242
DBMS_XPLAN

The `DBMS_XPLAN` package provides an easy way to display the output of the `EXPLAIN PLAN` command in several, predefined formats.

You can also use the `DBMS_XPLAN` package to display the plan of a statement stored in the Automatic Workload Repository (AWR) or stored in a SQL tuning set. It further provides a way to display the SQL execution plan and SQL execution runtime statistics for cached SQL cursors based on the information stored in the `V$SQL_PLAN` and `V$SQL_PLAN_STATISTICS_ALL` fixed views. Finally, it displays plans from a SQL plan baseline.

> **See Also:**
>
> - For more information on the `EXPLAIN PLAN` command, the AWR, and SQL tuning set, see *Oracle Database SQL Tuning Guide*.
> - For more information on the `V$SQL_PLAN` fixed view, see *Oracle Database Reference*
> - For more information on the `V$SQL_PLAN_STATISTICS` fixed view, see *Oracle Database Reference*

This chapter contains the following topics:

DBMS_XPLAN Overview

The `DBMS_XPLAN` package supplies five table functions.

These functions are listed below:

- `DISPLAY` - to format and display the contents of a plan table.
- `DISPLAY_AWR` - to format and display the contents of the execution plan of a stored SQL statement in the AWR.
- `DISPLAY_CURSOR` - to format and display the contents of the execution plan of any loaded cursor.
- `DISPLAY_SQL_PLAN_BASELINE` - to display one or more execution plans for the SQL statement identified by SQL handle
- `DISPLAY_SQLSET` - to format and display the contents of the execution plan of statements stored in a SQL tuning set.

DBMS_XPLAN Security Model

This package runs with the privileges of the calling user, not the package owner (SYS). The table function DISPLAY_CURSOR requires SELECT or READ privileges on the following fixed views: VSQL_PLAN, VSESSION and V$SQL_PLAN_STATISTICS_ALL. This function also requires SELECT/READ permissions on V$SQL.

DISPLAY_AWR Function requires the user to have SELECT or READ privileges on DBA_HIST_SQL_PLAN, AWR_ROOT_SQL_PLAN, AWR_PDB_SQL_PLAN, AWR_ROOT_SQLTEXT, DBA_HIST_SQLTEXT, AWR_PDB_SQLTEXT, and V$DATABASE.

DISPLAY_SQLSET Function requires the user to have the SELECT or READ privilege on ALL_SQLSET_STATEMENTS and ALL_SQLSET_PLANS.

DISPLAY_SQL_PLAN_BASELINE Function requires the user to have the SELECT or READ privilege on DBA_SQL_PLAN_BASELINES as well as the privileges to execute the SQL statement for which the user is trying to get the plan.

The preceding privileges are granted automatically as part of SELECT_CATALOG_ROLE.

DBMS_XPLAN Data Structures

The DBMS_XPLAN package defines a TABLE type.

Table Types

- DBMS_XPLAN PLAN_OBJECT_LIST Table Type

DBMS_XPLAN PLAN_OBJECT_LIST Table Type

This type allows for a list of generic objects as input to the COMPARE_PLANS function.

Syntax

```
TYPE plan_object_list IS TABLE OF generic_plan_object;
```

The generic object abstracts the common attributes of plans from all plan sources. Every plan source is a subclass of the generic_plan_object superclass. The following table summarizes the different plan sources. Note that when an optional parameter is null, it can correspond to multiple objects. For example, if you do not specify a child number for cursor_cache_object, then it matches all cursor cache statements with the specified SQL ID.

Table 242-1 Plan Sources for PLAN_OBJECT_LIST

Plan Source	Specification	Description
Plan table	`plan_table_object(owner, plan_table_name, statement_id, plan_id)`	The parameters are as follows: • `owner`—The owner of the plan table • `plan_table_name`—The name of the plan table • `statement_id`—The ID of the statement (optional) • `plan_id`—The ID of the plan (optional)
Cursor cache	`cursor_cache_object(sql_id, child_number)`	The parameters are as follows: • `sql_id`—The SQL ID of the plan • `child_number`—The child number of the plan in the cursor cache (optional)
AWR	`awr_object(sql_id, dbid, con_dbid, plan_hash_value)`	The parameters are as follows: • `sql_id`—The SQL ID of the plan • `dbid`—The database ID (optional) • `con_dbid`—The CDB ID (optional) • `plan_hash_value`—The hash value of the plan (optional)
SQL tuning set	`sqlset_object (sqlset_owner, sqlset_name, sql_id, plan_hash_value)`	The parameters are as follows: • `sqlset_owner`—The owner of the SQL tuning set • `sqlset_name`—The name of the SQL tuning set • `sql_id`—The SQL ID of the plan • `plan_hash_value`—The hash value of the plan (optional)
SQL plan management	`spm_object (sql_handle, plan_name)`	The parameters are as follows: • `sql_handle`—The SQL handle of plans protected by SQL plan management • `plan_name`—The name of the SQL plan baseline (optional)
SQL profile	`sql_profile_object (profile_name)`	The `profile_name` parameter specifies the name of the SQL profile.
Advisor	`advisor_object (task_name, execution_name, sql_id, plan_id)`	The parameters are as follows: • `task_name`—The name of the advisor task • `execution_name`—The name of the task execution • `sql_id`—The SQL ID of the plan • `plan_id`—The advisor plan ID (optional)

Examples

These examples show sample uses of DBMS_XPLAN.

Displaying a Plan Table Using DBMS_XPLAN.DISPLAY

Execute an explain plan command on a `SELECT` statement:

```
EXPLAIN PLAN FOR
SELECT * FROM emp e, dept d
   WHERE e.deptno = d.deptno
   AND e.ename='benoit';
```

Display the plan using the `DBMS_XPLAN.DISPLAY` table function

```
SET LINESIZE 130
SET PAGESIZE 0
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

This query produces the following output:

```
Plan hash value: 3693697075
--------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    57 |     6  (34)| 00:00:01 |
|*  1 |  HASH JOIN         |      |     1 |    57 |     6  (34)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| EMP  |     1 |    37 |     3  (34)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| DEPT |     4 |    80 |     3  (34)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("E"."DEPTNO"="D"."DEPTNO")
   2 - filter("E"."ENAME"='benoit')

15 rows selected.
```

Displaying a Cursor Execution Plan Using DBMS_XPLAN.DISPLAY_CURSOR

By default, the table function `DISPLAY_CURSOR` formats the execution plan for the last SQL statement executed by the session. For example:

```
SELECT ename  FROM  emp e, dept d
   WHERE   e.deptno = d.deptno
   AND   e.empno=7369;

ENAME
----------
SMITH
```

To display the execution plan of the last executed statement for that session:

```
SET PAGESIZE 0
SELECT * FROM DBMS_XPLAN.DISPLAY_CURSOR();
```

This query produces the following output:

```
Plan hash value: 3693697075, SQL hash value: 2096952573, child number: 0
-------------------------------------------------------------------
SELECT ename FROM emp e, dept d WHERE e.deptno = d.deptno
```

```
AND e.empno=7369

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |            |          |
|*  1 |  HASH JOIN         |      |     1 |    16 |     6  (34)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| EMP  |     1 |    13 |     3  (34)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| DEPT |     4 |    12 |     3  (34)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("E"."DEPTNO"="D"."DEPTNO")
   2 - filter("E"."EMPNO"=7369)

21 rows selected.
```

You can also use the table function `DISPLAY_CURSOR` to display the execution plan for any loaded cursor stored in the cursor cache. In that case, you must supply a reference to the child cursor to the table function. This includes the SQL ID of the statement and optionally the child number.

Run a query with a distinctive comment:

```
SELECT /* TOTO */ ename, dname
FROM dept d join emp e USING (deptno);
```

Get `sql_id` and `child_number` for the preceding statement:

```
SELECT sql_id, child_number
FROM v$sql
WHERE sql_text LIKE '%TOTO%';

SQL_ID         CHILD_NUMBER
----------     ------------------------------
gwp663cqh5qbf   0
```

Display the execution plan for the cursor:

```
SELECT * FROM DBMS_XPLAN.DISPLAY_CURSOR('gwp663cqh5qbf',0);

Plan hash value: 3693697075, SQL ID: gwp663cqh5qbf, child number: 0
-------------------------------------------------------------
SELECT /* TOTO */ ename, dname
FROM dept d JOIN emp e USING (deptno);

-----------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |       |       |     7 (100)|          |
|   1 |  SORT GROUP BY      |      |     4 |    64 |     7  (43)| 00:00:01 |
|*  2 |   HASH JOIN         |      |    14 |   224 |     6  (34)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| DEPT |     4 |    44 |     3  (34)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| EMP  |    14 |    70 |     3  (34)| 00:00:01 |
-----------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("E"."DEPTNO"="D"."DEPTNO")
```

Instead of issuing two queries, one to the get the sql_id and child_number pair and one to display the plan, you can combine these in a single query:

Display the execution plan of all cursors matching the string 'TOTO':

```
SELECT t.*
FROM v$sql s, DBMS_XPLAN.DISPLAY_CURSOR(s.sql_id, s.child_number) t WHERE sql_text LIKE
'%TOTO%';
```

Displaying a Plan Table with Parallel Information

By default, only relevant information is reported by the display and `display_cursor` table functions. In Displaying a Plan Table Using DBMS_XPLAN.DISPLAY, the query does not execute in parallel. Hence, information related to the parallelization of the plan is not reported. As shown in the following example, parallel information is reported only if the query executes in parallel.

```
ALTER TABLE emp PARALLEL;
EXPLAIN PLAN for
SELECT * FROM emp e, dept d
   WHERE e.deptno = d.deptno
   AND e.ename    ='hermann'
   ORDER BY e.empno;
```

Display the plan using the `DBMS_XPLAN.DISPLAY` table function

```
SET LINESIZE 130
SET PAGESIZE 0
SELECT * FROM DBMS_XPLAN.DISPLAY();
Plan hash value: 3693697345
```

```
-----------------------------------------------------------------------------------------------------------------
| Id | Operation          | Name    | Rows | Bytes | Cost (%CPU)| Time     | TQ    |INOUT |PQ Distrib   |
-----------------------------------------------------------------------------------------------------------------
| 0  | SELECT STATEMENT   |         | 1    | 117   | 6 (50)     | 00:00:01 |       |      |             |
| 1  | PX COORDINATOR     |         |      |       |            |          |       |      |             |
| 2  | PX SEND QC (ORDER) |:TQ10003 | 1    | 117   | 6 (50)     | 00:00:01 | Q1,03 | P->S | QC (ORDER)  |
| 3  | SORT ORDER BY      |         | 1    | 117   | 6 (50)     | 00:00:01 | Q1,03 | PCWP |             |
| 4  | PX RECEIVE         |         | 1    | 117   | 5 (40)     | 00:00:01 | Q1,03 | PCWP |             |
| 5  | PX SEND RANGE      |:TQ10002 | 1    | 117   | 5 (40)     | 00:00:01 | Q1,02 | P->P | RANGE       |
|* 6 | HASH JOIN          |         | 1    | 117   | 5 (40)     | 00:00:01 | Q1,02 | PCWP |             |
| 7  | PX RECEIVE         |         | 1    | 87    | 2 (50)     | 00:00:01 | Q1,02 | PCWP |             |
| 8  | PX SEND HASH       |:TQ10001 | 1    | 87    | 2 (50)     | 00:00:01 | Q1,01 | P->P | HASH        |
| 9  | PX BLOCK ITERATOR  |         | 1    | 87    | 2 (50)     | 00:00:01 | Q1,01 | PCWC |             |
|* 10| TABLE ACCESS FULL  | EMP     | 1    | 87    | 2 (50)     | 00:00:01 | Q1,01 | PCWP |             |
| 11 | BUFFER SORT        |         |      |       |            |          | Q1,02 | PCWC |             |
| 12 | PX RECEIVE         |         | 4    | 120   | 3 (34)     | 00:00:01 | Q1,02 | PCWP |             |
| 13 | PX SEND HASH       |:TQ10000 | 4    | 120   | 3 (34)     | 00:00:01 |       | S->P | HASH        |
| 14 | TABLE ACCESS FULL  | DEPT    | 4    | 120   | 3 (34)     | 00:00:01 |       |      |             |
-----------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
6 - access("E"."DEPTNO"="D"."DEPTNO")
10 - filter("E"."ENAME"='hermann')
---------------------------------------------------
```

When the query is parallel, information related to parallelism is reported: table queue number (`TQ` column), table queue type (`INOUT`) and table queue distribution method (`PQ Distrib`).

By default, if several plans in the plan table match the `statement_id` parameter passed to the display table function (default value is `NULL`), only the plan corresponding to the last `EXPLAIN PLAN` command is displayed. Hence, there is no need to purge the plan table after each `EXPLAIN PLAN`. However, you should purge the plan table regularly to ensure good performance in the execution of the `DISPLAY` table function. If no plan table is created, Oracle uses a global

temporary table to store any plan information for individual users and preserves its content throughout the lifespan of a session. Note that you cannot truncate the content of a global temporary table.

For ease of use, you can define a view on top of the display table function and then use that view to display the output of the `EXPLAIN PLAN` command:

Using a View to Display Last Explain Plan

```
# define plan view
CREATE VIEW PLAN AS SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

# display the output of the last explain plan command
SELECT * FROM PLAN;
```

Summary of DBMS_XPLAN Subprograms

This table lists the DBMS_XPLAN subprograms and briefly describes them.

Table 242-2 ***DBMS_XPLAN Package Subprograms***

Subprogram	Description
COMPARE_PLANS Function	Compares each plan in a list with a reference plan and returns the report
DIFF_PLAN Function	Compares plans
DISPLAY Function	Displays the contents of the plan table
DISPLAY_AWR Function	Displays the contents of an execution plan stored in the AWR
DISPLAY_CURSOR Function	Displays the execution plan of any cursor in the cursor cache
DISPLAY_PLAN Function	Displays the contents of the plan table in a variety of formats with `CLOB` output type
DISPLAY_SQL_PLAN_BASELINE Function	Displays one or more execution plans for the specified SQL handle of a SQL plan baseline
DISPLAY_SQLSET Function	Displays the execution plan of a given statement stored in a SQL tuning set

COMPARE_PLANS Function

This function compares each plan in a list with a reference plan and returns the report.

Syntax

```
DBMS_XPLAN.COMPARE_PLANS(
   reference_plan    IN generic_plan_object,
   compare_plan_list IN plan_object_list,
   type              IN VARCHAR2 := 'TEXT',
   level             IN VARCHAR2 := 'TYPICAL',
   section           IN VARCHAR2 := 'ALL')
 RETURN CLOB;
```

Parameters

Table 242-3 COMPARE_PLANS Function Parameters

Parameter	Description
reference_plan	The reference plan. This plan should always evaluate to a single plan.
compare_plan_list	List of plans to compare with reference plan. The compare_plan_list is a list of generic_object and each generic_object could correspond to one or more plans.
type	Type of the report. Possible values are: • TEXT • HTML • XML
level	Format of the report. Possible values are: • BASIC • TYPICAL • ALL
section	A particular section in the report. Possible values are: • SUMMARY • FINDINGS • PLANS • INFORMATION • ERRORS

Example 242-1 Examples

The following examples illustrate the usage of COMPARE_PLANS Function.

```
var report clob;
exec :report := dbms_xplan.compare_plans(cursor_cache_object('8mkxm7ur07za0', 2),
                - plan_object_list (cursor_cache_object('8mkxm7ur07za0', 4)));

print report
```

The above example compares the plan of child cursor number 2 for the SQL ID '8mkxm7ur07za0' with that of the child cursor number 4 for the same SQL ID. Returns the report in the text format (default).

```
var report clob;
exec :report := dbms_xplan.compare_plans(cursor_cache_object('8mkxm7ur07za0', 2),
                plan_object_list( spm_object('SQL_024d0f7d21351f5d',
'SQL_PLAN_sdfjkd')));

print report
```

The above example compares the plan of child cursor number 2 for the SQL ID '8mkxm7ur07za0' with that of the plan baseline captured by SPM for query whose SQL handle is 'SQL_024d0f7d21351f5d' and plan name is 'SQL_PLAN_sdfjkd'. Returns the report in the text format (default).

```
var report clob;
exec :report = dbms_xplan.compare_plans(cursor_cache_object('8mkxm7ur07za0', 2),
```

```
              plan_object_list( cursor_cache_object('8mkxm7ur07za0'),
sqlset_object('SH', 'SQLT_WORKLOAD', '               6vfqvav0rgyad'),
awr_object('6vfqvav0rgyad', 5), spm_object('SQL_024d0f7d21351f5d',
'SQL_PLAN_               sdfjkd'), plan_table_object('SH', 'plan_table', NULL, 38),
sql_profile_object('pe3r3ejsfd'), a              dvisor_object('TASK_1228',
'EXEC_1928', '8mkxm7ur07za0')), type => 'XML',
              level => 'ALL', section => 'SUMMARY');
```

The above example compares the plan of child cursor number 2 for the SQL ID '8mkxm7ur07za0' with each of the plans in the following list:

- cursor_cache_object('8mkxm7ur07za0'): All the plans in the cursor cache that are generated for the SQL ID '8mkxm7ur07za0'.
- sqlset_object('SH', 'SQLT_WORKLOAD', '6vfqvav0rgyad'): All the plans generated in the SQL tuning set SH. SQLT_WORKLOAD for the SQL ID '6vfqvav0rgyad'.
- awr_object('6vfqvav0rgyad', 5): All the plans in AWR that are captured for database ID 5 and SQL ID '6vfqvav0rgyad'.
- spm_object ('SQL_024d0f7d21351f5d, 'SQL_PLAN_sdfjkd'): The plan baseline for the query with SQL handle 'SQL_024d0f7d21351f5d' with name 'SQL_PLAN_sdfjkd'.
- plan_table_object('SH', 'plan_table', NULL, 38): Plan stored in SH.plan_table identified by plan_id=38.
- sql_profile_object('pe3r3ejsfd'): Plan identified by the SQL profile name 'pe3r3ejsfd'.
- advisor_object('TASK_1228', 'EXEC_1928', '8mkxm7ur07za0')): All the plans stored in SQL advisor identified by task name 'TASK_1228', execution name 'EXEC_1928' and SQL ID '8mkxm7ur07za0'.

DIFF_PLAN Function

This function compares two sql plans, the reference plan and the target plan. This function returns a `task_id` that can be used to retrieve the report of findings.

Syntax

```
DBMS_XPLAN.DIFF_PLAN(
   sql_text     IN   CLOB,
   outline      IN   CLOB,
   user_name    IN   VARCHAR2 := 'NULL')
 RETURN VARCHAR2;
```

Parameters

Table 242-4 DIFF_PLAN Function Parameters

Parameter	Description
`sql_text`	The text of the SQL statement.
`outline`	Used to generate the target plan.
`user_name`	The parsing schema name default to current user.

DISPLAY Function

This table function displays the contents of the plan table.

In addition, you can use this table function to display any plan (with or without statistics) stored in a table as long as the columns of this table are named the same as columns of the plan table (or `V$SQL_PLAN_STATISTICS_ALL` if statistics are included). You can apply a predicate on the specified table to select rows of the plan to display.

Syntax

```
DBMS_XPLAN.DISPLAY(
   table_name     IN  VARCHAR2  DEFAULT 'PLAN_TABLE',
   statement_id   IN  VARCHAR2  DEFAULT  NULL,
   format         IN  VARCHAR2  DEFAULT  'TYPICAL',
   filter_preds   IN  VARCHAR2 DEFAULT NULL);
```

Parameters

Table 242-5 DISPLAY Function Parameters

Parameter	Description
`table_name`	Specifies the table name where the plan is stored. This parameter defaults to `PLAN_TABLE`, which is the default plan table for the `EXPLAIN PLAN` command. If `NULL` is specified it also defaults to `PLAN_TABLE`.
`statement_id`	Specifies the `statement_id` of the plan to be displayed. This parameter defaults to `NULL`, which is the default when the `EXPLAIN PLAN` command is executed without a `set statement_id` clause. If no `statement_id` is specified, the function shows you the plan of the most recent explained statement.

Table 242-5 (Cont.) DISPLAY Function Parameters

Parameter	Description
format	Controls the level of details for the plan. It accepts the following values: • BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option. • TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS, and REMOTE SQL information (see below). • SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel. • ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed). For finer control on the display output, the following keywords can be added to the above three standard format options to customize their default behavior. Each keyword either represents a logical group of plan table columns (such as PARTITION) or logical additions to the base plan table output (such as PREDICATE). Format keywords must be separated by either a comma or a space: • ROWS - if relevant, shows the number of rows estimated by the optimizer • BYTES - if relevant, shows the number of bytes estimated by the optimizer • COST - if relevant, shows optimizer cost information • PARTITION - if relevant, shows partition pruning information • PARALLEL - if relevant, shows PX information (distribution method and table queue information) • PREDICATE - if relevant, shows the predicate section • PROJECTION -if relevant, shows the projection section • ALIAS - if relevant, shows the "Query Block Name / Object Alias" section • REMOTE - if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL) • NOTE - if relevant, shows the note section of the explain plan Format keywords can be prefixed by the sign '-' to exclude the specified information. For example, '-PROJECTION' excludes projection information. If the target plan table (see table_name parameter) also stores plan statistics columns (for example, it is a table used to capture the content of the fixed view V$SQL_PLAN_STATISTICS_ALL), additional format keywords can be used to specify which class of statistics to display when using the DISPLAY Function. These additional format keywords are IOSTATS, MEMSTATS, ALLSTATS, and LAST (see the DISPLAY_CURSOR Function or the DISPLAY_SQLSET Function for a full description of these four keywords).
filter_preds	SQL filter predicate(s) to restrict the set of rows selected from the table where the plan is stored. When value is NULL (the default), the plan displayed corresponds to the last executed explain plan. For example: filter_preds=>'plan_id = 10' Can reference any column of the table where the plan is stored and can contain any SQL construct (for example, sub-query, function calls (see **WARNING** under Usage Notes)

Usage Notes

Here are some ways you might use variations on the format parameter:

- Use 'ALL -PROJECTION -NOTE' to display everything except the projection and note sections.
- Use 'TYPICAL PROJECTION' to display using the typical format with the additional projection section (which is normally excluded under the typical format). Since typical is default, using simply 'PROJECTION' is equivalent.
- Use '-BYTES -COST -PREDICATE' to display using the typical format but excluding optimizer cost and byte estimates as well as the predicate section.
- Use 'BASIC ROWS' to display basic information with the additional number of rows estimated by the optimizer.

> **WARNING:**
>
> Application developers should expose the `filter_preds` parameter to end-users only after careful consideration because this could expose the application to SQL injection. Indeed, `filter_preds` can potentially reference any table or execute any server function for which the database user invoking the table function has privileges.

Examples

To display the result of the last `EXPLAIN PLAN` command stored in the plan table:

```
SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY);
```

To display from other than the default plan table, "`my_plan_table`":

```
SELECT * FROM table (DBMS_XPLAN.DISPLAY('my_plan_table'));
```

To display the minimum plan information:

```
SELECT * FROM table (DBMS_XPLAN.DISPLAY('plan_table', null, 'basic'));
```

To display the plan for a statement identified by '`foo`', such as `statement_id='sales_query'`:

```
SELECT * FROM table (DBMS_XPLAN.DISPLAY('plan_table', 'sales_query'));
```

DISPLAY_AWR Function

This table function displays the contents of an execution plan stored in AWR.

> **Note:**
>
> This function is deprecated. Use `DISPLAY_WORKLOAD_REPOSITORY` instead. `DISPLAY_AWR` only works with snapshots for the local DBID, whereas `DISPLAY_WORKLOAD_REPOSITORY` supports all snapshots inside AWR, including remote and imported snapshots.

Syntax

```
DBMS_XPLAN.DISPLAY_AWR(
   sql_id            IN      VARCHAR2,
   plan_hash_value   IN      NUMBER DEFAULT NULL,
   db_id             IN      NUMBER DEFAULT NULL,
   format            IN      VARCHAR2 DEFAULT TYPICAL);
```

Parameters

Table 242-6 DISPLAY_AWR Table Function Parameters

Parameter	Description
sql_id	Specifies the SQL_ID of the SQL statement. You can retrieve the appropriate value for the SQL statement of interest by querying the column SQL_ID in DBA_HIST_SQLTEXT.
plan_hash_value	Specifies the PLAN_HASH_VALUE of a SQL statement. This parameter is optional. If omitted, the table function returns all stored execution plans for a given SQL_ID.
db_id	Specifies the database_id for which the plan of the SQL statement, identified by SQL_ID should be displayed. If not supplied, the database_id of the local database is used, as shown in V$DATABASE.

Table 242-6 (Cont.) DISPLAY_AWR Table Function Parameters

Parameter	Description
format	Controls the level of details for the plan. It accepts four values: • BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option. • TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS and REMOTE SQL information (see below). • SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel. • ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed). For finer control on the display output, the following keywords can be added to the above four standard format options to customize their default behavior. Each keyword either represents a logical group of plan table columns (such as PARTITION) or logical additions to the base plan table output (such as PREDICATE). Format keywords must be separated by either a comma or a space: • ROWS - if relevant, shows the number of rows estimated by the optimizer • BYTES - if relevant, shows the number of bytes estimated by the optimizer • COST - if relevant, shows optimizer cost information • PARTITION - if relevant, shows partition pruning information • PARALLEL - if relevant, shows PX information (distribution method and table queue information) • PREDICATE - if relevant, shows the predicate section • PROJECTION -if relevant, shows the projection section • ALIAS - if relevant, shows the "Query Block Name / Object Alias" section • REMOTE - if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL) • NOTE - if relevant, shows the note section of the explain plan Format keywords can be prefixed by the sign '-' to exclude the specified information. For example, '-PROJECTION' excludes projection information.

Usage Notes

- To use the DISPLAY_AWR functionality, the calling user must have SELECT or READ privilege on DBA_HIST_SQL_PLAN, AWR_ROOT_SQL_PLAN, AWR_PDB_SQL_PLAN, DBA_HIST_SQLTEXT, AWR_ROOT_SQLTEXT, AWR_PDB_SQLTEXT, and V$DATABASE, otherwise it shows an appropriate error message. By default, select privilege for these views is granted to the select_catalog role.
- The following examples show different ways of using the format parameter:
  - Use 'BASIC ROWS' to display basic information with the additional number of rows estimated by the optimizer.
  - Use 'ALL -PROJECTION -NOTE' to display everything except the projection and note sections.

- Use 'TYPICAL PROJECTION' to display using the typical format with the additional projection section (which is normally excluded under the typical format). Since typical is default, using simply 'PROJECTION' is equivalent.
- Use '-BYTES -COST -PREDICATE' to display using the typical format but excluding optimizer cost and byte estimates and the predicate section.

Examples

To display the different execution plans associated with the SQL ID 'atfwcg8anrykp':

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY_AWR('atfwcg8anrykp'));
```

To display all execution plans of all stored SQL statements containing the string 'TOTO':

```
SELECT tf.*
FROM   DBA_HIST_SQLTEXT ht, table(DBMS_XPLAN.DISPLAY_AWR(ht.sql_id,null,
null,  'ALL' )) tf
WHERE  ht.sql_text like '%TOTO%';
```

DISPLAY_CURSOR Function

This table function displays the explain plan of any cursor loaded in the cursor cache. In addition to the explain plan, various plan statistics (such as. I/O, memory and timing) can be reported (based on the V$SQL_PLAN_STATISTICS_ALL VIEWS).

Syntax

```
DBMS_XPLAN.DISPLAY_CURSOR(
   sql_id            IN  VARCHAR2  DEFAULT  NULL,
   cursor_child_no   IN  NUMBER    DEFAULT  0,
   format            IN  VARCHAR2  DEFAULT  'TYPICAL');
```

Parameters

Table 242-7 DISPLAY_CURSOR Function Parameters

Parameter	Description
sql_id	Specifies the SQL_ID of the SQL statement in the cursor cache. You can retrieve the appropriate value by querying the column SQL_ID in V$SQL or V$SQLAREA. Alternatively, you could choose the column PREV_SQL_ID for a specific session out of V$SESSION. This parameter defaults to NULL in which case the plan of the last cursor executed by the session is displayed.
cursor_child_no	Child number of the cursor to display. If not supplied, the execution plan of the child_number=0 cursor matching the supplied sql_id parameter are displayed. The child_number can be specified only if sql_id is specified.

Table 242-7 (Cont.) DISPLAY_CURSOR Function Parameters

Parameter	Description
format	Controls the level of details for the plan. It accepts five values: • BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option. • TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS and REMOTE SQL information (see below). • SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel. • ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed). • ADAPTIVE: – Displays the final plan, or the current plan if the execution has not completed. This section includes notes about runtime optimizations that affect the plan, such as switching from a Nested Loops join to a Hash join. – Plan lineage. This section shows the plans that were run previously due to automatic reoptimization. It also shows the default plan, if the plan changed due to dynamic plans. – Recommended plan. In reporting mode, the plan is chosen based on execution statistics displayed. Note that displaying the recommended plan for automatic reoptimization requires re-compiling the query with the optimizer adjustments collected in the child cursor. Displaying the recommended plan for a dynamic plan does not require this. – Dynamic plans. This summarizes the portions of the plan that differ from the default plan chosen by the optimizer. For finer control on the display output, you can add the following keywords to the preceding format options to customize their default behavior. Each keyword either represents a logical group of plan table columns (such as PARTITION) or logical additions to the base plan table output (such as PREDICATE). Format keywords must be separated by either a comma or a space: • ROWS - if relevant, shows the number of rows estimated by the optimizer • BYTES - if relevant, shows the number of bytes estimated by the optimizer • COST - if relevant, shows optimizer cost information • PARTITION - if relevant, shows partition pruning information • PARALLEL - if relevant, shows PX information (distribution method and table queue information) • PREDICATE - if relevant, shows the predicate section • PROJECTION -if relevant, shows the projection section • ALIAS - if relevant, shows the "Query Block Name / Object Alias" section • REMOTE - if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL) • NOTE - if relevant, shows the note section of the explain plan • IOSTATS - assuming that basic plan statistics are collected when SQL statements are executed (either by using the gather_plan_statistics hint or by setting the parameter

Table 242-7 (Cont.) DISPLAY_CURSOR Function Parameters

Parameter	Description
	statistics_level to ALL), this format shows IO statistics for `ALL` (or only for the `LAST` as shown below) executions of the cursor. • `MEMSTATS` - Assuming that PGA memory management is enabled (that is, `pga_aggregate_target` parameter is set to a non 0 value), this format allows to display memory management statistics (for example, execution mode of the operator, how much memory was used, number of bytes spilled to disk, and so on). These statistics only apply to memory intensive operations like hash-joins, sort or some bitmap operators. • `ALLSTATS` - A shortcut for `'IOSTATS MEMSTATS'` • `LAST` - By default, plan statistics are shown for all executions of the cursor. The keyword `LAST` can be specified to see only the statistics for the last execution. The following formats are deprecated but supported for backward compatibility: • `RUNSTATS_TOT` - Same as `IOSTATS`, that is, displays IO statistics for all executions of the specified cursor. • `RUNSTATS_LAST` - Same as `IOSTATS LAST`, that is, displays the runtime statistics for the last execution of the cursor You can prefix format keywords with the sign '`-`' to exclude the specified information. For example, '`-PROJECTION`' excludes projection information.

Usage Notes

- To use the `DISPLAY_CURSOR` functionality, the calling user must have `SELECT` or `READ` privilege on the fixed views `V$SQL_PLAN_STATISTICS_ALL`, `V$SQL` and `V$SQL_PLAN`, otherwise it shows an appropriate error message.
- Here are some ways you might use variations on the `format` parameter:
  - Use '`ALL -PROJECTION -NOTE`' to display everything except the projection and note sections.
  - Use '`TYPICAL PROJECTION`' to display using the typical format with the additional projection section (which is normally excluded under the typical format). Since typical is default, using simply '`PROJECTION`' is equivalent.
  - Use '`-BYTES -COST -PREDICATE`' to display using the typical format but excluding optimizer cost and byte estimates as well as the predicate section.
  - Use '`BASIC ROWS`' to display basic information with the additional number of rows estimated by the optimizer.

Examples

To display the execution plan of the last `SQL` statement executed by the current session:

```
SELECT * FROM table (
   DBMS_XPLAN.DISPLAY_CURSOR);
```

To display the execution plan of all children associated with the SQL ID 'atfwcg8anrykp':

```
SELECT * FROM table (
   DBMS_XPLAN.DISPLAY_CURSOR('atfwcg8anrykp'));
```

To display runtime statistics for the cursor included in the preceding statement:

```
SELECT * FROM table (
   DBMS_XPLAN.DISPLAY_CURSOR('atfwcg8anrykp', NULL, 'ALLSTATS LAST');
```

DISPLAY_PLAN Function

This table function displays the contents of the plan table in a variety of formats with CLOB output type.

Syntax

```
DBMS_XPLAN.DISPLAY_PLAN (
   table_name       IN    VARCHAR2   DEFAULT 'PLAN_TABLE',
   statement_id     IN    VARCHAR2   DEFAULT NULL,
   format           IN    VARCHAR2   DEFAULT 'TYPICAL',
   filter_preds     IN    VARCHAR2   DEFAULT NULL,
   type             IN    VARCHAR2   DEFAULT 'TEXT')
  RETURN CLOB;
```

Parameters

Table 242-8 DISPLAY_PLAN Function Parameters

Parameter	Description
table_name	Specifies the table name where the plan is stored. This parameter defaults to PLAN_TABLE, which is the default plan table for the EXPLAIN PLAN command. If NULL is specified it also defaults to PLAN_TABLE.
statement_id	Specifies the statement_id of the plan to be displayed. This parameter defaults to NULL, which is the default when the EXPLAIN PLAN command is executed without a set statement_id clause.If no statement_id is specified, the function shows you the plan of the most recent explained statement.
filter_preds	SQL filter predicate(s) to restrict the set of rows selected from the table where the plan is stored. When value is NULL (the default), the plan displayed corresponds to the last executed explain plan. For example: filter_preds=>'plan_id = 10' Can reference any column of the table where the plan is stored and can contain any SQL construct (for example, sub-query, function calls (see **WARNING** under Usage Notes)

Table 242-8 (Cont.) DISPLAY_PLAN Function Parameters

Parameter	Description
format	Controls the level of details for the plan. It accepts five values: • BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option. • TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS and REMOTE SQL information (see below). • SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel. • ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed). • ADAPTIVE: Displays the default plan, and for each dynamic subplan (if stipulated): - A list of the rowsources from the original which may be replaced, and the rowsources to replace them - If outline display is specified in the format argument, the hints for each option in the dynamic subplan are displayed For finer control on the display output, the following keywords can be added to the above three standard format options to customize their default behavior. Each keyword either represents a logical group of plan table columns (such as PARTITION) or logical additions to the base plan table output (such as PREDICATE). Format keywords must be separated by either a comma or a space: • ROWS - if relevant, shows the number of rows estimated by the optimizer • BYTES - if relevant, shows the number of bytes estimated by the optimizer • COST - if relevant, shows optimizer cost information • PARTITION - if relevant, shows partition pruning information • PARALLEL - if relevant, shows PX information (distribution method and table queue information) • PREDICATE - if relevant, shows the predicate section • PROJECTION -if relevant, shows the projection section • ALIAS - if relevant, shows the "Query Block Name / Object Alias" section • REMOTE - if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL) • NOTE - if relevant, shows the note section of the explain plan Format keywords can be prefixed by the sign '-' to exclude the specified information. For example, '-PROJECTION' excludes projection information. If the target plan table (see table_name parameter) also stores plan statistics columns (for example, it is a table used to capture the content of the fixed view V$SQL_PLAN_STATISTICS_ALL), additional format keywords can be used to specify which class of statistics to display when using the DISPLAY Function. These additional format keywords are IOSTATS, MEMSTATS, ALLSTATS and LAST (see the DISPLAY_CURSOR Function or the DISPLAY_SQLSET Function for a full description of these four keywords).
type	Output type, one of: 'TEXT', 'ACTIVE', 'HTML', or 'XML' (see Usage Notes regarding type ACTIVE). '

Return Values

Returns the requested report as `CLOB`

Usage Notes

Active reports have a rich, interactive user interface akin to that found in 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.

> **WARNING:**
>
> Application developers should expose the `filter_preds` parameter to end-users only after careful consideration because this could expose the application to SQL injection. Indeed, `filter_preds` can potentially reference any table or execute any server function for which the database user invoking the table function has privileges.

DISPLAY_SQL_PLAN_BASELINE Function

This table function displays one or more execution plans for the specified SQL handle of a SQL plan baseline.

Syntax

```
DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE (
   sql_handle      IN VARCHAR2 := NULL,
   plan_name       IN VARCHAR2 := NULL,
   format          IN VARCHAR2 := 'TYPICAL')
 RETURN dbms_xplan_type_table;
```

Parameters

Table 242-9 DISPLAY_SQL_PLAN_BASELINE Function Parameters

Parameter	Description
`sql_handle`	SQL statement handle. It identifies a SQL statement whose plans are to be displayed.
`plan_name`	Plan name. It identifies a specific plan. Default NULL means all plans associated with identified SQL statement are explained and displayed.
`format`	Format string determines what information stored in the plan displayed. The following format values are possible, each representing a common use case: `BASIC`, `TYPICAL`, and `ALL`.

Return Values

A PL/SQL type table

Usage Notes

This function uses plan information stored in the plan baseline to explain and display the plans. The `plan_id` stored in the SQL management base may not match the `plan_id` of the generated plan. A mismatch between the stored `plan_id` and generated `plan_id` means that it is a non-reproducible plan. Such a plan is deemed invalid and is bypassed by the optimizer during SQL compilation.

Examples

Display all plans of a SQL statement identified by the SQL handle `SYS_SQL_b1d49f6074ab95af` using `TYPICAL` format

```
SET LINESIZE 150
SET PAGESIZE 2000
SELECT t.*
FROM TABLE(DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE('SYS_SQL_b1d49f6074ab95af'))
t;
```

Display all plans of one or more SQL statements containing the string `HR2` using `BASIC` format:

```
SET LINESIZE 150
SET PAGESIZE 2000
SELECT t.*
FROM (SELECT DISTINCT sql_handle FROM dba_sql_plan_baselines WHERE sql_text
LIKE '%HR2%') pb,
     TABLE(DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE(pb.sql_handle, NULL,
'BASIC')) t;
```

DISPLAY_SQLSET Function

This table function displays the execution plan of a given statement stored in a SQL tuning set.

Syntax

```
DBMS_XPLAN.DISPLAY_SQLSET(
   sqlset_name      IN  VARCHAR2,
   sql_id           IN  VARCHAR2,
   plan_hash_value  IN NUMBER := NULL,
   format           IN  VARCHAR2  := 'TYPICAL',
   sqlset_owner     IN  VARCHAR2  := NULL)
  RETURN DBMS_XPLAN_TYPE_TABLE PIPELINED;
```

Parameters

Table 242-10 DISPLAY_SQLSET Function Parameters

Parameter	Description
`sqlset_name`	Name of the SQL Tuning Set

Table 242-10 (Cont.) DISPLAY_SQLSET Function Parameters

Parameter	Description
`sql_id`	Specifies the sql_id value for a SQL statement having its plan stored in the SQL tuning set. You can find all stored SQL statements by querying table function `DBMS_SQLTUNE.SELECT_SQLSET`
`plan_hash_value`	Optional parameter. Identifies a specific stored execution plan for a SQL statement. If suppressed, all stored execution plans are shown.

Table 242-10 (Cont.) DISPLAY_SQLSET Function Parameters

Parameter	Description
format	Controls the level of details for the plan. It accepts four values: • BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option. • TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS and REMOTE SQL information (see below). • SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel. • ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed). For finer control on the display output, the following keywords can be added to the above three standard format options to customize their default behavior. Each keyword either represents a logical group of plan table columns (such as PARTITION) or logical additions to the base plan table output (such as PREDICATE). Format keywords must be separated by either a comma or a space: • ROWS - if relevant, shows the number of rows estimated by the optimizer • BYTES - if relevant, shows the number of bytes estimated by the optimizer • COST - if relevant, shows optimizer cost information • PARTITION - if relevant, shows partition pruning information • PARALLEL - if relevant, shows PX information (distribution method and table queue information) • PREDICATE - if relevant, shows the predicate section • PROJECTION -if relevant, shows the projection section • ALIAS - if relevant, shows the "Query Block Name / Object Alias" section • REMOTE - if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL) • NOTE - if relevant, shows the note section of the explain plan • IOSTATS - assuming that basic plan statistics are collected when SQL statements are executed (either by using the gather_plan_statistics hint or by setting the parameter STATISTICS_LEVEL to ALL), this format shows IO statistics for ALL (or only for the LAST as shown below) executions of the cursor. • MEMSTATS - Assuming that PGA memory management is enabled (that is, pga_aggregate_target parameter is set to a non 0 value), this format allows to display memory management statistics (for example, execution mode of the operator, how much memory was used, number of bytes spilled to disk, and so on). These statistics only apply to memory intensive operations like hash-joins, sort or some bitmap operators. • ALLSTATS - A shortcut for 'IOSTATS MEMSTATS' • LAST - By default, plan statistics are shown for all executions of the cursor. The keyword LAST can be specified to see only the statistics for the last execution. The following two formats are deprecated but supported for backward compatibility: • RUNSTATS_TOT - Same as IOSTATS, that is, displays IO statistics for all executions of the specified cursor.

Table 242-10 (Cont.) DISPLAY_SQLSET Function Parameters

Parameter	Description
	• RUNSTATS_LAST - Same as IOSTATS LAST, that is, displays the runtime statistics for the last execution of the cursor Format keywords can be prefixed by the sign '-' to exclude the specified information. For example, '-PROJECTION' excludes projection information.
sqlset_owner	The owner of the SQL tuning set. The default is the current user.

Usage Notes

Here are some ways you might use variations on the format parameter:

- Use 'ALL -PROJECTION -NOTE' to display everything except the projection and note sections.
- Use 'TYPICAL PROJECTION' to display using the typical format with the additional projection section (which is normally excluded under the typical format). Since typical is default, using simply 'PROJECTION' is equivalent.
- Use '-BYTES -COST -PREDICATE' to display using the typical format but excluding optimizer cost and byte estimates as well as the predicate section.
- Use 'BASIC ROWS' to display basic information with the additional number of rows estimated by the optimizer.

Examples

To display the execution plan for the SQL statement associated with SQL ID 'gwp663cqh5qbf' and PLAN HASH 3693697075 in the SQL Tuning Set called 'OLTP_optimization_0405':

```
SELECT * FROM table
(DBMS_XPLAN.DISPLAY_SQLSET('OLTP_optimization_0405','gwp663cqh5qbf',
3693697075));
```

To display all execution plans of the SQL ID 'atfwcg8anrykp' stored in the SQL tuning set:

```
SELECT * FROM table
(DBMS_XPLAN.DISPLAY_SQLSET('OLTP_optimization_0405','gwp663cqh5qbf'));
```

To display runtime statistics for the SQL statement included in the preceding statement:

```
SELECT * FROM table (
   DBMS_XPLAN.DISPLAY_SQLSET(
      'OLTP_optimization_0405', 'gwp663cqh5qbf', NULL, 'ALLSTATS LAST');
```

DISPLAY_WORKLOAD_REPOSITORY Function

This table function displays the contents of an execution plan stored in AWR.

> **Note:**
>
> This function replaces `DISPLAY_AWR`, which is deprecated.

Syntax

```
DBMS_XPLAN.DISPLAY_WORKLOAD_REPOSITORY(
   sql_id            IN      VARCHAR2,
   plan_hash_value   IN      NUMBER   DEFAULT NULL,
   format            IN      VARCHAR2 DEFAULT 'TYPICAL'
   dbid              IN      NUMBER   DEFAULT NULL,
   con_dbid          IN      NUMBER   DEFAULT NULL,
   awr_location      IN      VARCHAR2 DEFAULT 'AWR_ROOT'
);
```

Parameters

Table 242-11 DISPLAY_WORKLOAD_REPOSITORY Table Function Parameters

Parameter	Description
`sql_id`	Specifies the `SQL_ID` of the SQL statement. You can retrieve the appropriate value for the SQL statement of interest by querying the column `SQL_ID` in `DBA_HIST_SQLTEXT`.
`plan_hash_value`	Specifies the `PLAN_HASH_VALUE` of a SQL statement. This parameter is optional. If omitted, the table function returns all stored execution plans for a given `SQL_ID`.

Table 242-11 (Cont.) DISPLAY_WORKLOAD_REPOSITORY Table Function Parameters

Parameter	Description
format	Controls the level of details for the plan. It accepts four values: • BASIC: Displays the minimum information in the plan—the operation ID, the operation name and its option. • TYPICAL: This is the default. Displays the most relevant information in the plan (operation id, name and option, #rows, #bytes and optimizer cost). Pruning, parallel and predicate information are only displayed when applicable. Excludes only PROJECTION, ALIAS and REMOTE SQL information (see below). • SERIAL: Like TYPICAL except that the parallel information is not displayed, even if the plan executes in parallel. • ALL: Maximum user level. Includes information displayed with the TYPICAL level with additional information (PROJECTION, ALIAS and information about REMOTE SQL if the operation is distributed). For finer control on the display output, the following keywords can be added to the above four standard format options to customize their default behavior. Each keyword either represents a logical group of plan table columns (such as PARTITION) or logical additions to the base plan table output (such as PREDICATE). Format keywords must be separated by either a comma or a space: • ROWS - if relevant, shows the number of rows estimated by the optimizer • BYTES - if relevant, shows the number of bytes estimated by the optimizer • COST - if relevant, shows optimizer cost information • PARTITION - if relevant, shows partition pruning information • PARALLEL - if relevant, shows PX information (distribution method and table queue information) • PREDICATE - if relevant, shows the predicate section • PROJECTION -if relevant, shows the projection section • ALIAS - if relevant, shows the "Query Block Name / Object Alias" section • REMOTE - if relevant, shows the information for distributed query (for example, remote from serial distribution and remote SQL) • NOTE - if relevant, shows the note section of the explain plan Format keywords can be prefixed by the sign '-' to exclude the specified information. For example, '-PROJECTION' excludes projection information.
dbid	Identifies the plans for a specific database. If this parameter is omitted, then the value defaults to the DBID of the AWR repository pointed to by the initialization parameter AWR_LOCATION. In a CDB, if AWR_LOCATION is set to AWR_ROOT, then the value is set to the DBID of the CDB root. If it is set to AWR_PDB, then the value is set to the DBID of the container.
con_dbid	Identifies the plans for a specific container. If this parameter is omitted, then the value defaults to SYS_CONTEXT('userenv', 'con_id').
awr_location	Specifies the location of the AWR repository. Supported values are: • AWR_ROOT when the AWR to be accessed is in the root container. This is the default. • 'AWR_PDB', if the AWR to be accessed is in the local container.

Example 242-2 Querying an AWR Plan

Assume that you log in as an administrator and issue the following query:

```
select count(*) from sh.sAleS
```

You create an AWR snapshot as follows:

```
EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;
```

You query joint `DBA_HIST_SQLTEST` to the function output as follows:

```
SET LINESIZE 150
SET PAGESIZE 5000

SELECT t.*
FROM   DBA_HIST_SQLTEXT ht,
       TABLE(DBMS_XPLAN.DISPLAY_WORKLOAD_REPOSITORY
         (ht.sql_id, null, '-PREDICATE +ALIAS',null,null,'AWR_ROOT')) t
WHERE  ht.SQL_TEXT LIKE '%sAleS%';

SQL_ID 2f4cx9qjnqd70
--------------------
select count(*) from sh.sAleS

Plan hash value: 1123225294

-------------------------------------------------------------------------------
----------------------------
| Id  | Operation                     | Name            | Rows  | Cost
(%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------
----------------------------
|   0 | SELECT STATEMENT              |                 |       |    27
(100)|          |       |       |
|   1 |  SORT AGGREGATE               |                 |     1 |
|          |       |       |
|   2 |   PARTITION RANGE ALL         |                 |   918K|    27
(0)| 00:00:01 |     1 |    28 |
|   3 |    BITMAP CONVERSION COUNT    |                 |   918K|    27
(0)| 00:00:01 |       |       |
|   4 |     BITMAP INDEX FAST FULL SCAN| SALES_PROMO_BIX |       |
|          |     1 |    28 |
-------------------------------------------------------------------------------
----------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1
   3 - SEL$1 / "SALES"@"SEL$1"
```