150 DBMS_PLSQL_CODE_COVERAGE

The `DBMS_PLSQL_CODE_COVERAGE` package provides an interface for the collection of code coverage data of PL/SQL applications at the basic block level.

This chapter contains the following topics:

DBMS_PLSQL_CODE_COVERAGE Overview

The `DBMS_PLSQL_CODE_COVERAGE` package provides an interface for collecting code coverage information at the basic block level of PL/SQL applications. A basic block refers to a single entry single exit block of PL/SQL code. PL/SQL developers want to know how well their test infrastructure exercised their code. The coverage tables are created using the `CREATE_COVERAGE_TABLES` procedure.

A typical code coverage run in a session involves calls to :

- `START_COVERAGE`
- Run PL/SQL code
- `STOP_COVERAGE`

The `GET_BLOCK_MAP` function helps you calculate your total coverage.

See Also:

- *Oracle Database Development Guide* for more information about using PL/SQL basic block coverage to maintain quality
- *Oracle Database PL/SQL Language Reference* for the `COVERAGE PRAGMA` syntax and semantics
- *Oracle Database PL/SQL Language Reference* for more information about the PLSQL_OPTIMIZE_LEVEL compilation parameter

DBMS_PLSQL_CODE_COVERAGE Security Model

The user must have `EXECUTE` privilege on the DBMS_PLSQL_CODE_COVERAGE package.

The user must have `CREATE` privilege on the unit to collect coverage information about this unit.

PL/SQL basic block coverage data is collected when program units use INTERPRETED compilation (parameter set PLSQL_CODE_TYPE = INTERPRETED). PL/SQL basic block coverage data is not collected when program units use NATIVE compilation. You can disable the NATIVE compiler by setting the parameter PLSQL_OPTIMIZE_LEVEL <= 1. Regardless of the compilation mode, coverage data for wrapped units is not collected.

DBMS_PLSQL_CODE_COVERAGE Constants

The DBMS_PLSQL_CODE_COVERAGE package provides constants that are used with the namespace parameter of the GET_BLOCK_MAP function.

These constants are described in the following table.

Table 150-1 DBMS_PLSQL_CODE_COVERAGE Constants

Name	Type	Value	Description
function_namespace	NUMBER	1	Specifies the function namespace
package_spec_namespace	NUMBER	1	Specifies the package specification namespace
package_body_namespace	NUMBER	2	Specifies the package definition (body) namespace
procedure_namespace	NUMBER	1	Specifies the procedure namespace
trigger_namespace	NUMBER	3	Specifies the trigger namespace
type_spec_namespace	NUMBER	1	Specifies the type specification namespace
type_body_namespace	NUMBER	2	Specifies the type definition (body) namespace

DBMS_PLSQL_CODE_COVERAGE Data Structures

Record Types

- MAP_REC Record Type

Table Types

- T_MAP_REC Table Type

MAP_REC Record Type

The MAP_REC record type defines the PL/SQL basic block location in the source code.

Syntax

```
TYPE map_rec IS RECORD (
    procedure_name VARCHAR2(32767),
    block_num      NUMBER,
    line           NUMBER,
    col            NUMBER,
    not_feasible   NUMBER);
```

Fields

Table 150-2   MAP_REC Fields

Field	Description
`procedure_name`	The name of the procedure containing the basic block
`block_num`	Identifies the basic block
`line`	Starting line of the basic block
`col`	Starting column of the basic block
`not_feasible`	Not_feasible marking of the basic block

T_MAP_REC Table Type

The T_MAP_REC table type specifies the collection of PL/SQL basic blocks in a unit.

Syntax

```
TYPE t_map_rec IS TABLE OF map_rec;
```

Summary of DBMS_PLSQL_CODE_COVERAGE Subprograms

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

Table 150-3   DBMS_PLSQL_CODE_COVERAGE Package Subprograms

Subprogram	Description
CREATE_COVERAGE_TABLES Procedure	Creates coverage tables
GET_BLOCK_MAP Function	Gets the mapping of basic blocks to PL/SQL source
START_COVERAGE Function	Starts the coverage data collection in the user's session and returns the RUN_ID
STOP_COVERAGE Procedure	Ends the current coverage run

CREATE_COVERAGE_TABLES Procedure

This procedure creates the tables used for coverage data collection.

Syntax

```
DBMS_PLSQL_CODE_COVERAGE.CREATE_COVERAGE_TABLES (
   FORCE_IT      IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
`FORCE_IT`	The default is to raise an error if the coverage tables already exists. If set to TRUE, the tables are dropped silently if the tables already exist, and new tables are created.

Exceptions

Table 150-4 CREATE_COVERAGE_TABLES Exceptions

Exception	Description
`COVERAGE_ERROR`	The `FORCE_IT` parameter is FALSE and the tables already exist.

GET_BLOCK_MAP Function

This function gets the mapping of basic blocks to PL/SQL source.

Syntax

```
DBMS_PLSQL_CODE_COVERAGE.GET_BLOCK_MAP(
                       unit_owner IN VARCHAR2,
                       unit_name  IN VARCHAR2,
                       namespace  IN POSITIVE)
  RETURN T_MAP_REC;
```

Table 150-5 Parameters

Parameter	Description
`unit_owner`	The owner of the unit. The unit owner is case insensitive. If the unit_owner is empty or NULL, then it defaults to the current schema.
`unit_name`	The unit whose mapping is to be gotten. The unit_name is case insensitive.
`namespace`	Namespace to which this unit_name gets resolved. See DBMS_PLSQL_CODE_COVERAGE Constants for a list of valid namespace values.

START_COVERAGE Function

This function starts the coverage data collection in the user's session and returns a unique identifier `RUN_ID` for the run.

Syntax

```
DBMS_PLSQL_CODE_COVERAGE.START_COVERAGE (
   run_comment   IN VARCHAR2)
   RETURN NUMBER;
```

Parameters

Parameter	Description
`run_comment`	Allows the user to name a run and identify the test.

STOP_COVERAGE Procedure

This procedure ends the current coverage run.

Syntax

```
DBMS_PLSQL_CODE_COVERAGE.STOP_COVERAGE;
```

Exceptions

Table 150-6 STOP_COVERAGE Exceptions

Exception	Description
`COVERAGE_ERROR`	An error is raised if the coverage tables do not exist.