99
DBMS_HPROF

The `DBMS_HPROF` package provides an interface for profiling the execution of PL/SQL applications. It provides services for collecting the hierarchical profiler data, analyzing the raw profiler output and profiling information generation.

This chapter contains the following topic:

- Summary of DBMS_HPROF Subprograms

> **See Also:**
>
> *Oracle Database Development Guide* for more information about the "PL/SQL Hierarchical Profiler"

DBMS_HPROF Security Model

You must have the following privileges to use the `DBMS_HPROF` package:

- An `EXECUTE` privilege on the `DBMS_HPROF` package.
- A `WRITE` privilege on the directory that you specify when you call the `DBMS_HPROF.START_PROFILING` procedure.

Summary of DBMS_HPROF Subprograms

This table lists and briefly describes the `DBMS_HPROF` package subprograms.

Table 99-1 DBMS_HPROF Package Subprograms

Subprogram	Description
ANALYZE Function	Analyzes the raw profiler output and produces hierarchical profiler information in database tables.
CREATE_TABLES Procedure	Creates the hierarchical profiler database tables and data structures in the user's session.
START_PROFILING Procedure	Starts hierarchical profiler data collection in the user's session.
STOP_PROFILING Procedure	Stops profiler data collection in the user's session.

ANALYZE Function

This function analyzes the raw profiler output and produces hierarchical profiler information in database tables or generates out-of-the-box HTML reports.

Syntax

```
DBMS_HPROF.ANALYZE (
   trace_id          IN NUMBER,
   summary_mode      IN BOOLEAN DEFAULT FALSE,
   trace             IN VARCHAR2 DEFAULT NULL,
   skip              IN PLS_INTEGER DEFAULT 0,
   collect           IN PLS_INTEGER DEFAULT NULL,
   run_comment       IN VARCHAR2 DEFAULT NULL)
  RETURN NUMBER;

DBMS_HPROF.ANALYZE (
   trace_id          IN NUMBER,
   report_clob       OUT CLOB,
   trace             IN VARCHAR2 DEFAULT NULL,
   skip              IN PLS_INTEGER DEFAULT 0,
   collect           IN PLS_INTEGER DEFAULT NULL);
```

Parameters

Table 99-2 ANALYZE Function Parameters

Parameter	Description
`trace_id`	The `trace_id` of the raw profiler data entry in the raw profiler data table (`dbmshp_trace_data`).
`summary_mode`	By default (that is, when `summary_mode` is `FALSE`), a detailed analysis is done. When `summary_mode` is `TRUE`, only top-level summary information is generated into the database table.
`report_clob`	The analyzed HTML report.
`trace`	Analyzes only the subtrees rooted at the specified trace entry. By default (when trace is `NULL`), the analysis/reporting is generated for the entire run. The trace entry must be specified in a special quoted qualified format. For example, `'"HR"."PKG"."FOO"'` or `'"."."__plsql_vm"'`. If multiple overloads exist for the specified name, all of them will be analyzed.
`skip`	Used only when `trace` is specified. Analyze only the subtrees rooted at the specified trace, but ignore the first `skip` invocations to trace. The default value for skip is `0`.
`collect`	Used only when `trace` is specified. Analyze `collect` number of invocations of traces (starting from `skip`+1'th invocation). By default, only 1 invocation is collected.
`run_comment`	User-provided comment for this run.

Return Values

A unique run identifier for this run of the analyzer. This can then be used to look up the results corresponding to this run from the hierarchical profiler tables.

Usage Notes

- Use the `DBMS_HPROF.CREATE_TABLES` subprogram to create the hierarchical profiler database tables and other data structures required for persistently storing the results of analyzing the raw profiler data.
- Calling the `DBMS_HPROF.CREATE_TABLES` with default value (`FALSE`) will raise error if table already exists.
- Use `DBMS_HPROF.CREATE_TABLES(TRUE)` to drop any previously created hierarchical profiler tables.
- Use the `DBMS_HPROF.CREATE_TABLES` to drop any previously created hierarchical profiler tables. By default, `force_it` is `FALSE`; therefore, to drop any previously created hierarchical profiler tables you must set the value of `force_it` to `TRUE`.
- If `trace_id` entry is `NULL`, error is raised.
- If `trace_id` entry in the raw profiler data table does not exist, error is raised.
- If raw data of the `trace_id` entry in the raw profiler data table is `NULL` or is `zero` size, error is raised.

Examples

The following snippet installs the hierarchical profiler tables in HR schema.

```
connect HR/<password>;
```

The following example analyzes and generates HTML CLOB report from a raw profiler data table.

```
DECLARE
        reportclob clob;
        trace_id number;
BEGIN
    -- create raw profiler data and analysis tables
    -- force_it =>TRUE will dropped the tables if table exists
    DBMS_HPROF.CREATE_TABLES(force_it =>TRUE);

    -- Start profiling
    -- Write raw profiler data in raw profiler data table
    trace_id := DBMS_HPROF.START_PROFILING;

    -- Run procedure to be profiled
    test;

    -- Stop profiling
    DBMS_HPROF.STOP_PROFILING;

    -- analyzes trace_id entry in raw profiler data table and produce
    -- analyzed HTML report in reportclob
    DBMS_HPROF.ANALYZE(trace_id , reportclob);
END;
/
```

CREATE_TABLES Procedure

Creates the hierarchical profiler database tables and data structures in the user's session.

Syntax

```
DBMS_HPROF.CREATE_TABLES (
   force_it         IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 99-3 CREATE_TABLES Procedure Parameters

Parameter	Description
`force_it`	If `FALSE` and `DBMS_HPROF` tables are present, then an `HPROF` error is raised. If `TRUE`, then the procedure creates tables. If the tables already exist, then they are dropped and new tables are created.

> **Note:**
>
> Users need not use the `dbmshptab.sql` script located in the `rdbms/admin` directory to create the hierarchical profiler database tables and data structures anymore.
>
> The `dbmshptab.sql` script is deprecated starting in Oracle Database 18*c*.

START_PROFILING Procedure

This procedure starts hierarchical profiler data collection in the user's session.

Syntax

```
DBMS_HPROF.START_PROFILING(
   max_depth     IN PLS_INTEGER DEFAULT NULL,
   sqlmonitor    IN BOOLEAN DEFAULT TRUE,
   run_comment   IN VARCHAR2 DEFAULT NULL)
RETURN NUMBER;
```

Parameters

Table 99-4 START_PROFILING Procedure Parameters

Parameter	Description
`max_depth`	By default (that is, when `max_depth` value is `NULL`) profile information is gathered for all functions irrespective of their call depth. When a non-`NULL` value is specified for `max_depth`, the profiler collects data only for functions up to a call depth level of `max_depth`.

Table 99-4 (Cont.) START_PROFILING Procedure Parameters

Parameter	Description
sqlmonitor	Generates a real-time monitoring report for a profiler run when the profiler run ends. The default value is TRUE.
run_comment	User provided comment for the profiler data collection run.

Return Values

Unique run identifier for this profiler run. This can then be used to look up the results corresponding to this run from the hierarchical profiler tables.

Usage Notes

- Even though the profiler does not individually track functions at depth greater than max_depth, the time spent in such functions is charged to the ancestor function at depth max_depth.
- Raw profiler data is generated in the raw profiler data table with an unique trace_id.
- The unique trace_id is used to manage the raw profiler output stored in the raw profiler data table.

STOP_PROFILING Procedure

This procedure stops profiler data collection in the user's session. This subprogram also has the side effect of flushing data collected so far in the session, and it signals the end of a run. When the STOP_PROFILING procedure returns CLOB, it contains the Real-Time Monitoring report for the profiler run.

Syntax

```
DBMS_HPROF.STOP_PROFILING;


DBMS_HPROF.STOP_PROFILING
  RETURN CLOB;
```

Examples

Profiling with raw profiler data table

```
DECLARE
    analyze_runid number;
    trace_id number;
BEGIN
    -- create raw profiler data and analysis tables
    -- call create_tables with force_it =>FALSE (default) when
    -- raw profiler data and analysis tables do not exist already
    DBMS_HPROF.CREATE_TABLES;
    -- Start profiling
    -- Write raw profiler data in raw profiler data table
    trace_id := DBMS_HPROF.START_PROFILING;
    -- Run the procedure to be profiled
    test;
    -- Stop profiling
```

```
    DBMS_HPROF.STOP_PROFILING;
    -- analyzes trace_id entry in raw profiler data table and writes
    -- hierarchical profiler information in hprof's analysis tables
    analyze_runid := DBMS_HPROF.ANALYZE(trace_id);
END;
/
```