15
DBMS_ACTIVITY

`DBMS_ACTIVITY` contains functions and procedures allowing authorized users to control the activity information captured by OATS.

This package is owned by `SYS`, so `EXECUTE` package privilege is required of the non-`SYS` users. Users with DBA role are granted the `EXECUTE` privilege on this package.

This chapter contains the following topics:

Using DBMS_ACTIVITY

Object Activity Tracking System (OATS) is a generic RDBMS based tracking service that provides information about various types of activities associated with different database objects. An activity represents a user or system-initiated action such as scanning or loading a table. Most of the activities are tracked in the form of frequencies (i.e. counts over fixed time intervals) such as the number of scans of a table in 15 minutes.

The database objects whose activities are tracked include tables and materialized views. Different types of activities include row insert, delete and update, table scan, load and truncate, partition maintenance operations (create, drop, move, split, merge, and exchange), materialized view rewrite and refresh.

Another important class of activities is related to the usage or non-usage of auxiliary structures such as MVs, indexes and zone maps.

Depending on the type of activity the tracking technique can be precise, approximate, or probabilistic. Most of the activities are tracked approximately mostly for efficiency reasons.

Depending on the type of activity and its usage by different clients a certain tracking technique would be more suitable than others. For example, counters are useful for tracking the index and materialized view usage. Counters are maintained for fixed time intervals in order to capture the object usage pattern over time. The same technique can be used to track the update activity of table columns by maintaining update counters for each. Setting bits in a bitvector is another form of tracking technique which is suitable for registering the occurrence of a certain activity within a time interval without saying how many times that activity occurred.

Precise Tracking: an activity is accounted for with 100% accuracy. Some form of atomic update or some latching is required to get precise tracking in a multi-processor environment. OATS generally avoidS precise tracking, unless a client needs that kind of precision.

Approximate Tracking: an activity is accounted for in almost all cases but with rare exceptions. An example of approximate tracking is the dirty update of activity counters. Dirty updating is very efficient because no locking or latching is performed and atomic operations are not used when counters are changed. However, dirty updating can result in lost updates leading to under counting. If lost updates are rare then dirty updating is a very efficient technique for approximate tracking.

Probabilistic Tracking: an activity is sampled with a certain probability, and it is accounted for when it becomes part of the sample. Probabilistic tracking produces activity data that is less accurate but it prevents frequent tracking actions.

Occurrence Tracking: records certain activity as having occurred within a time interval. Specifically, occurrence tracking provides information that says that certain object activity occurred at least once in a given time interval without saying how many times it occurred. The use of bitvector is a popular technique for occurrence tracking. Depending on how the bits are set in a bitvector the occurrence tracking can be either precise (no collisions) or approximate (possible collisions).

Counter Effects of Transactions and Rollback: Except for the precise tracking counters, none of the other types of counters is transactional or affected by rollback. This means that if a transaction fails, the approximate counters may still record any actions from that transaction. And when the database is rolled back to a save point, the approximate counters are not set back to their previous state.

Summary of DBMS_ACTIVITY Subprograms

`DBMS_ACTIVITY` uses the `CONFIGURE`, `CREATE_SNAPSHOP`, and `DELETE_SNAPSHOTS` procedures.

Table 15-1 DBMS_ACTIVITY Package Subprograms

Subprogram	Description
CONFIGURE Procedure	Allows an authorized user to set configuration parameters for OATS in parameter/value format.
CREATE_SNAPSHOT Procedure	Allows an authorized user to manually create an activity snapshot on local instance or on all database instances by flushing the activity information maintained in memory to disk
DELETE_SNAPSHOTS Procedure	Allows an authorized user to manually delete all older snapshots based on an input snapshot id called `BEFORE_SNAP_ID`.
DELETE_SNAPSHOTS	Allows an authorized user to manually delete all older snapshots based on an input timestamp value called `BEFORE_TIME`.

CONFIGURE Procedure

This procedure allows an authorized user to set configuration parameters for OATS in parameter/value format.

This function can be called numerous times, each time setting a different configuration parameter, or same parameter but with different value. The parameter setting applies to the indicated database or the local database.

Syntax

```
DBMS_ACTIVITY.CONFIGURE (
   PARAMETER_NAME       IN VARCHAR2
   PARAMETER_VALUE      IN NUMBER,
   CON_DBNAME           IN VARCHAR2 := NULL
);
```

Parameters

Table 15-2 CONFIGURE Procedure Parameters

Parameter	Description
PARAMETER_NAME	Name of the configuration parameter to set. Parameters available: • ACTIVITY_INTERVAL_MINUTES: The interval in minutes for maintaining activity information before it is flushed and reset. The default is 15. Other values: 30, 60, 120, 180, 240, 360, 480, 720, 1440 • ACTIVITY_RETENTION_DAYS: the number of days to maintain the activity information before it is purged. The default is 400. From 8 to 2000. • ACTIVITY_SPACE_PERCENT: soft limit in percent of the SYSAUX space for storing the activity information. The default is 5. From 1 to 25.
PARAMETER_VALUE	Value of the configuration parameter to use.
CON_DBNAME	Name of a container in the consolidated database (CDB). It is either root or a pluggable database. When omitted, the default is the local database.

> **Note:**
>
> The current CDB or PDB name is the only non-null value supported.

Usage Notes

The user must be SYS, or must have the DBA role, or granted the EXECUTE package privilege.

CREATE_SNAPSHOT Procedure

This procedure allows an authorized user to manually create an activity snapshot on local instance or on all database instances by flushing the activity information maintained in memory to disk.

Syntax

```
DBMS_ACTIVITY.CREATE_SNAPSHOT(
   ALL_INSTANCES       IN BOOLEAN := TRUE,
   CON_DBNAME          IN VARCHAR2 := NULL
);
```

Parameters

Table 15-3 CREATE_SNAPSHOT Procedure Parameters

Parameter	Description
ALL_INSTANCES	Specify FALSE if activity snapshot should be created only for the local instance. The default is to create snapshot on all database instances.

Table 15-3 (Cont.) CREATE_SNAPSHOT Procedure Parameters

Parameter	Description
CON_DBNAME	Name of a container in the consolidated database (CDB). It is either root or a pluggable database. When omitted, the default is the local database.

> **Note:**
>
> The current CDB or PDB name is the only non-null value supported.

Usage Notes

The SNAP_ID of snapshot created.

The user must be SYS, or must have the DBA role, or granted the EXECUTE package privilege.

DELETE_SNAPSHOTS Procedure

This procedure allows an authorized user to manually delete all older snapshots based on an input snapshot id called BEFORE_SNAP_ID.

Syntax

```
DBMS_ACTIVITY.DELETE_SNAPSHOTS (
   BEFORE_SNAP_ID      IN NUMBER,
   CON_DBNAME          IN VARCHAR2 := NULL
);
```

Parameters

Table 15-4 DELETE_SNAPSHOTS Procedure Parameters

Parameter	Description
BEFORE_SNAP_ID	All snapshots with SNAP_ID value less than this argument value is removed from the disk storage (SYSAUX).
CON_DBNAME	Name of a container in the consolidated database (CDB). It is either root or a pluggable database. When omitted, the default is the local database.

> **Note:**
>
> The current CDB or PDB name is the only non-null value supported.

Usage Notes

Returns TRUE if one or more snapshots were deleted; FALSE otherwise.

The user must be SYS, or must have the DBA role, or granted the EXECUTE package privilege.

DELETE_SNAPSHOTS Procedure

This procedure allows an authorized user to manually delete all older snapshots based on an input timestamp value called BEFORE_TIME.

Syntax

```
DBMS_ACTIVITY.DELETE_SNAPSHOTS (
   BEFORE_TIME          IN TIMESTAMP,
   CON_DBNAME           IN VARCHAR2 := NULL
);
```

Parameters

Table 15-5 DELETE_SNAPSHOTS Procedure Parameters

Parameter	Description
BEFORE_TIME	All snapshots associated with a time less than this argument value are removed from the disk storage (SYSAUX).
CON_DBNAME	Name of a container in the consolidated database (CDB). It is either root or a pluggable database. When omitted, the default is the local database.

> **Note:**
>
> The current CDB or PDB name is the only non-null value supported.

Usage Notes

Returns TRUE if one or more snapshots were deleted; FALSE otherwise.

The user must be SYS, or must have the DBA role, or granted the EXECUTE package privilege.