192
DBMS_SQLPA

The DBMS_SQLPA package provides the interface to implement the SQL Performance Analyzer.

The chapter contains the following topics:

- Overview
- Security Model
- Summary of DBMS_SQLPA Subprograms

DBMS_SQLPA Overview

The `DBMS_SQLPA` package provides a capacity to help users predict the impact of system environment changes on the performance of a SQL workload. The interface lets users build and then compare two different versions of the workload performance, analyze the differences between the two versions, and unmask the SQL statements that might be impacted by the changes.

The package provides a task-oriented interface to implement the SQL Performance Analyzer. For example

1. You use the CREATE_ANALYSIS_TASK Functions to create an analysis task for a single statement or a group of SQL statements.s
2. The EXECUTE_ANALYSIS_TASK Function & Procedure executes a previously created analysis task.
3. The REPORT_ANALYSIS_TASK Function displays the results of an analysis task.

DBMS_SQLPA Security Model

This package is available to `PUBLIC` and performs its own security checking. All analysis task interfaces (`XXX_ANALYSIS_TASK`) require privilege `ADVISOR`.

Summary of DBMS_SQLPA Subprograms

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

Table 192-1 DBMS_SQLPA Package Subprograms

Subprogram	Description
CANCEL_ANALYSIS_TASK Procedure	Cancels the currently executing task analysis of one or more SQL statements
CREATE_ANALYSIS_TASK Functions	Creates an advisor task to process and analyze one or more SQL statements
DROP_ANALYSIS_TASK Procedure	Drops a SQL analysis task

Table 192-1 (Cont.) DBMS_SQLPA Package Subprograms

Subprogram	Description
EXECUTE_ANALYSIS_TASK Function & Procedure	Executes a previously created analysis task
INTERRUPT_ANALYSIS_TASK Procedure	Interrupts the currently executing analysis task
REPORT_ANALYSIS_TASK Function	Displays the results of an analysis task
RESET_ANALYSIS_TASK Procedure	Resets the currently executing analysis task to its initial state
RESUME_ANALYSIS_TASK Procedure	Resumes a previously interrupted analysis task that was created to process a SQL tuning set.
SET_ANALYSIS_TASK_PARAMETER Procedures	Sets the SQL analysis task parameter value
SET_ANALYSIS_DEFAULT_PARAMETER Procedures	Sets the SQL analysis task parameter default value

CANCEL_ANALYSIS_TASK Procedure

This procedure cancels the currently executing analysis task. All intermediate result data is removed from the task.

Syntax

```
DBMS_SQLPA.CANCEL_ANALYSIS_TASK(
 task_name         IN VARCHAR2);
```

Parameters

Table 192-2 CANCEL_ANALYSIS_TASK Procedure Parameters

Parameter	Description
task_name	Name of the task to cancel

Examples

Canceling a task when there is a need to stop it executing and it is not required to view any already-completed results:

```
EXEC DBMS_SQLPA.CANCEL_ANALYSIS_TASK(:my_task);
```

CREATE_ANALYSIS_TASK Functions

These functions create an advisor task to process and analyze one or more SQL statements.

> **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.

You can use different forms of this function to:

- Create an analysis task for a single statement given its text.
- Create an analysis task for a single statement from the cursor cache given its identifier.
- Create an analysis task for a single statement from the workload repository given a range of snapshot identifiers.
- Create an analysis task for a SQL tuning set.

In all cases, the function creates an advisor task and sets its parameters.

Syntax

SQL text format. This form of the function is called to prepare the analysis of a single statement given its text.

```
DBMS_SQLPA.CREATE_ANALYSIS_TASK(
  sql_text         IN CLOB,
  bind_list        IN sql_binds := NULL,
  parsing_schema   IN VARCHAR2  := NULL,
  task_name        IN VARCHAR2  := NULL,
  description      IN VARCHAR2  := NULL)
RETURN VARCHAR2;
```

SQL ID format. This form of the function is called to prepare the analysis of a single statement from the cursor cache given its identifier.

```
DBMS_SQLPA.CREATE_ANALYSIS_TASK(
  sql_id           IN VARCHAR2,
  plan_hash_value  IN NUMBER    := NULL,
  task_name        IN VARCHAR2  := NULL,
  con_name         IN VARCHAR2    DEFAULT,
  description      IN VARCHAR2  := NULL)
RETURN VARCHAR2;
```

Workload Repository format. This form of the function is called to prepare the analysis of a single statement from the workload repository given a range of snapshot identifiers.

```
DBMS_SQLPA.CREATE_ANALYSIS_TASK(
  dbid             IN NUMBER       DEFAULT,
  begin_snap       IN NUMBER,
  end_snap         IN NUMBER,
  sql_id           IN VARCHAR2,
  plan_hash_value  IN NUMBER    := NULL,
```

```
  task_name        IN VARCHAR2  := NULL,
  description      IN VARCHAR2  := NULL)
  con_name         IN VARCHAR2     DEFAULT,
RETURN VARCHAR2;
```

SQLSET format. This form of the function is called to prepare the analysis of a SQL tuning set.

```
DBMS_SQLPA.CREATE_ANALYSIS_TASK(
  sqlset_name       IN VARCHAR2,
  basic_filter      IN VARCHAR2 :=  NULL,
  con_name          IN VARCHAR2     DEFAULT,
  order_by          IN VARCHAR2 :=  NULL,
  top_sql           IN VARCHAR2 :=  NULL,
  task_name         IN VARCHAR2 :=  NULL,
  description       IN VARCHAR2 :=  NULL
  sqlset_owner      IN VARCHAR2 :=  NULL)
RETURN VARCHAR2;
```

Parameters

Table 192-3 *CREATE_ANALYSIS_TASK Function Parameters*

Parameter	Description
sql_text	Text of a SQL statement
bind_list	A set of bind values
parsing_schema	Name of the schema where the statement can be compiled
task_name	Optional analysis task name
dbid	The DBID for imported or PDB-level AWR data. If NULL, then the current database DBID is used
con_name	Container for the SPA task. The semantics depend on the function format: For the SQL ID format, this parameter specifies the container from which the database fetches the SQL statement for using with SPA. SPA will analyze the statement in this container. If null, then the database uses the current PDB for SPA analysis. For the AWR format, this parameter specifies the container from whose AWR data the database fetches the SQL statement for using with SPA. SPA will analyze the statement in this container. If null, then the database uses the current PDB for SPA analysis. The following statements are true of all function formats: • In a non-CDB, this parameter is ignored. • In a PDB, this parameter must be null or match the container name of the PDB. Otherwise, error occurs. • In a CDB root, this parameter must be null or match the container name of a container in this CDB. Otherwise, error occurs.
description	Description of the SQL analysis task to a maximum of 256 characters
sql_id	Identifier of a SQL statement
plan_hash_value	Hash value of the SQL execution plan
begin_snap	Begin snapshot identifier
end_snap	End snapshot identifier
sqlset_name	SQL tuning set name
basic_filter	SQL predicate to filter the SQL from the SQL tuning set

Table 192-3 (Cont.) *CREATE_ANALYSIS_TASK Function Parameters*

Parameter	Description
order_by	Order-by clause on the selected SQL
top_sql	Top N SQL after filtering and ranking
sqlset_owner	The owner of the SQL tuning set, or NULL for the current schema owner

Return Values

A SQL analysis task name that is unique by user (two different users can give the same name to their advisor tasks).

Examples

```
variable stmt_task VARCHAR2(64);
variable sts_task  VARCHAR2(64);

-- Sql text format
EXEC :stmt_task := DBMS_SQLPA.CREATE_ANALYSIS_TASK(
     sql_text => 'select quantity_sold from sales s, times t where s.time_id = t.time_id
and s.time_id = TO_DATE(''24-NOV-00'')');

-- Sql id format (cursor cache)
EXEC :stmt_task := DBMS_SQLPA.CREATE_ANALYSIS_TASK(
    sql_id       => 'ay1m3ssvtrh24');

-- Workload repository format
exec :stmt_task := DBMS_SQLPA.CREATE_ANALYSIS_TASK(
    begin_snap  => 1,
    end_snap    => 2,
    sql_id      => 'ay1m3ssvtrh24');

-- Sql tuning set format (first we need to load an STS, then analyze it)
EXEC :sts_task := DBMS_SQLPA.CREATE_ANALYSIS_TASK( -
  sqlset_name    =>   'my_workload', -
  order_by       =>   'BUFFER_GETS', -
  description    =>   'process workload ordered by buffer gets');
```

DROP_ANALYSIS_TASK Procedure

This procedure drops a SQL analysis task.The task and all its result data are deleted.

Syntax

```
DBMS_SQLPA.DROP_ANALYSIS_TASK(
 task_name         IN VARCHAR2);
```

Parameters

Table 192-4 DROP_ANALYSIS_TASK Procedure Parameters

Parameter	Description
task_name	The name of the analysis task to drop

EXECUTE_ANALYSIS_TASK Function & Procedure

This function and procedure executes a previously created analysis task, the function version returning the new execution name.

Syntax

```
DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
   task_name         IN VARCHAR2,
   execution_type    IN VARCHAR2               := 'test execute',
   execution_name    IN VARCHAR2               := NULL,
   execution_params  IN dbms_advisor.argList   := NULL,
   execution_desc    IN VARCHAR2               := NULL)
 RETURN VARCHAR2;

DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
   task_name         IN VARCHAR2,
   execution_type    IN VARCHAR2               := 'test execute',
   execution_name    IN VARCHAR2               := NULL,
   execution_params  IN dbms_advisor.argList   := NULL,
   execution_desc    IN VARCHAR2               := NULL);
```

Parameters

Table 192-5 EXECUTE_ANALYSIS_TASK Function & Procedure Parameters

Parameter	Description
task_name	Identifier of the task to execute
execution_type	Type of the action to perform by the function. If NULL it will default to the value of the DEFAULT_EXECUTION_TYPE parameter. Possible values are: • [TEST] EXECUTE - test-execute every SQL statement and collect its execution plans and execution statistics. The resulting plans and statistics will be stored in the advisor framework. This is default. • EXPLAIN PLAN - generate explain plan for every statement in the SQL workload. This is similar to the EXPLAIN PLAN command. The resulting plans will be stored in the advisor framework in association with the task. • COMPARE [PERFORMANCE] - analyze and compare two versions of SQL performance data. The performance data is generated by test-executing or generating explain plan of the SQL statements. Use this option when two executions of type EXPLAIN_PLAN or TEST_EXECUTE already exist in the task • CONVERT SQLSET - used to read the statistics captured in a SQL Tuning Set and model them as a task execution. This can be used when you wish to avoid executing the SQL statements because valid data for the experiment already exists in the SQL Tuning Set.
execution_name	A name to qualify and identify an execution. If not specified, it will be generated by the advisor and returned by function.
execution_params	List of parameters (name, value) for the specified execution. The execution parameters have effect only on the execution for which they are specified. They will override the values for the parameters stored in the task (set through the SET_ANALYSIS_DEFAULT_PARAMETER Procedures).
execution_desc	A 256-length string describing the execution

Usage Notes

SQL performance analyzer task can be executed multiples times without having to reset it. For example, when a task is created to perform a change impact analysis on a SQL workload, the created task has to be executed before making any change in the system environment to build a version of the workload that will be used as a reference for performance analysis. Once the change has been made, a second execution is required to build the post-change version of the workload. Finally, the task has to be executed a third time to let the advisor analyze and compare the performance of the workload in both versions.

Examples

1. Create a task with a purpose of change impact analysis

```
EXEC :tname := DBMS_SQLPA.CREATE_ANALYSIS_TASK(
    sqlset_name     => 'my_sts');
```

2. Make baseline or the before change execution

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
    task_name       => :tname,
    execution_type  => 'test execute',
    execution_name  => 'before_change');
```

3. Make change

```
...
```

4. Make the after change version of the workload performance

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
   task_name          => :tname, -
   execution_type     => 'test execute',
   execution_name     => 'after_change')
```

5. Compare the two versions of the workload

By default we always compare the results of the two last executions. The SQL Performance Analyzer uses the `elapsed_time` as a default metric for comparison. Here we are changing it to `buffer_gets` instead.

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(
   :tname,'comparison_metric', 'buffer_gets');
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
   task_name         => :tname, -
   execution_type    => 'compare performance', -
   execution_name    => 'after_change');
```

Use the following call if you would like to explicitly specify the two executions to compare as well as the comparison metric to use.

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
   task_name        => :tname, -
   execution_type   => 'compare performance',
   execution_params => dbms_advisor.arglist(
      'execution_name1',
      'before_change',
      'execution_name2',
      'after_change',
      'comparison_metric',
      'buffer_gets'));
```

INTERRUPT_ANALYSIS_TASK Procedure

This procedure interrupts the currently executing analysis task. All intermediate result data will not be removed from the task.

Syntax

```
DBMS_SQLPA.INTERRUPT_ANALYSIS_TASK(
   task_name         IN VARCHAR2);
```

Parameters

Table 192-6 INTERRUPT_ANALYSIS_TASK Procedure Parameters

Parameter	Description
task_name	Identifier of the analysis task to interrupt

Examples

```
EXEC DBMS_SQLPA.INTERRUPT_ANALYSIS_TASK(:my_task);
```

REPORT_ANALYSIS_TASK Function

This procedure displays the results of an analysis task.

Syntax

```
DBMS_SQLPA.REPORT_ANALYSIS_TASK(
   task_name       IN   VARCHAR2,
   type            IN   VARCHAR2   := 'TEXT',
   level           IN   VARCHAR2   := 'TYPICAL',
   section         IN   VARCHAR2   := 'SUMMARY',
   object_id       IN   NUMBER     := NULL,
   top_sql         IN   NUMBER     := 100,
   execution_name  IN   VARCHAR2   := NULL,
   task_owner      IN   VARCHAR2   := NULL,
   order_by        IN   VARCHAR2   := NULL)
RETURN CLOB;
```

Parameters

Table 192-7 REPORT_ANALYSIS_TASK Function Parameters

Parameter	Description
task_name	Name of the task to report
type	Type of the report to produce. Possible values are TEXT (default), HTML, XML and ACTIVE (see Usage Notes).

Table 192-7 (Cont.) REPORT_ANALYSIS_TASK Function Parameters

Parameter	Description
level	Level of detail in the report: • ALL - details of all SQL • BASIC - currently the same as typical • CHANGED - only SQL with changed performance • CHANGED_PLANS - only SQL with plan changes • ERRORS - SQL with errors only • IMPROVED - only improved SQL • REGRESSED - only regressed SQL • TIMEOUT - only SQL which timed-out during execution • TYPICAL (default) - show information about every statement analyzed, including changing and errors • UNCHANGED - only SQL with unchanged performance • UNCHANGED_PLANS - only SQL with unchanged plans • UNSUPPORTED - only SQL not supported by SPAs
section	Optionally limit the report to a single section (ALL for all sections): • SUMMARY (default) - workload summary only • ALL - summary and details on SQL
object_id	Identifier of the advisor framework object that represents a given SQL in a tuning set (STS)
top_sql	Number of SQL statements in a STS for which the report is generated
execution_name	Name of the task execution to use. If NULL, the report will be generated for the last task execution.
task_owner	Owner of the relevant analysis task. Defaults to the current schema owner.
order_by	How to sort SQL statements in the report (summary and body). Possible values: • CHANGE_DIFF - sort SQL statements by change difference in SQL performance in terms of the comparison Metric • NULL (default) - order SQL statement by impact on workload • SQL_IMPACT - order SQL statement by change impact on SQL • WORKLOAD_IMPACT - same as NULL • METRIC_DELTA - same as CHANGE_DIFF

Return Values

A CLOB containing the desired report.

Usage Notes

ACTIVE reports have a rich, interactive user interface similar to Enterprise Manager while not requiring any EM installation. The report file built is in HTML format so it can be interpreted by most modern browsers. The code powering the active report is downloaded transparently by the web browser when the report is first viewed, hence viewing it requires outside connectivity.

Examples

```
-- Get the whole report for the single statement case.
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(:stmt_task) from dual;
```

```
-- Show me the summary for the sts case.
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(:sts_task, 'TEXT', 'TYPICAL', 'SUMMARY')
FROM DUAL;

-- Show me the findings for the statement I'm interested in.
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(:sts_task, 'TEXT', 'TYPICAL', 'ALL', 5) from dual;
```

RESET_ANALYSIS_TASK Procedure

This procedure is called on an analysis task that is not currently executing to prepare it for re-execution.

All intermediate result data will be deleted.

Syntax

```
DBMS_SQLPA.RESET_ANALYSIS_TASK(
 task_name         IN VARCHAR2);
```

Parameters

Table 192-8 RESET_ANALYSIS_TASK Procedure Parameters

Parameter	Description
task_name	Identifier of the analysis task to reset

Examples

```
-- reset and re-execute a task
EXEC DBMS_SQLPA.RESET_ANALYSIS_TASK(:sts_task);

-- re-execute the task
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(:sts_task);
```

RESUME_ANALYSIS_TASK Procedure

This procedure resumes a previously interrupted or `FAILED` (with a fatal error) task execution.

Syntax

```
DBMS_SQLPA.RESUME_ANALYSIS_TASK(
 task_name         IN VARCHAR2,
 basic_filter      IN VARCHAR2 := NULL);
```

Parameters

Table 192-9 RESUME_ANALYSIS_TASK Procedure Parameters

Parameter	Description
task_name	Identifier of the analysis task to resume
basic_filter	A SQL predicate to filter the SQL from the SQL tuning set. Note that this filter will be applied in conjunction with the basic filter (parameter `basic_filter`) that was specified when calling the CREATE_ANALYSIS_TASK Functions.

Usage Notes

Resuming a single SQL analysis task (a task that was created to analyze a single SQL statement as compared to a SQL Tuning Set) is not supported.

Examples

```
-- Interrupt the task
EXEC DBMS_SQLPA.INTERRUPT_ANALYSIS_TASK(:conc_task);

-- Once a task is interrupted, we can elect to reset it, resume it, or check
-- out its results and then decide.  For this example we will just resume.

EXEC DBMS_SQLPA.RESUME_ANALYSIS_TASK(:conc_task);
```

SET_ANALYSIS_TASK_PARAMETER Procedures

This procedure sets the SQL analysis task parameter value.

Syntax

This form of the procedure updates the value of a SQL analysis parameter of type `VARCHAR2`.

```
DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(
  task_name         IN VARCHAR2,
  parameter         IN VARCHAR2,
  value             IN VARCHAR2,
  test_execute_dop  IN NUMBER DEFAULT 0,
  compare_resultset IN BOOLEAN DEFAULT TRUE);
```

This form of the procedure updates the value of a SQL analysis parameter of type `NUMBER`.

```
DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(
  task_name         IN VARCHAR2,
  parameter         IN VARCHAR2,
  value             IN NUMBER
  test_execute_dop  IN NUMBER DEFAULT 0,
  compare_resultset IN BOOLEAN DEFAULT TRUE);
```

Parameters

Table 192-10 SET_ANALYSIS_TASK_PARAMETER Procedure Parameters

Parameter	Description
task_name	Identifier of the task to execute

Table 192-10 (Cont.) SET_ANALYSIS_TASK_PARAMETER Procedure Parameters

Parameter	Description
parameter	Name of the parameter to set. The possible analysis parameters that can be set by this procedure are: • APPLY_CAPTURED_COMPILENV: indicates whether the advisor could use the compilation environment captured with the SQL statements. The default is 0 (that is, NO). • BASIC_FILTER: basic filter for SQL tuning set • CELL_SIMULATION_ENABLED: for more details, see the helper script tcellsim.sql in the ADMIN directory. • COMPARISON_METRIC: specify an expression of execution statistics to use in performance comparison (Example: buffer_gets, cpu_time + buffer_gets * 10) • DATABASE_LINK: can be set to the global name of a PUBLIC database link. When it is set, SQL Performance Analyzer will use the database link for all TEST EXECUTE and EXPLAIN PLAN operations by sending the SQL statements to the remote database to be processed remotely. The analysis results will still be stored on the local database. • DAYS_TO_EXPIRE: number of days until the task is deleted • DEFAULT_EXECUTION_TYPE: the task will default to this type of execution when none is specified by the EXECUTE_ANALYSIS_TASK Function & Procedure.

Table 192-10 (Cont.) SET_ANALYSIS_TASK_PARAMETER Procedure Parameters

Parameter	Description
parameter (contd.)	• DISABLE_MULTI_EXEC: SQL statements are executed multiple times and runtime statistics are then averaged. Set this parameter to TRUE to disable this capability. In this case, each SQL in the SQL tuning set is executed only once. • EXECUTE_TRIGGERS: Set this parameter to TRUE to execute all statement-level triggers in the FULLDML mode. If the parameter is set to FALSE, then the triggers will not be executed even in FULLDML mode of test execution. Any changes incurred due to potential execution of triggers are always rolled back by SPA. The default value of this parameter is FALSE. • EXECUTION_DAYS_TO_EXPIRE: number of days until the tasks's executions will be deleted (without deleting the task) • EXECUTE_FULLDML: TRUE to execute DML statement fully, including acquiring row locks and modifying rows; FALSE (default) to execute only the query part of the DML without modifying data. When TRUE, SQL Performance Analyzer will issue a rollback following DML execution to prevent persistent changes from being made by the DML. • EXECUTION_NAME1: name of the first task execution to analyze • EXECUTION_NAME2: name of the second task execution to analyze • LOCAL_TIME_LIMIT: per-statement time out (seconds) • METRIC_DELTA_THRESHOLD: threshold of the difference between the SQL performance metric before and after the change. The default value is zero. • NUM_ROWS_TO_FETCH: specifies the number of rows to be fetched for an SQL query. You can use one of the following values: – ALL_ROWS: Fetches all the rows for an SQL query – AVERAGE: Number of result rows is calculated as the ratio of total rows processed and total executions for each SQL in the STS – AUTO: Number of result rows is determined using the value of optimizer_mode parameter of the optimizer environment captured in the STS. If the value of optimizer_mode is ALL_ROWS, then all result rows will be fetched. If its value is FIRST_ROWS_n, then n result rows will be fetched by the SPA. – A valid number: Fetches the exact number of rows specified by in the SQL query The default value is ALL_ROWS. • PLAN_FILTER: plan filter for SQL tuning set (see SELECT_SQLSET for possible values) • PLAN_LINES_COMPARISON: - ALWAYS --line by line comparison of plans in all scenarios.- AUTO - Line by Line comparison of plans only if phv2 is not available and phv1 is different- NONE (default) - line by line comparison of plans only if phv is unknown • RANK_MEASURE1: first ranking measure for SQL tuning set • RANK_MEASURE2: second possible ranking measure for SQL tuning set • RANK_MEASURE3: third possible ranking measure for SQL tuning set

Table 192-10 (Cont.) SET_ANALYSIS_TASK_PARAMETER Procedure Parameters

Parameter	Description
	• REPLACE_SYSDATE_WITH: Returns a fixed date for all calls to SYSDATE within the SPA task execution. You can use one of the following values: – CURRENT_SYSDATE: SYSDATE calls return the current date. – SQLSET_SYSDATE: SYSDATE calls return the value of the column LAST_EXEC_START_TIME in the STS The default value is CURRENT_SYSDATE. • RESUME_FILTER: a extra filter for SQL tuning sets besides BASIC_FILTER • SQL_IMPACT_THRESHOLD: threshold of a change impact on a SQL statement. Same as the previous parameter, but at the level of the SQL statement. • SQL_LIMIT: maximum number of SQL statements to process • SQL_PERCENTAGE: percentage filter of SQL tuning set statements • SQLSET_NAME: name of the SQL tuning set to associate to the specified task or task execution. This parameter is mainly using in comparing two SQL tuning sets using SPA. • SQLSET_OWNER: owner of the SQL tuning set specified using task parameter SQLSET_NAME. • TIME_LIMIT: global time out (seconds)
parameter (contd.)	• WORKLOAD_IMPACT_THRESHOLD: threshold of a SQL statement impact on a workload. Statements which workload change impact is below the absolute value of this threshold will be ignored and not considered for improvement or regression. • CON_DBID_MAPPING: provide a mapping of multitenant container database (CDB) IDs. When it is set, SQL Performance Analyzer uses the new CDB ID when it finds a match for the old CDB ID and executes the SQL in that container.
value	New value of the specified parameter
test_execute_dop	Specifies the requested level of concurrency with which a SPA task should be executed. Values 0 or 1 indicate that the SPA task will run with no additional processes as it used to run in releases prior to Oracle Database 18c Release. A value of n (higher than 1) means that n background SPA processes are being requested to concurrently process the input workload.
compare_resultset	Directs SPA to detect if the result-sets between the two trials being compared are different. If differences are seen in the result-sets of any SQL statement between the two trials being compared, the SPA comparison report will indicate this for every such SQL statement. • If set to TRUE the result set comparison will be performed • If set to FALSE result set comparison will not be performed.

Usage Notes

The actual number of processes granted might be equal to or lower than the number requested using the test_execute_dop parameter. This parameter applies only to test-execute or explain plan type of trials that process a SQL Tuning set.

Examples

To request two concurrent processes to execute the SPA task:

```
    dbms_sqlpa.set_analysis_task_parameter(:tname,'TEST_EXECUTE_DOP',2)
```

To enable result-set validation

```
   exec
dbms_sqlpa.set_analysis_task_parameter(:atname,'COMPARE_RESULTSET','TRUE')
```

To disable result-set validation:

```
    exec
dbms_sqlpa.set_analysis_task_parameter(:atname,'COMPARE_RESULTSET','FALSE')
```

SET_ANALYSIS_DEFAULT_PARAMETER Procedures

This procedure sets the SQL analysis task parameter default value.

Syntax

This form of the procedure updates the default value of an analyzer parameter of type `VARCHAR2`.

```
DBMS_SQLPA.SET_ANALYSIS_DEFAULT_PARAMETER(
   parameter    IN  VARCHAR2,
   value        IN  VARCHAR2);
```

This form of the procedure updates the `default` value of an analyzer parameter of type `NUMBER`.

```
DBMS_SQLPA.SET_ANALYSIS_DEFAULT_PARAMETER(
   parameter    IN  VARCHAR2,
   value        IN  NUMBER);
```

Parameters

Table 192-11 SET_ANALYSIS_DEFAULT_PARAMETER Procedure Parameters

Parameter	Description
parameter	Name of the parameter to set. The possible analysis parameters that can be set by this procedure are: • APPLY_CAPTURED_COMPILENV: indicates whether the advisor could use the compilation environment captured with the SQL statements. The default is 0 (that is, NO). • BASIC_FILTER: basic filter for SQL tuning set • COMPARISON_METRIC: specify an expression of execution statistics to use in performance comparison (Example: buffer_gets, cpu_time + buffer_gets * 10) • DATABASE_LINK: can be set to the global name of a PUBLIC database link. When it is set, SQL Performance Analyzer will use the database link for all TEST EXECUTE and EXPLAIN PLAN operations by sending the SQL statements to the remote database to be processed remotely. The analysis results will still be stored on the local database. • DAYS_TO_EXPIRE: number of days until the task is deleted • DEFAULT_EXECUTION_TYPE: the task will default to this type of execution when none is specified by the EXECUTE_ANALYSIS_TASK Function & Procedure. • EXECUTE_FULLDML: TRUE to execute DML statement fully, including acquiring row locks and modifying rows; FALSE (default) to execute only the query part of the DML without modifying data. When TRUE, SQL Performance Analyzer will issue a rollback following DML execution to prevent persistent changes from being made by the DML. • EXECUTION_DAYS_TO_EXPIRE: number of days until the tasks's executions will be deleted (without deleting the task) • EXECUTION_NAME1: name of the first task execution to analyze • EXECUTION_NAME2: name of the second task execution to analyze • LOCAL_TIME_LIMIT: per-statement time out (seconds)
parameter (contd.)	• PLAN_FILTER: plan filter for SQL tuning set (see SELECT_SQLSET for possible values) • RANK_MEASURE1: first ranking measure for SQL tuning set • RANK_MEASURE2: second possible ranking measure for SQL tuning set • RANK_MEASURE3: third possible ranking measure for SQL tuning set • RESUME_FILTER: a extra filter for SQL tuning sets besides BASIC_FILTER • SQL_IMPACT_THRESHOLD: threshold of a change impact on a SQL statement. Same as the previous parameter, but at the level of the SQL statement. • SQL_LIMIT: maximum number of SQL statements to process • SQL_PERCENTAGE: percentage filter of SQL tuning set statements • TIME_LIMIT: global time out (seconds) • WORKLOAD_IMPACT_THRESHOLD: threshold of a SQL statement impact on a workload. Statements which workload change impact is below the absolute value of this threshold will be ignored and not considered for improvement or regression.

Table 192-11 (Cont.) SET_ANALYSIS_DEFAULT_PARAMETER Procedure Parameters

Parameter	Description
`value`	New value of the specified parameter