, and global. • AUTO — Determines the granularity based on the partitioning type. This is the default value. • DEFAULT — Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. Use GLOBAL AND PARTITION for this functionality. • GLOBAL — Gathers global statistics. • GLOBAL AND PARTITION — Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. • PARTITION — Gathers partition-level statistics. • SUBPARTITION — Gathers subpartition-level statistics.
INCREMENTAL	Determines whether the global statistics for a partitioned table are maintained without performing a full table scan. When a table is partitioned, an application typically loads data into a new partition. As new partitions are added and data is loaded, global table statistics must be kept up to date. If the following conditions are met, then the database updates the global table statistics by scanning only the changed partitions instead of the entire table: • The INCREMENTAL value for the partitioned table is set to TRUE. • The PUBLISH value for the partitioned table is set to TRUE. • The user specifies AUTO_SAMPLE_SIZE for ESTIMATE_PERCENT and AUTO for GRANULARITY when gathering statistics on the table. If the INCREMENTAL value for the partitioned table was set to FALSE (default value), then the database uses a full table scan to maintain the global statistics. This technique is a much more resource-intensive and time-consuming operation for large tables.

Table 197-124 (Cont.) Statistics Preferences

Preference	Description
INCREMENTAL_LEVEL	Controls which synopses to collect when INCREMENTAL preference is set to TRUE. It takes the following values: • PARTITION — Gathers partition-level synopses. This is the default value. If PARTITION is set on a nonpartitioned table, then the database does not gather synopses. • TABLE — Gathers table-level synopses. Specify this value when you want to exchange this table with a partition. Before the exchange, you can run GATHER_TABLE_STATS on this table with INCREMENTAL set to TRUE and INCREMENTAL_LEVEL to TABLE. The result is that the database gathers table-level synopses on this table. After the exchange, the partition has synopses that come from the table-level synopses of the table before the exchange. You can only use preference value in the SET_TABLE_PREFS procedure: this value is not allowed in the other SET_*_PREFS procedures.

Table 197-124 (Cont.) Statistics Preferences

Preference	Description

INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'.
	The parameter accepts the following values:
	• USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference.
	For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics.
	Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics.
	• USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes.
	For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics.
	• ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist.
	• NULL—A partition or subpartition is considered stale when it has any DML changes.
	For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.
	Note that the following two executions are different:
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	The first execution uses single quotes to set the preference to the value NULL, whereas the second sets the preference to the default, which is ALLOW_MIXED_FORMAT.

Table 197-124 (Cont.) Statistics Preferences

Preference	Description
METHOD_OPT	Controls column statistics collection and histogram creation. When setting preferences at the global, schema, database, or dictionary level, only FOR ALL syntax is allowed: • FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] The size_clause is defined as size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} integer — Specifies the number of histogram buckets. The number must be between 1 and 2048. REPEAT — Collects histograms only on the columns that already have histograms. AUTO — Determines the columns on which to collect histograms based on data distribution and the workload of the columns. SKEWONLY — Determines the columns on which to collect histograms based on the data distribution of the columns. The default is FOR ALL COLUMNS SIZE AUTO. You can change the value using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
NO_INVALIDATE	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. If set to TRUE, then the database not invalidate dependent cursors. If set to FALSE, then the procedure invalidates dependent cursors immediately.
OPTIONS	Specifies which objects require statistics to be gathered. Valid values are as follows: • GATHER — Gathers statistics on all objects in the database. This is the default. • GATHER AUTO — Gathers all necessary statistics automatically. This is the default. The database implicitly determines which objects need new statistics and determines how to gather those statistics. When GATHER AUTO is specified, the only additional valid parameters are comp_id, no_invalidate, stattab, statid, and statown; all other parameter settings will be ignored. Also, the database returns a list of objects processed.

Table 197-124 (Cont.) Statistics Preferences

Preference	Description
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether the database publishes newly gathered statistics after the gathering job completes. You can gather statistics without publishing them immediately. This technique enables you to test new statistics before publishing them.
STALE_PERCENT	Determines the percentage of rows in a table that must change before the statistics on that table are stale and need to be regathered. The valid domain for stale_percent is non-negative numbers. The default value is 10, which means that a table having more than 10% of changes is considered stale.
STAT_CATEGORY	Specifies which statistics to import or export, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics (default) • SYNOPSES — information to support incremental statistics • REALTIME_STATS — specifies only real-time statistics. The value 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.

Security Model

To run this procedure, you must have the SYSDBA role or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privileges.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Invalid or illegal input values

Usage Notes

Both arguments are of type VARCHAR2 and values are enclosed in quotes, even when they represent numbers.

Examples

```
DBMS_STATS.SET_DATABASE_PREFS('CASCADE', 'DBMS_STATS.AUTO_CASCADE');
DBMS_STATS.SET_DATABASE_PREFS('ESTIMATE_PERCENT','9');
DBMS_STATS.SET_DATABASE_PREFS('DEGREE','99');
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set optimizer statistics preferences

SET_GLOBAL_PLSQL_PREFS Procedure

Sets the global preference for PL/SQL functions.

By default, the global preference is set to `CHOOSE`.

Global preferences set with SET_GLOBAL_PLSQL_PREFS are persistent across sessions.

Syntax

```
DBMS_STATS.SET_GLOBAL_PLSQL_PREFS(
    pname           IN VARCHAR2,
    pvalue          IN VARCHAR2
);
```

Parameters

Table 197-125 SET_GLOBAL_PLSQL_PREFS Procedure Parameters

Parameter	Description
`pname`	The preference that you want to set. Currently only `dynamic_stats` is accepted.
`pvalue`	The value of the preference.

Table 197-126 PL/SQL Preferences

Preference Name	Description and Possible Values
dynamic_stats	Dynamic sampling is a performance feature that instructs the optimizer to read a sample of data during query compilation to formulate dynamic statistics. Dynamic statistics supplement optimizer statistics such as table and index block counts, table and join cardinalities (estimated number of rows), join column statistics, and GROUP BY statistics. This information helps the optimizer improve plans by making better estimates for predicate cardinality. The preference value controls whether functions can be executed during dynamic sampling to improve cardinality estimates. High-performance PL/SQL is generally suitable for use in dynamic sampling, but you should exclude long-running PL/SQL functions since they will increase SQL compilation times. These are the options for the dynamic_stats preference: • ON - Allow dynamic statistics for PL/SQL functions. • OFF - Disallow dynamic statistics for PL/SQL functions. • CHOOSE - Leave enabling or disabling dynamic statistics to the discretion of the optimizer. This is the default for the dynamic_stats preference. • NULL - the global preference is set to the default.

Examples

Enable dynamic statistics for PL/SQL functions:

```
EXEC DBMS_STATS.SET_PLSQL_GLOBAL_PREFS('dynamic_stats', 'ON')
```

Disable dynamic statistics for PL/SQL functions:

```
EXEC DBMS_STATS.SET_PLSQL_GLOBAL_PREFS('dynamic_stats', 'OFF')
```

Let the optimizer choose whether or not enable the use of dynamic statistics for PL/SQL functions:

```
EXEC DBMS_STATS.SET_PLSQL_GLOBAL_PREFS('dynamic_stats', 'CHOOSE')
EXEC DBMS_STATS.SET_PLSQL_GLOBAL_PREFS('dynamic_stats', NULL)
```

SET_GLOBAL_PREFS Procedure

This procedure sets statistics preferences at the global level.

Syntax

```
DBMS_STATS.SET_GLOBAL_PREFS (
    pname     IN   VARCHAR2,
    pvalue    IN   VARCHAR2);
```

Parameters

Table 197-127 SET_GLOBAL_PREFS Procedure Parameters

Parameter	Description
pname	Preference name. The default value for the following preferences can be set: • APPROXIMATE_NDV_ALGORITHM • AUTO_STAT_EXTENSIONS • AUTO_TASK_STATUS • AUTO_TASK_MAX_RUN_TIME • AUTO_TASK_INTERVAL • AUTOSTATS_TARGET • CASCADE • CONCURRENT • DEGREE • ESTIMATE_PERCENT • GLOBAL_TEMP_TABLE_STATS • GRANULARITY • INCREMENTAL • INCREMENTAL_LEVEL • INCREMENTAL_STALENESS • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • STAT_CATEGORY • TABLE_CACHED_BLOCKS • WAIT_TIME_TO_UPDATE_STATS
pvalue	Preference value. If NULL is specified, it will set the Oracle default values

Table 197-128 Global Statistics Preferences

Preference	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. Consider a synopsis as an internal management structure that samples distinct values. You can specify the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
AUTO_TASK_STATUS	Enables or disables the high-frequency automatic optimizer statistics collection. Values are: • ON — Enables high-frequency automatic optimizer statistics collection. • OFF — Disables high-frequency automatic optimizer statistics collection. This is the default.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description
AUTO_TASK_MAX_RUN_TIME	Configures the maximum runtime in seconds of an execution of the the high-frequency automatic statistics collection. The default is 3600 but varies among Oracle Database platforms.
AUTO_TASK_INTERVAL	Specifies the interval in seconds between executions of high-frequency automatic optimizer statistics collection. The minimum value is 60. The default is 900 (equal to 15 minutes).
AUTOSTATS_TARGET	Controls the objects considered for statistics collection. It takes the following values: • 'ALL' - Statistics collected for all objects in system • 'ORACLE' - Statistics collected for all Oracle owned objects • 'AUTO' - Oracle decides on which objects to collect statistics This preference is applicable only for automatic statistics collection.
CASCADE	Determines whether to collect index statistics as part of gathering table statistics.
CONCURRENT	Determines whether statistics are gathered concurrently on multiple objects, or serially, one object at a time. Valid values are: • MANUAL — Concurrency is enabled only for manual statistics gathering. • AUTOMATIC — Concurrency is enabled only for the automatic statistics gathering. • ALL — Concurrency is enabled for both manual and automatic statistics gathering. • OFF — Concurrency is disabled for both manual and automatic statistics. Note the following guidelines: The CONCURRENT preference determines whether statistics are gathered concurrently when the user issues GATHER_*_STATS procedures. DBMS_STATS can collect statistics for a single object in parallel based on the value of the DEGREE parameter. However, parallelism is limited to one object. The CONCURRENT preference extends the scope of parallelism to multiple database objects. This approach is primarily intended for multi-CPU systems, and may not be suitable for small databases on single-CPU computers. To gather statistics concurrently, Resource Manager must be enabled, and the setting for the JOB_QUEUE_PROCESSES initialization parameter must be at least 4.
DEGREE	Determines the degree of parallelism used for gathering statistics.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description
ESTIMATE_PERCENT	Determines the percentage of rows to sample. The valid range is between 0.000001 and 100. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to enable the database to determine the appropriate sample size for optimal statistics. This is the default.
GLOBAL_TEMP_TABLE_STATS	Controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics. This preference takes two values: • SHARED — All sessions see the same set of statistics • SESSION — Statistics gathered by the GATHER_TABLE_STATS procedure on a global temporary table are session-specific. Thus, the database only uses them for queries issued in the same session as the statistics gathering process. The database deletes session-specific statistics when a session terminates.
GRANULARITY	Determines the granularity of statistics to collect. This preference is only relevant for partitioned tables. The following values are valid: • ALL — Gathers all statistics: subpartition, partition, and global. • AUTO — Determines the granularity based on the partitioning type. This is the default value. • DEFAULT — Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. Use GLOBAL AND PARTITION for this functionality. • GLOBAL — Gathers global statistics. • GLOBAL AND PARTITION — Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. • PARTITION — Gathers partition-level statistics. • SUBPARTITION — Gathers subpartition-level statistics.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description
INCREMENTAL	Determines whether the global statistics for a partitioned table are maintained without performing a full table scan. When a table is partitioned, an application typically loads data into a new partition. As new partitions are added and data is loaded, global table statistics must be kept up to date. If the following conditions are met, then the database updates the global table statistics by scanning only the changed partitions instead of the entire table: • The INCREMENTAL value for the partitioned table is set to TRUE. • The PUBLISH value for the partitioned table is set to TRUE. • The user specifies AUTO_SAMPLE_SIZE for ESTIMATE_PERCENT and AUTO for GRANULARITY when gathering statistics on the table. If the INCREMENTAL value for the partitioned table was set to FALSE (default value), then the database uses a full table scan to maintain the global statistics. This technique is a much more resource-intensive and time-consuming operation for large tables.
INCREMENTAL_LEVEL	Controls which synopses to collect when INCREMENTAL preference is set to TRUE. It takes the following values: • PARTITION — Gathers partition-level synopses. This is the default value. If PARTITION is set on a nonpartitioned table, then the database does not gather synopses. • TABLE — Gathers table-level synopses. Specify this value when you want to exchange this table with a partition. Before the exchange, you can run GATHER_TABLE_STATS on this table with INCREMENTAL set to TRUE and INCREMENTAL_LEVEL to TABLE. The result is that the database gathers table-level synopses on this table. After the exchange, the partition has synopses that come from the table-level synopses of the table before the exchange. You can only use preference value in the SET_TABLE_PREFS procedure: this value is not allowed in the other SET_*_PREFS procedures.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description
INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'. The parameter accepts the following values: • USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference. For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics. Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics. • USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes. For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics. • ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist. • NULL—A partition or subpartition is considered stale when it has any DML changes. For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description

	Note that the following two executions are different:
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	
	The first execution uses single quotes to set the preference to the value `NULL`, whereas the second sets the preference to the default, which is `ALLOW_MIXED_FORMAT`.

`METHOD_OPT`	Controls column statistics collection and histogram creation. It accepts either of the following options, or both in combination: • `FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause]` `size_clause` is defined as `size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY}` – `integer`: Number of histogram buckets. Must be in the range [1,2048]. – `REPEAT` : Collects histograms only on the columns that already have histograms – `AUTO` : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns – `SKEWONLY` : Oracle determines the columns on which to collect histograms based on the data distribution of the columns - `column_name` : name of a column - `extension` : can be either a column group in the format of `(column_name, colume_name [, ...])` or an expression The default is `FOR ALL COLUMNS SIZE AUTO`.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description
NO_INVALIDATE	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. If set to TRUE, then the database not invalidate dependent cursors. If set to FALSE, then the procedure invalidates dependent cursors immediately.
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS procedure. The preference takes the following values: • GATHER — Gathers statistics for all objects in the table. This is the default. • GATHER AUTO — Oracle recommends using GATHER AUTO to gather necessary statistics, such as histograms, after a table has been bulk loaded and acquired online statistics. This is applicable only to tables that are not using INCREMENTAL statistics. For partitioned tables using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO will gather statistics for a table if it is marked stale or has no statistics. In addition, statistics will be gathered for partitions and sub-partitions that are marked stale or have no statistics. For tables not using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO may gather statistics for partitions and sub-partitions, even if they are not marked stale.
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether the database publishes newly gathered statistics after the gathering job completes. You can gather statistics without publishing them immediately. This technique enables you to test new statistics before publishing them.

Table 197-128 (Cont.) Global Statistics Preferences

Preference	Description
STALE_PERCENT	Determines the percentage of rows in a table that must change before the statistics on that table are stale and need to be regathered. The valid domain for stale_percent is non-negative numbers. The default value is 10, which means that a table having more than 10% of changes is considered stale.
STAT_CATEGORY	Specifies which statistics to import or export, accepting multiple values separated by a comma. Values supported: • OBJECT_STATS — table statistics, column statistics and index statistics (default) • SYNOPSES — information to support incremental statistics • REALTIME_STATS — specifies only real-time statistics. The value 'OBJECT_STATS, SYNOPSES' specifies table statistics, column statistics, index statistics, and synopses.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.
WAIT_TIME_TO_UPDATE_STATS	Specifies the number of minutes before timing out for locks and pins required for updating statistics. It accepts values in the range 0 to 65535. The default value is 0.0033 minutes. The value 0 gets the locks and pins in no-wait mode.

Security Model

To run this procedure, you must have the SYSDBA or both ANALYZE ANY DICTIONARY and ANALYZE ANY system privilege.

Exceptions

ORA-20000: Insufficient privileges

ORA-20001: Invalid or illegal input values

Usage Notes

- This setting is honored only if there is no preference specified for the table to be analyzed.
- Both arguments are of type VARCHAR2 and values are enclosed in quotes, even when they represent numbers.

Example 197-16 Overriding Statistics Preferences at the Global Level

You use the `SET_GLOBAL_PREFS` procedure to set the `ESTIMATE_PERCENT` preference to `5` for every table in the database that does not have a table preference set. Because `sh.costs` does not have a preference set, the global setting applies to this table.

```
SQL> EXEC DBMS_STATS.SET_GLOBAL_PREFS ('ESTIMATE_PERCENT', '5');

PL/SQL procedure successfully completed.
```

You use `SET_TABLE_PREFS` to set the `PREFERENCE_OVERRIDES_PARAMETER` preference to true for the `sh.sales` table only.

```
SQL> EXEC
DBMS_STATS.SET_TABLE_PREFS('sh','sales','PREFERENCE_OVERRIDES_PARAMETER','TRUE
');

PL/SQL procedure successfully completed.
```

A script attempts to set `ESTIMATE_PERCENT` to `10` when gathering statistics for `sh.sales`. However, because `PREFERENCE_OVERRIDES_PARAMETER` is `TRUE` for this table, and because a global preference is defined, Oracle Database gathers statistics using the global setting of `5` rather than the specified setting of `10`:

```
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS ('sh', 'costs', ESTIMATE_PERCENT=>10);

PL/SQL procedure successfully completed.
```

Example 197-17 Configuring High-Frequency Automatic Optimizer Statistics Collection

Oracle Database 19c introduces high-frequency automatic optimizer statistics collection. This lightweight task periodically gathers statistics for stale objects. The default interval is 15 minutes. In contrast to the automated statistics collection job, the high-frequency task does not perform actions such as purging statistics for non-existent objects or invoking Optimizer Statistics Advisor.

In this example, you enable high-frequency collection, set its maximum run time to a half hour, and then set the frequency interval to 10 minutes:

```
EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_STATUS','ON');
EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_MAX_RUN_TIME','1800');
EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_INTERVAL','600');
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set optimizer statistics preferences

SET_INDEX_STATS Procedures

These procedures set index-related statistics.

The version of this procedure that accepts `ext_stats` sets statistics for use with domain indexes. The statistics type specified is the type to store in the dictionary, in addition to the actual user-defined statistics. If this statistics type is null, then the database stores the statistics type associated with the index or column.

Syntax

```
DBMS_STATS.SET_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2  DEFAULT NULL,
   stattab       VARCHAR2  DEFAULT NULL,
   statid        VARCHAR2  DEFAULT NULL,
   numrows       NUMBER    DEFAULT NULL,
   numlblks      NUMBER    DEFAULT NULL,
   numdist       NUMBER    DEFAULT NULL,
   avglblk       NUMBER    DEFAULT NULL,
   avgdblk       NUMBER    DEFAULT NULL,
   clstfct       NUMBER    DEFAULT NULL,
   indlevel      NUMBER    DEFAULT NULL,
   flags         NUMBER    DEFAULT NULL,
   statown       VARCHAR2  DEFAULT NULL,
   no_invalidate BOOLEAN   DEFAULT to_no_invalidate_type(
                                    get_param('NO_INVALIDATE')),
   guessq        NUMBER    DEFAULT NULL,
   cachedblk     NUMBER    DEFAULT NULL,
   cachehit      NUMBER    DEFUALT NULL,
   force         BOOLEAN   DEFAULT FALSE);
```

Use the following syntax for user-defined domain index statistics:

```
DBMS_STATS.SET_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2  DEFAULT NULL,
   stattab       VARCHAR2  DEFAULT NULL,
   statid        VARCHAR2  DEFAULT NULL,
   ext_stats     RAW,
   stattypown    VARCHAR2 DEFAULT NULL,
   stattypname   VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT to_no_invalidate_type(
                                    get_param('NO_INVALIDATE')),
   force         BOOLEAN   DEFAULT FALSE);
```

Parameters

Table 197-129 SET_INDEX_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
indname	Name of the index
partname	Name of the index partition in which to store the statistics. If the index is partitioned and if partname is NULL, then the statistics are stored at the global index level.
stattab	User statistics table identifier describing where to store the statistics. If stattab is NULL, then the statistics are stored directly in the dictionary.
statid	Identifier (optional) to associate with these statistics within stattab (Only pertinent if stattab is not NULL)
ext_stats	User-defined statistics
stattypown	Schema of the statistics type
stattypname	Name of the statistics type
numrows	Number of rows in the index (partition)
numlblks	Number of leaf blocks in the index (partition)
numdist	Number of distinct keys in the index (partition)
avglblk	Average integral number of leaf blocks in which each distinct key appears for this index (partition). If not provided, then this value is derived from numlblks and numdist.
avgdblk	Average integral number of data blocks in the table pointed to by a distinct key for this index (partition). If not provided, then this value is derived from clstfct and numdist.
clstfct	See clustering_factor column of the all_indexes view for a description
indlevel	Height of the index (partition)
flags	For internal Oracle use (should be left as NULL)
statown	Schema containing stattab (if different than ownname)
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
guessq	Guess quality. See the pct_direct_access column of the all_indexes view for a description.
cachedblk	Internal use only. Do not set.
cachehit	Internal use only. Do not set.
force	Sets the values even if statistics of the index are locked

Security Model

To invoke this procedure you must be owner of the table, or you need the `ANALYZE ANY` privilege. For objects owned by `SYS`, you need to be either the owner of the table, or you need the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

`ORA-20001`: Invalid input value

`ORA-20005`: Object statistics are locked

Usage Notes

- The Optimizer uses the cached data to estimate number of cached blocks for index or statistics table access. The total cost of the operation will be combined from the I/O cost of reading not cached blocks from disk, the CPU cost of getting cached blocks from the buffer cache, and the CPU cost of processing the data.
- Oracle maintains `cachedblk` and `cachehit` at all times but uses correspondent caching statistics for optimization as part of the table and index statistics only when the user calls `DBMS_STATS.GATHER_[TABLE/INDEX/SCHEMA/DATABASE]_STATS` procedure for auto mode or `DBMS_STATS.GATHER_SYSTEM_STATS` for manual mode. In order to prevent the user from utilizing inaccurate and unreliable data, the optimizer will compute a 'confidence factor' for each `cachehit` and a `cachedblk` for each object. If the 'confidence factor' for the value meets confidence criteria, this value will be used, otherwise the defaults will be used.
- The automatic maintenance algorithm for object caching statistics assumes that there is only one major workload for the system and adjusts statistics to this workload, ignoring other "minor" workloads. If this is not the case, you must use manual mode for maintaining object caching statistics.
- The object caching statistics maintenance algorithm for auto mode prevents you from using statistics in the following situations
  - When not enough data has been analyzed, such as when an object has been recently create
  - When the system does not have one major workload resulting in averages not corresponding to real values.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set artificial statistics

SET_PARAM Procedure

This deprecated procedure sets default values for parameters of `DBMS_STATS` procedures.

> **Note:**
>
> This subprogram has been replaced by improved technology and is maintained only for purposes of backward compatibility. In this case, use the SET_GLOBAL_PREFS Procedure.
>
> See also DBMS_STATS Deprecated Subprograms.

You can use the `GET_PARAM Function` to get the current default value of a parameter.

Syntax

```
DBMS_STATS.SET_PARAM (
   pname       IN    VARCHAR2,
   pval        IN    VARCHAR2);
```

Parameters

Table 197-130 SET_PARAM Procedure Parameters

Parameter	Description
`pname`	The parameter name The default value for following parameters can be set. • `CASCADE` - The default value for `CASCADE` set by `SET_PARAM` is not used by export/import procedures.It is used only by gather procedures. • `DEGREE` • `ESTIMATE_PERCENT` • `METHOD_OPT` • `NO_INVALIDATE` • `GRANULARITY` • `AUTOSTATS_TARGET` - This parameter is applicable only for auto statistics collection. The value of this parameter controls the objects considered for statistics collection (see `pval`)
`pval`	The parameter value. If `NULL` is specified, it will set the default value determined by Oracle. When `pname` is `AUTOSTATS_TARGET`, the following are valid values: • `'ALL'` - Statistics are collected for all objects in the system • `'ORACLE'` - Statistics are collected for all Oracle owned objects • `'AUTO'` - Oracle decides for which objects to collect statistics

Usage Notes

- To run this procedure, you must have the `SYSDBA` or both the `ANALYZE ANY DICTIONARY` and `ANALYZE ANY` system privileges.

- Note that both arguments are of type `VARCHAR2` and the values need to be enclosed in quotes even when they represent numbers.
- Note also the difference between `NULL` and `'NULL'`:
  - When `NULL` is unquoted, this sets the parameter to the value Oracle recommends.
  - In the case of the quoted `'NULL'`, this sets the value of the parameter to `NULL`.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

`ORA-20001`: Invalid or illegal input value

Examples

```
DBMS_STATS.SET_PARAM('CASCADE','DBMS_STATS.AUTO_CASCADE');
DBMS_STATS.SET_PARAM('ESTIMATE_PERCENT','5');
DBMS_STATS.SET_PARAM('DEGREE','NULL');
```

SET_PLSQL_PREFS Procedure

Sets the preference either for a specific PL/SQL function or for all functions in the same package.

Syntax

```
DBMS_STATS.SET_PLSQL_PREFS(
    ownname         IN VARCHAR2,
    package_name    IN VARCHAR2,
    function_name   IN VARCHAR2,
    pname           IN VARCHAR2,
    pvalue          IN VARCHAR2
);
```

Parameters

Table 197-131 SET_PLSQL_PREFS Procedure Parameters

Parameter	Description
`user`	Owner of the function. If NULL, the current user is used.
`package_name`	The PL/SQL package name. If NULL, this specifies a stand-alone, top-level function.
`function_name`	Name of the PL/SQL function. To apply the preference to all functions within the package, specify NULL.
`pname`	The preference that you want to set. Currently only `'dynamic_stats'` is accepted.
`pvalue`	The value of the preference.

> **Note:**
>
> Preference settings have the following order of precedence from higher to lowest:
>
> 1. The plsql_function_dynamic_stats hint
> 2. The session level parameter plsql_function_dynamic_stats
> 3. The specific PL/SQL function preference that is set using `SET_PLSQL_PREFS`
> 4. The global preference that is set using `SET_GLOBAL_PLSQL_PREFS`
>
> PL/SQL preferences are persistent across sessions. The session level parameter `plsql_function_dynamic_stats` is not persistent across sessions.

Table 197-132 PL/SQL Preferences

Preference Name	Description and Possible Values
`dynamic_stats`	Dynamic sampling is a performance feature that instructs the optimizer to read a sample of data during query compilation to formulate dynamic statistics. Dynamic statistics supplement optimizer statistics such as table and index block counts, table and join cardinalities (estimated number of rows), join column statistics, and GROUP BY statistics. This information helps the optimizer improve plans by making better estimates for predicate cardinality. The preference value controls whether functions can be executed during dynamic sampling to improve cardinality estimates. High-performance PL/SQL is generally suitable for use in dynamic sampling, but you should exclude long-running PL/SQL functions since they will increase SQL compilation times. These are the options for the `'dynamic_stats'` preference: • `ON` - Allow dynamic statistics for this PL/SQL function • `OFF` - Disallow dynamic statistics for this PL/SQL function • `CHOOSE` - Leave enabling or disabling dynamic statistics to the discretion of the optimizer. • `NULL` - the dynamic statistics preference set for this function is deleted and the global preference is used.

Examples

In this example the owner is the current schema and so the `USER` parameter is set to `NULL`. Dynamic statistics for the named function are turned off.

```
EXEC SET_PLSQL_PREFS(NULL, '<PACKAGE_NAME>', '<FUNCTION_NAME>',
'dynamic_stats', 'OFF')
```

In this example, the `function_name` parameter is set to NULL. Therefore the dynamic statistics preference is set on all functions within the named PL/SQL package.

```
EXEC SET_PLSQL_PREFS('<USER>', '<PACKAGE_NAME>', NULL, 'dynamic_stats', 'ON')
```

In the example below, `pvalue` is set to NULL. The preference set for this function is deleted and the global preference is used. Setting `pvalue` to NULL in this procedure has the same effect as executing `DELETE_PLSQL_PREFS` on the function.

```
EXEC SET_PLSQL_PREFS('<USER>', '<PACKAGE_NAME>',
'<FUNCTION_NAME>','dynamic_stats', NULL)
```

Usage Notes

> **See Also:**
>
> The following privileges scheme is used for setting PL/SQL preferences:
>
> - The owner of the PL/SQL function (or package) can set preferences on those functions.
> - If the session user has SYSDBA privileges, they can set preferences on any PL/SQL function.
> - If the session user has the ANALYZE ANY DICTIONARY privilege, they can set preferences on any function or package.
> - If the user in the session has the ANALYZE ANY privilege, they can set preferences on any PL/SQL function or package except those owned by SYS.
>
> For additional controls over PL/SQL preferences see Controls for Dynamic Sampling With PL/SQL Functions in the *Database SQL Tuning Guide*. Oracle provides a hint to control PL/SQL dynamic sampling behavior for an individual SQL statement, as well as a session-level parameter.
>
> The *SQL Tuning Guide* also provides details on dynamic statistics at Configuring Options for Dynamic Statistics

SET_PROCESSING_RATE Procedure

This procedure sets the value of rate of processing for a given operation.

Syntax

```
DBMS_STATS.SET_PROCESSING_RATE (
   opname      IN    VARCHAR2,
   procrate    IN    NUMBER);
```

Parameters

Table 197-133 SET_PROCESSING_RATE Procedure Parameters

Parameter	Description
opname	Name of the operation. Valid values are as follows: • AGGR • ALL • CPU • CPU_ACCESS • CPU_AGGR • CPU_BYTES_PER_SEC • CPU_FILTER • CPU_GBY • CPU_HASH_JOIN • CPU_JOIN • CPU_NL_JOIN • CPU_RANDOM_ACCESS • CPU_SEQUENTIAL_ACCESS • CPU_SM_JOIN • CPU_SORT • HASH • IO • IO_ACCESS • IO_BYTES_PER_SEC • IO_RANDOM_ACCESS • IO_SEQUENTIAL_ACCESS • MEMCMP • MEMCPY
procrate	Processing rate.

Security Model

You must have the OPTIMIZER_PROCESSING_RATE role to run this procedure.

Usage Notes

AUTO DOP uses processing rates to determine the optimal degree of parallelism for a SQL statement.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or illegal input value

SET_SCHEMA_PREFS Procedure

This procedure sets the statistics preferences of all tables owned by the specified user.

Syntax

```
DBMS_STATS.SET_SCHEMA_PREFS (
    ownname    IN   VARCHAR2,
    pname      IN   VARCHAR2,
    pvalue     IN   VARCHAR2);
```

Parameters

Table 197-134 SET_SCHEMA_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
pname	Preference name. You can set the default value for the following preferences: • AUTO_STAT_EXTENSIONS • CASCADE • DEGREE • ESTIMATE_PERCENT • GLOBAL_TEMP_TABLE_STATS • GRANULARITY • INCREMENTAL • INCREMENTAL_LEVEL • INCREMENTAL_STALENESS • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • TABLE_CACHED_BLOCKS
pvalue	Preference value. If NULL is specified, it sets the database default value.

Table 197-135 Statistics Preferences

Preference	Description
AUTO_STAT_EXTENSIONS	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. You can consider a synopsis as an internal management structure that samples distinct values. You can set the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
CASCADE	Determines whether index statistics are collected as part of gathering table statistics.
DEGREE	Determines degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	Determines the percentage of rows to estimate. The valid range is [0.000001,100]. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to have Oracle determine the appropriate sample size for good statistics. This is the default.
GLOBAL_TEMP_TABLE_STATS	This preference takes the following values: • SHARED - All sessions see the same set of statistics • SESSION - Statistics gathered by the GATHER_TABLE_STATS Procedure on a global temporary table are session specific, and hence are only going to be used by the queries issued in the same session as the statistics gathering process. Session-specific statistics are deleted when a session is ended.

Table 197-135 (Cont.) Statistics Preferences

Preference	Description
GRANULARITY	Determines the granularity of statistics to collect (only pertinent if the table is partitioned). Possible values are: 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION'- gathers partition-level statistics 'SUBPARTITION' - gathers subpartition-level statistics.
INCREMENTAL	Determines whether the global statistics of a partitioned table will be maintained without doing a full table scan. With partitioned tables it is very common to load new data into a new partition. As new partitions are added and data loaded, the global table statistics need to be kept up to date. Oracle will update the global table statistics by scanning only the partitions that have been changed instead of the entire table if the following conditions hold: • INCREMENTAL value for the partitioned table is set to TRUE • PUBLISH value for the partitioned table is set to TRUE; • User specifies AUTO_SAMPLE_SIZE for ESTIMATE_PERCENT and AUTO for GRANULARITY when gathering statistics on the table If the INCREMENTAL value for the partitioned table was set to FALSE (default value), a full table scan is used to maintain the global statistics which is a much more resource intensive and time-consuming operation for large tables.

Table 197-135 (Cont.) Statistics Preferences

Preference	Description
INCREMENTAL_LEVEL	Controls which synopses to collect when INCREMENTAL preference is set to TRUE It takes two values: • TABLE - table level synopses are gathered. This is used when you want to exchange this table with a partition. You can run GATHER_TABLE_STATS on this table with INCREMENTAL to TRUE and INCREMENTAL_LEVEL to TABLE before the exchange. The result is that table level synopses are gathered on this table (currently Oracle supports only table level synopses on non-predestined tables). Once the exchange occurs, the partition will have synopses which come from the table level synopses of the table before exchange. This preference value can be only used in the SET_TABLE_PREFS Procedure. It is not allowed in the SET_GLOBAL/DATABASE/SCHEMA_PREFS procedures. • PARTITION - partition level synopses are gathered. This is the default value. If PARTITION is set on a non partitioned table, no synopses are gathered.

Table 197-135 (Cont.) Statistics Preferences

Preference	Description

INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT, USE_LOCKED_STATS, ALLOW_MIXED_FORMAT'.
	The parameter accepts the following values:
	• USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference.
	For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCEMENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics.
	Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCEMENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics.
	• USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes.
	Assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics.
	• ALLOW_MIXED_FORMAT—Partitions with synopses in adaptive sampling format are not considered stale, even when the APPROXIMATE_NDV_ALGORITHM preference is set to HYPERLOGLOG. The database uses existing synopses to derive global NDV. This is the default setting.
	• NULL—A partition or subpartition is considered stale when it has any DML changes.
	For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.
	Note that the following two executions are different:
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	`EXEC DBMS_STATS.SET_TABLE_PREFS`

Table 197-135 (Cont.) Statistics Preferences

Preference	Description
	`('sh', 'sales', 'INCREMENTAL_STALENESS', null);` The first execution uses single quotes to set the preference to the value `NULL`, whereas the second sets the preference to the default, which is `ALLOW_MIXED_FORMAT`.
`METHOD_OPT`	Controls column statistics collection and histogram creation. It accepts either of the following options, or both in combination: • `FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause]` `size_clause` is defined as `size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY}` – `integer`: Number of histogram buckets. Must be in the range [1,2048]. – `REPEAT` : Collects histograms only on the columns that already have histograms – `AUTO` : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns – `SKEWONLY` : Oracle determines the columns on which to collect histograms based on the data distribution of the columns - `column_name` : name of a column - `extension` : can be either a column group in the format of `(column_name, colume_name [, ...])` or an expression The default is `FOR ALL COLUMNS SIZE AUTO`.
`NO_INVALIDATE`	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • `TRUE`: Dependent cursors are not invalidated. • `FALSE`: Dependent cursors are marked for immediate invalidation. • `AUTO`: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.

Table 197-135 (Cont.) Statistics Preferences

Preference	Description
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS procedure. The preference takes the following values: • GATHER — Gathers statistics for all objects in the table. This is the default. • GATHER AUTO — Oracle recommends using GATHER AUTO to gather necessary statistics, such as histograms, after a table has been bulk loaded and acquired online statistics. This is applicable only to tables that are not using INCREMENTAL statistics. For partitioned tables using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO will gather statistics for a table if it is marked stale or has no statistics. In addition, statistics will be gathered for partitions and sub-partitions that are marked stale or have no statistics. For tables not using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO may gather statistics for partitions and sub-partitions, even if they are not marked stale.
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether newly gathered statistics will be published after the statistics gathering job completes. In releases before Oracle Database 11*g* Release 1 (11.1), when a statistic gathering job completed, the new statistics were automatically published in the dictionary tables. In subsequent releases, you can gather statistics without publishing them immediately. Thus, you can test new statistics before publishing them.
STALE_PERCENT	Determines the percentage of rows in a table that have to change before the statistics on that table are deemed stale and should be regathered. The valid domain for stale_percent is non-negative numbers. The default value is 10%. Note that if you set stale_percent to zero the AUTO STATS gathering job will gather statistics for this table every time a row in the table is modified.
TABLE_CACHED_BLOCKS	Specifies the average number of blocks assumed to be cached in the buffer cache when calculating the index clustering factor. The preference applies only when gathering statistics using DBMS_STATS. Index statistics gathered during CREATE INDEX or REBUILD INDEX operations will use the default value 1.

Security Model

To run this procedure, you must be the schema owner, or have the `SYSDBA` privilege, or have the `ANALYZE ANY` system privilege.

Exceptions

`ORA-20000`: Object does not exist or insufficient privileges

`ORA-20001`: Invalid or illegal input value

Usage Notes

Both arguments are of type `VARCHAR2` and values are enclosed in quotes, even when they represent numbers.

Examples

```
DBMS_STATS.SET_SCHEMA_PREFS('SH','CASCADE','DBMS_STATS.AUTO_CASCADE');
DBMS_STATS.SET_SCHEMA_PREFS('SH','ESTIMATE_PERCENT','9');
DBMS_STATS.SET_SCHEMA_PREFS('SH','DEGREE','99');
```

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set optimizer statistics preferences

SET_SYSTEM_STATS Procedure

This procedure sets systems statistics.

Syntax

```
DBMS_STATS.SET_SYSTEM_STATS (
   pname          VARCHAR2,
   pvalue         NUMBER,
   stattab   IN   VARCHAR2 DEFAULT NULL,
   statid    IN   VARCHAR2 DEFAULT NULL,
   statown   IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Table 197-136 SET_SYSTEM_STATS Procedure Parameters

Parameter	Description
pname	The parameter name to get, which can have one of the following values: • iotfrspeed—I/O transfer speed in bytes for each millisecond • ioseektim - Seek time + latency time + operating system overhead time, in milliseconds • sreadtim - Average time to read single block (random read), in milliseconds • mreadtim - Average time to read an mbrc block at once (sequential read), in milliseconds • cpuspeed - Average number of CPU cycles for each second, in millions, captured for the workload (statistics collected using 'INTERVAL' or 'START' and 'STOP' options) • cpuspeednw - Average number of CPU cycles for each second, in millions, captured for the no-workload (statistics collected using 'NOWORKLOAD' option. • mbrc - Average multiblock read count for sequential read, in blocks • maxthr - Maximum I/O system throughput, in bytes/second • slavethr - Average slave I/O throughput, in bytes/second
pvalue	Parameter value to get
stattab	Identifier of the user statistics table where the statistics will be obtained. If stattab is null, the statistics will be obtained from the dictionary.
statid	Optional identifier associated with the statistics saved in the stattab
statown	Schema containing stattab (if different from current schema)

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid input value

ORA-20002: Bad user statistics table; may need to be upgraded

ORA-20003: Unable to set system statistics

ORA-20004: Parameter does not exist

Usage Notes

To run this procedure, you need the GATHER_SYSTEM_STATISTICS role.

SET_TABLE_PREFS Procedure

This procedure sets the statistics preferences of the specified table in the specified schema.

Syntax

```
DBMS_STATS.SET_TABLE_PREFS (
    ownname    IN  VARCHAR2,
    tabname    IN  VARCHAR2,
```

```
    pname       IN  VARCHAR2,
    pvalue      IN  VARCHAR2);
```

Parameters

Table 197-137 SET_TABLE_PREFS Procedure Parameters

Parameter	Description
ownname	Owner name
tabname	Table name
pname	Preference name. You can set the default value for following preferences: • APPROXIMATE_NDV_ALGORITHM • AUTO_STAT_EXTENSIONS • CASCADE • DEGREE • ESTIMATE_PERCENT • GRANULARITY • INCREMENTAL • INCREMENTAL_LEVEL • INCREMENTAL_STALENESS • METHOD_OPT • NO_INVALIDATE • OPTIONS • PREFERENCE_OVERRIDES_PARAMETER • PUBLISH • STALE_PERCENT • TABLE_CACHED_BLOCKS
pvalue	Preference value. If NULL is specified, it will set the Oracle default value.

Table 197-138 Statistics Preferences

Preference	Description
APPROXIMATE_NDV_ALGORITHM	Specifies the synopsis generation algorithm. A synopsis is special type of statistic that tracks the number of distinct values (NDV) for each column in a partition. Consider a synopsis as an internal management structure that samples distinct values. You can specify the following preferences: • REPEAT OR HYPERLOGLOG This is the default. If INCREMENTAL is enabled on the table, then the database preserves the format of any existing synopses that use the adaptive sampling algorithm. However, the database creates any new synopses in HyperLogLog format. This approach is attractive when existing performance is acceptable, and you do not want to incur the performance cost of reformatting legacy content. • ADAPTIVE SAMPLING The database uses the adaptive sampling algorithm for all synopses. This is the most conservative option. • HYPERLOGLOG The database uses the HyperLogLog algorithm for all new and stale synopses. In contrast to adaptive sampling, the HyperLogLog algorithm uses a randomization technique. The advantages of HyperLogLog over adaptive sampling are: – The accuracy of the new algorithm is similar to the original algorithm. – The memory required is significantly lower, which typically leads to huge reductions in synopsis size.
AUTO_STAT_EXTENSIONS	Controls the automatic creation of extensions when database statistics are gathered. You can set the following values: • ON — When applicable, a SQL plan directive can trigger the creation of column group statistics based on usage of columns in the predicates in the workload. • OFF— The database does not create column group statistics automatically. The database creates them only when the CREATE_EXTENDED_STATS function is executed, or when extended statistics are specified explicitly in the METHOD_OPT clause of DBMS_STATS. This is the default.
CASCADE	Determines whether to collect index statistics as part of gathering table statistics.

Table 197-138 (Cont.) Statistics Preferences

Preference	Description
CONCURRENT	This preference determines whether statistics will be gathered concurrently on multiple objects, or serially, one object at a time: • 'MANUAL' - Concurrency will be turned on only for manual statistics gathering. • 'AUTOMATIC': Concurrency will be turned on only for the automatic statistics gathering. • 'ALL': Concurrency will be turned on for both manual and automatic statistics gathering. • 'OFF': Concurrency will be turned off for both manual and automatic statistics
DEGREE	Determines degree of parallelism used for gathering statistics.
ESTIMATE_PERCENT	Determines the percentage of rows to estimate. The valid range is [0.000001,100]. Use the constant DBMS_STATS.AUTO_SAMPLE_SIZE to have Oracle determine the appropriate sample size for good statistics. This is the default.
GLOBAL_TEMP_TABLE_STATS	This controls whether the statistics gathered for a global temporary table should be stored as shared statistics or session statistics. It takes two values: • SHARED - All sessions see the same set of statistics • SESSION - Statistics gathered by the GATHER_TABLE_STATS Procedure on a global temporary table are session specific, and hence are only going to be used by the queries issued in the same session as the statistics gathering process. Session-specific statistics are deleted when a session is ended.
GRANULARITY	Determines granularity of statistics to collect (only pertinent if the table is partitioned). 'ALL' - Gathers all (subpartition, partition, and global) statistics 'AUTO'- Determines the granularity based on the partitioning type. This is the default value. 'DEFAULT' - Gathers global and partition-level statistics. This option is obsolete, and while currently supported, it is included in the documentation for legacy reasons only. You should use the 'GLOBAL AND PARTITION' for this functionality. Note that the default value is now 'AUTO'. 'GLOBAL' - Gathers global statistics 'GLOBAL AND PARTITION' - Gathers the global and partition level statistics. No subpartition level statistics are gathered even if it is a composite partitioned object. 'PARTITION'- Gathers partition-level statistics 'SUBPARTITION' - Gathers subpartition-level statistics.

Table 197-138 (Cont.) Statistics Preferences

Preference	Description
INCREMENTAL	Determines whether or not the global statistics of a partitioned table will be maintained without doing a full table scan. With partitioned tables it is very common to load new data into a new partition. As new partitions are added and data loaded, the global table statistics need to be kept up to date. Oracle will update the global table statistics by scanning only the partitions that have been changed instead of the entire table if the following conditions hold: • INCREMENTAL value for the partitioned table is set to TRUE; • PUBLISH value for the partitioned table is set to TRUE; • User specifies AUTO_SAMPLE_SIZE for ESTIMATE_PERCENT and AUTO for GRANULARITY when gathering statistics on the table. If the INCREMENTAL value for the partitioned table was set to FALSE (default value), a full table scan is used to maintain the global statistics which is a much more resource intensive and time-consuming operation for large tables.
INCREMENTAL_LEVEL	This value controls what synopses to collect when INCREMENTAL preference is set to TRUE It takes two values: • TABLE - table level synopses are gathered. This is used when you want to exchange this table with a partition. You can run GATHER_TABLE_STATS on this table with INCREMENTAL to TRUE and INCREMENTAL_LEVEL to TABLE before the exchange. The result is that table level synopses are gathered on this table (currently Oracle supports only table level synopses on non-predestined tables). Once the exchange occurs, the partition will have synopses which come from the table level synopses of the table before exchange. This preference value can be only used in the SET_TABLE_PREFS Procedure. It is not allowed in the SET_GLOBAL/DATABASE/SCHEMA_PREFS procedures. • PARTITION - partition level synopses are gathered. This is the default value. If PARTITION is set on a non partitioned table, no synopses are gathered.

Table 197-138 (Cont.) Statistics Preferences

Preference	Description
INCREMENTAL_STALENESS	Specifies when a partition or subpartition is considered stale. This parameter takes an enumeration of values, such as 'USE_STALE_PERCENT' and 'USE_LOCKED_STATS'. You can also specify multiple values, such as 'USE_STALE_PERCENT,USE_LOCKED_STATS,ALLOW_MIXED_FORMAT'. The parameter accepts the following values: • USE_STALE_PERCENT—A partition or subpartition is not considered stale when DML changes are below the threshold set by the STALE_PERCENT preference. For example, assume that STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. The partition has 5% DML changes. The database does not regather statistics. Assume a different case in which STALE_PERCENT is 10. You specify USE_STALE_PERCENT for INCREMENTAL_STALENESS. However, in this case the partition is locked and has 20% of DML changes. Because the partition is locked, the database does not regather statistics. • USE_LOCKED_STATS—Locked partitions or subpartitions statistics are never considered stale, regardless of DML changes. For example, assume that STALE_PERCENT is 10. You specify 'USE_LOCKED_STATS, USE_STALE_PERCENT'. The partition, which is locked, has 20% DML changes. The partition is not considered stale. The database uses existing statistics to derive global statistics. • ALLOW_MIXED_FORMAT—Adaptive sampling synopses and HyperLogLog synopses are permitted to coexist. • NULL—A partition or subpartition is considered stale when it has any DML changes. For example, assume that STALE_PERCENT is 10. You specify the value 'NULL' for INCREMENTAL_STALENESS. The partition has 5% of DML changes. The database regathers statistics.

Table 197-138 (Cont.) Statistics Preferences

Preference	Description

	Note that the following two executions are different:
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', 'NULL');`
	
	`EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'sales', 'INCREMENTAL_STALENESS', null);`
	
	The first execution uses single quotes to set the preference to the value `NULL`, whereas the second sets the preference to the default, which is `ALLOW_MIXED_FORMAT`.

`METHOD_OPT`	The value controls column statistics collection and histogram creation. It accepts either of the following options, or both in combination: • `FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause]` `size_clause` is defined as `size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY}` – `integer`: Number of histogram buckets. Must be in the range [1,2048]. – `REPEAT` : Collects histograms only on the columns that already have histograms – `AUTO` : Oracle determines the columns on which to collect histograms based on data distribution and the workload of the columns – `SKEWONLY` : Oracle determines the columns on which to collect histograms based on the data distribution of the columns - `column_name` : name of a column - `extension` : can be either a column group in the format of `(column_name, colume_name [, ...])` or an expression The default is `FOR ALL COLUMNS SIZE AUTO`.

Table 197-138 (Cont.) Statistics Preferences

Preference	Description
NO_INVALIDATE	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
OPTIONS	Determines the options parameter used in the GATHER_TABLE_STATS procedure. The preference takes the following values: • GATHER — Gathers statistics for all objects in the table. This is the default. • GATHER AUTO — Oracle recommends using GATHER AUTO to gather necessary statistics, such as histograms, after a table has been bulk loaded and acquired online statistics. This is applicable only to tables that are not using INCREMENTAL statistics. For partitioned tables using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO will gather statistics for a table if it is marked stale or has no statistics. In addition, statistics will be gathered for partitions and sub-partitions that are marked stale or have no statistics. For tables not using INCREMENTAL statistics, GATHER_TABLE_STATS with GATHER AUTO may gather statistics for partitions and sub-partitions, even if they are not marked stale.
PREFERENCE_OVERRIDES_PARAMETER	Determines whether to override the input value of a parameter with the preference value of that parameter for a statistics operation. Possible values are: • TRUE — Ignores input parameter values, and uses the value of the corresponding preference. • FALSE — Obeys input parameter values. Specifying this preference does not change the order of precedence of table, global, and default.
PUBLISH	Determines whether the database publishes newly gathered statistics after the gathering job completes. You can gather statistics without publishing them immediately. This technique enables you to test new statistics before publishing them.

Table 197-138 (Cont.) Statistics Preferences

Preference	Description
STALE_PERCENT	Determines the percentage of rows in a table that must change before the statistics on that table are stale and need to be regathered. The valid domain for stale_percent is non-negative numbers. The default value is 10, which means that a table having more than 10% of changes is considered stale.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid or illegal input values

Usage Notes

- To run this procedure, you must connect as owner of the table or have the ANALYZE ANY system privilege.
- All arguments are of type VARCHAR2 and values are enclosed in quotes, even when they represent numbers.

Examples

```
DBMS_STATS.SET_TABLE_PREFS('SH', 'SALES', 'CASCADE', 'DBMS_STATS.AUTO_CASCADE');
DBMS_STATS.SET_TABLE_PREFS('SH', 'SALES', 'ESTIMATE_PERCENT', '9');
DBMS_STATS.SET_TABLE_PREFS('SH', 'SALES', 'DEGREE', '99');
```

Example 197-18 Overriding Statistics Preferences

In this example, legacy scripts set ESTIMATE_PERCENT explicitly rather than using the recommended AUTO_SAMPLE_SIZE. Your goal is to prevent users from using these scripts to set preferences on the sh.costs table.

No preference for ESTIMATE_PERCENT is set for sh.costs or at the global level, so the preference defaults to AUTO_SAMPLE_SIZE:

```
SELECT DBMS_STATS.GET_PREFS ('ESTIMATE_PERCENT', 'sh','costs') AS
"STAT_PREFS" FROM DUAL;

STAT_PREFS
----------
DBMS_STATS.AUTO_SAMPLE_SIZE
```

By default, Oracle Database accepts preferences that are passed to the statistics gathering procedures. To override these parameters, use SET_TABLE_PREFS to set the PREFERENCE_OVERRIDES_PARAMETER preference to TRUE for the costs table only:

```
EXEC DBMS_STATS.SET_TABLE_PREFS ('sh', 'costs',
'PREFERENCE_OVERRIDES_PARAMETER', 'TRUE');
```

A user-created script attempts to set `estimate_percent` to `100` when gathering statistics for `sh.costs`.

```
EXEC DBMS_STATS.GATHER_TABLE_STATS('sh', 'costs', ESTIMATE_PERCENT=>100);
```

However, because `PREFERENCE_OVERRIDES_PARAMETER` is `TRUE` for this table, Oracle Database gathers statistics using `AUTO_SAMPLE_SIZE`, which is the default, rather than the specified value of `100`.

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set optimizer statistics preferences

SET_TABLE_STATS Procedure

This procedure creates artificial table statistics for testing purposes.

Syntax

```
DBMS_STATS.SET_TABLE_STATS (
   ownname         VARCHAR2,
   tabname         VARCHAR2,
   partname        VARCHAR2 DEFAULT NULL,
   stattab         VARCHAR2 DEFAULT NULL,
   statid          VARCHAR2 DEFAULT NULL,
   numrows         NUMBER   DEFAULT NULL,
   numblks         NUMBER   DEFAULT NULL,
   avgrlen         NUMBER   DEFAULT NULL,
   flags           NUMBER   DEFAULT NULL,
   statown         VARCHAR2 DEFAULT NULL,
   no_invalidate   BOOLEAN  DEFAULT to_no_invalidate_type (
                                  get_param('NO_INVALIDATE')),
   cachedblk       NUMBER   DEFAULT NULL,
   cachehit        NUMBER   DEFAULT NULL,
   force           BOOLEAN  DEFAULT FALSE,
   im_imcu_count   NUMBER   DEFAULT NULL,
   im_block_count  NUMBER   DEFAULT NULL,
   scanrate        NUMBER   DEFAULT NULL);
```

Parameters

Table 197-139 SET_TABLE_STATS Procedure Parameters

Parameter	Description
`ownname`	Name of the schema.
`tabname`	Name of the table.

Table 197-139 (Cont.) SET_TABLE_STATS Procedure Parameters

Parameter	Description
partname	Name of the table partition in which to store the statistics. If the table is partitioned and partname is NULL, then the statistics are stored at the global table level.
stattab	Table in which to store the statistics. If stattab is NULL, then the database stores the statistics in the data dictionary.
statid	Identifier (optional) to associate with these statistics within stattab. This identifier is only relevant if stattab is not NULL.
numrows	Number of rows in the table or partition.
numblks	Number of blocks that the table or partition occupies.
avgrlen	Average row length for the table or partition.
flags	For internal use only. Do not set.
statown	Schema containing stattab (if different than ownname).
no_invalidate	Controls the invalidation of dependent cursors when statistics are gathered. The parameter takes the following values: • TRUE: Dependent cursors are not invalidated. • FALSE: Dependent cursors are marked for immediate invalidation. • AUTO: This is the default value. Rolling invalidation is used to invalidate all dependent cursors over a period of time. The performance impact on the database is reduced especially in cases where a large number of cursors are invalidated. You can change the default using the SET_DATABASE_PREFS Procedure, SET_GLOBAL_PREFS Procedure, SET_SCHEMA_PREFS Procedure and SET_TABLE_PREFS Procedure.
cachedblk	For internal use only. Do not set.
cachehit	For internal use only. Do not set.
force	A flag that determines the behavior when statistics are locked. If TRUE, then the procedure sets the values even if the table statistics are locked. By default, the setting is FALSE.
im_imcu_count	The number of In-Memory Compression Units (IMCUs) in the table or partition.
im_block_count	The number of In-Memory blocks in the table or partition.
scanrate	The rate, in MB/s, at which the database scans external tables. This parameter is relevant only for external tables.

Security Model

To invoke this procedure you must be owner of the table, or have the ANALYZE ANY privilege. For objects owned by SYS, you must be either the owner of the table, or have the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Invalid input value

ORA-20002: Bad user statistics table; may need to upgrade it

`ORA-20005`: Object statistics are locked

Usage Notes

For testing purposes, you can manually create artificial statistics for a table, index, or the system using the `DBMS_STATS.SET_*_STATS` procedures. These procedures insert the artificial statistics into the data dictionary directly (when `stattab` is null) or into a user-created table.

> **Note:**
>
> The `DBMS_STATS.SET_*_STATS` procedures are intended for development testing only. Do not use them in a production database. If you set statistics in the data dictionary, then Oracle Database considers the set statistics as the “real” statistics, which means that statistics gathering jobs may not re-gather artificial statistics when they do not meet the criteria for staleness.

The most typical use cases for the `DBMS_STATS.SET_*_STATS` procedures are showing how execution plans change as the numbers of rows or blocks in a table change, or creating realistic statistics for temporary tables.

- The optimizer uses the cached data to estimate number of cached blocks for index or statistics table access. The database calculates the total cost of the operation by combining the I/O cost of reading not cached blocks from disk, the CPU cost of getting cached blocks from the buffer cache, and the CPU cost of processing the data.
- The database maintains `cachedblk` and `cachehit` at all times. However, the database uses the corresponding caching statistics for optimization as part of the table and index statistics only when the user calls the `DBMS_STATS.GATHER_[TABLE/INDEX/SCHEMA/DATABASE]_STATS` procedure for automatic mode or `DBMS_STATS.GATHER_SYSTEM_STATS` for manual mode. To prevent the user from utilizing inaccurate and unreliable data, the optimizer computes a “confidence factor” for each `cachehit` and a `cachedblk` for each object. If the confidence factor for the value meets confidence criteria, then the database uses this value; otherwise, the database uses defaults.
- The automatic maintenance algorithm for object caching statistics assumes that only one major database workload exists. The algorithm adjusts statistics to this workload, ignoring other "minor" workloads. If this assumption is false, then you must use manual mode for maintaining object caching statistics.
- The object caching statistics maintenance algorithm for automatic mode prevents you from using statistics in the following situations:
  - When not enough data has been analyzed, such as when an object has been recently created
  - When the database does not have one major workload, resulting in averages that do not correspond to real values

> **See Also:**
>
> *Oracle Database SQL Tuning Guide* to learn how to set artificial statistics

SHOW_EXTENDED_STATS_NAME Function

This function returns the name of the statistics entry that is created for the user-specified extension. It raises an error if no extension has been created.

Syntax

```
DBMS_STATS.SHOW_EXTENDED_STATS_NAME (
   ownname    VARCHAR2,
   tabname    VARCHAR2,
   extension  VARCHAR2)
 RETURN VARCHAR2;
```

Parameters

Table 197-140 SHOW_EXTENDED_STATS_NAME Function Parameters

Parameter	Description
ownname	Owner name of a table
tabname	Name of the table
extension	Can be either a column group or an expression. Suppose the specified table has two column c1, c2. An example column group can be "(c1, c2)" and an example expression can be "(c1 + c2)".

Exceptions

ORA-20000: Object does not exist or insufficient privileges

ORA-20001: Error when processing extension

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

TRANSFER_STATS Procedure

This procedure transfers statistics for specified table(s) from a remote database specified by dblink to the local database.

The statistics at the source database are retained. It likewise transfers statistics-related structures such as synopses and DML monitoring information.

Syntax

```
DBMS_STATS.TRANSFER_STATS (
   ownname     IN     VARCHAR2,
   tabname     IN     VARCHAR2,
   dblink      IN     VARCHAR2,   options     IN     NUMBER DEFAULT NULL);
```

Parameters

Table 197-141 TRANSFER_STATS Procedure Parameters

Parameter	Description
ownname	Owner name of a table. If NULL all schemas in the database. If NULL, the procedure will transfer global preferences as well.
tabname	Name of the table. If NULL, all tables in OWNNAME.
dblink	Database link name
options	By default the procedure does not transfer the global preferences. Specifying ADD_GLOBAL_PREFS copies global preferences.

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

UNLOCK_PARTITION_STATS Procedure

This procedure enables the user to unlock statistics for a partition.

Syntax

```
DBMS_STATS.UNLOCK_PARTITION_STATS (
    ownname    VARCHAR2,
    tabname    VARCHAR2,
    partname   VARCHAR2);
```

Parameters

Table 197-142 UNLOCK_PARTITION_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema to unlock
tabname	Name of the table
partname	[Sub]Partition name

Usage Notes

To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.

UNLOCK_SCHEMA_STATS Procedure

This procedure unlocks the statistics on all the tables in schema.

Syntax

```
DBMS_STATS.UNLOCK_SCHEMA_STATS (
   ownname     VARCHAR2);
```

Parameters

Table 197-143 UNLOCK_SCHEMA_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema

Usage Notes

- To invoke this procedure you must be owner of the table, or you need the `ANALYZE ANY` privilege. For objects owned by `SYS`, you need to be either the owner of the table, or you need the `ANALYZE ANY DICTIONARY` privilege or the `SYSDBA` privilege.
- When statistics on a table is locked, all the statistics depending on the table, including table statistics, column statistics, histograms and statistics on all dependent indexes, are considered to be locked.
- The `SET_*, DELETE_*, IMPORT_*, GATHER_*` procedures that modify statistics in the dictionary of an individual table, index or column will raise an error if statistics of the object is locked.
- Procedures that operates on multiple objects (such as `GATHER_SCHEMA_STATS`) will skip modifying the statistics of an object if it is locked. Many procedures have force argument to override the lock.
- Neither the UNLOCK_SCHEMA_STATS Procedure nor the UNLOCK_TABLE_STATS Procedure is designed to unlock statistics of corresponding partitions. When you invoke theLOCK_TABLE_STATS Procedure, it sets the statistics lock bit at the table level. In that case, you cannot gather statistics on dependent objects such as partitions and indexes. By the same token, if table statistics are locked, the dependents are locked and you do not need to explicitly invoke the LOCK_PARTITION_STATS Procedure.

UNLOCK_TABLE_STATS Procedure

This procedure unlocks the statistics on the table.

Syntax

```
DBMS_STATS.UNLOCK_TABLE_STATS (
   ownname     VARCHAR2,
   tabname     VARCHAR2);
```

Parameters

Table 197-144 UNLOCK_TABLE_STATS Procedure Parameters

Parameter	Description
ownname	Name of the schema
tabname	Name of the table

Usage Notes

- To invoke this procedure you must be owner of the table, or you need the ANALYZE ANY privilege. For objects owned by SYS, you need to be either the owner of the table, or you need the ANALYZE ANY DICTIONARY privilege or the SYSDBA privilege.
- When statistics on a table is locked, all the statistics depending on the table, including table statistics, column statistics, histograms and statistics on all dependent indexes, are considered to be locked.
- The SET_*, DELETE_*, IMPORT_*, GATHER_* procedures that modify statistics in the dictionary of an individual table, index or column will raise an error if statistics of the object is locked.
- Procedures that operates on multiple objects (such as GATHER_SCHEMA_STATS) will skip modifying the statistics of an object if it is locked. Many procedures have force argument to override the lock.
- Neither the UNLOCK_SCHEMA_STATS Procedure nor the UNLOCK_TABLE_STATS Procedure is designed to unlock statistics of corresponding partitions. When you invoke theLOCK_TABLE_STATS Procedure, it sets the statistics lock bit at the table level. In that case, you cannot gather statistics on dependent objects such as partitions and indexes. By the same token, if table statistics are locked, the dependents are locked and you do not need to explicitly invoke the LOCK_PARTITION_STATS Procedure.

UPGRADE_STAT_TABLE Procedure

This procedure upgrades a user statistics table from an older version.

Syntax

```
DBMS_STATS.UPGRADE_STAT_TABLE (
   ownname    VARCHAR2,
   stattab    VARCHAR2);
```

Parameters

Table 197-145 UPGRADE_STAT_TABLE Procedure Parameters

Parameter	Description
ownname	Name of the schema
stattab	Name of the table

Exceptions

ORA-20000: Unable to upgrade table

Usage Notes

To invoke this procedure you need the privileges to drop and create a table.