205
DBMS_TRANSACTION

The DBMS_TRANSACTION package provides access to SQL transaction statements from stored procedures.

See Also:

Oracle Database SQL Language Reference

This chapter contains the following topics:

- DBMS_TRANSACTION Security Model
- Summary of DBMS_TRANSACTION Subprograms

DBMS_TRANSACTION Security Model

This package runs with the privileges of calling user, rather than the package owner `SYS`.

Summary of DBMS_TRANSACTION Subprograms

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

Table 205-1 DBMS_TRANSACTION Package Subprograms

Subprogram	Description
ADVISE_COMMIT Procedure	Equivalent to the SQL statement: `ALTER SESSION ADVISE COMMIT`
ADVISE_NOTHING Procedure	Equivalent to the SQL statement: `ALTER SESSION ADVISE NOTHING`
ADVISE_ROLLBACK Procedure	Equivalent to the SQL statement: `ALTER SESSION ADVISE ROLLBACK`
COMMIT Procedure	Equivalent to the SQL statement: `COMMIT`
COMMIT_COMMENT Procedure	Equivalent to the SQL statement: `COMMIT COMMENT <text>`
COMMIT_FORCE Procedure	Equivalent to the SQL statement: `COMMIT FORCE <text>, <number>"`
LOCAL_TRANSACTION_ID Function	Returns the local (to instance) unique identifier for the current transaction
GET_TRANSACTION_ID Function	Returns the transaction identifier for the Sessionless transaction or the XA transaction in the session

Table 205-1 (Cont.) DBMS_TRANSACTION Package Subprograms

Subprogram	Description
GET_TRANSACTION_TIMEOUT Function	Returns the timeout, in seconds, of the XA transaction branch or the Sessionless transaction within a session
GET_TRANSACTION_TYPE Function	Returns the type of the transaction active in the session
PURGE_LOST_DB_ENTRY Procedure	Enables removal of incomplete transactions from the local site when the remote database is destroyed or re-created before recovery completes
PURGE_MIXED Procedure	Deletes information about a given mixed outcome transaction
READ_ONLY Procedure	Equivalent to the SQL statement: `SET TRANSACTION READ ONLY`
READ_WRITE Procedure	equivalent to the SQL statement: `SET TRANSACTION READ WRITE`
ROLLBACK Procedure	Equivalent to the SQL statement: `ROLLBACK`
ROLLBACK_FORCE Procedure	Equivalent to the SQL statement: `ROLLBACK FORCE <text>`
ROLLBACK_SAVEPOINT Procedure	Equivalent to the SQL statement: `ROLLBACK TO SAVEPOINT <savepoint_name>`
SAVEPOINT Procedure	Equivalent to the SQL statement: `SAVEPOINT <savepoint_name>`
START_TRANSACTION Function	Starts a local (vanilla) transaction or a Sessionless transaction
STEP_ID Function	Returns local (to local transaction) unique positive integer that orders the DML operations of a transaction
SUSPEND_TRANSACTION Procedure	Suspends the Sessionless transaction that is active within a session
USE_ROLLBACK_SEGMENT Procedure	Equivalent to the SQL statement: `SET TRANSACTION USE ROLLBACK SEGMENT <rb_seg_name>`

ADVISE_COMMIT Procedure

This procedure is equivalent to the SQL statement: `ALTER SESSION ADVISE COMMIT`

Syntax

```
DBMS_TRANSACTION.ADVISE_COMMIT;
```

ADVISE_NOTHING Procedure

This procedure is equivalent to the SQL statement: `ALTER SESSION ADVISE NOTHING`

Syntax

```
DBMS_TRANSACTION.ADVISE_NOTHING;
```

ADVISE_ROLLBACK Procedure

This procedure is equivalent to the SQL statement: `ALTER SESSION ADVISE ROLLBACK`

Syntax

```
DBMS_TRANSACTION.ADVISE_ROLLBACK;
```

COMMIT Procedure

This procedure is equivalent to the SQL statement: `COMMIT`

This procedure is included for completeness, the functionality being already implemented as part of PL/SQL.

Syntax

```
DBMS_TRANSACTION.COMMIT;
```

COMMIT_COMMENT Procedure

This procedure is equivalent to the SQL statement: `COMMIT COMMENT <text>`

Syntax

```
DBMS_TRANSACTION.COMMIT_COMMENT (
   cmnt VARCHAR2);
```

Parameters

Table 205-2 COMMIT_COMMENT Procedure Parameters

Parameter	Description
`cmnt`	Comment to associate with this commit.

COMMIT_FORCE Procedure

This procedure is equivalent to the SQL statement: `COMMIT FORCE <text>, <number>"`

Syntax

```
DBMS_TRANSACTION.COMMIT_FORCE (
   xid VARCHAR2,
   scn VARCHAR2 DEFAULT NULL);
```

Parameters

Table 205-3 COMMIT_FORCE Procedure Parameters

Parameter	Description
`xid`	Local or global transaction ID.
`scn`	System change number.

GET_TRANSACTION_ID Function

This function returns the transaction identifier in the session.

This function returns the hexadecimal representations of the transaction identifier (in `varchar2` format) of the active XA transaction or Sessionless transaction in the session. If there are none of these types of transactions active within the session, it returns `NULL`.

If it's an XA transaction, the function returns a hex representation of format ID and GTRID separated by a dot (`<format id>.<GTRID>`).

If it is a Sessionless transaction, the function returns a hex representation of GTRID. Call `GET_TRANSACTION_TYPE` to get whether the transaction is of type XA or Sessionless.

Syntax

```
GET_TRANSACTION_ID
   RETURN VARCHAR2;
```

GET_TRANSACTION_TIMEOUT Function

This function returns the timeout, in seconds, of the XA transaction branch or the Sessionless transaction within a session.

If a local transaction is active within the session, the function returns `UB4MAXVAL`.

If no transaction is active within the session, the function returns `NULL`.

Syntax

```
GET_TRANSACTION_TIMEOUT
   RETURN NUMBER;
```

GET_TRANSACTION_TYPE Function

This function returns the type of the transaction active in the session.

The type of the transaction returned can be one of the following: `TRANSACTION_TYPE_LOCAL`, `TRANSACTION_TYPE_SESSIONLESS`, `TRANSACTION_TYPE_XA`, or `NULL` (if no transaction is active within the session).

Syntax

```
GET_TRANSACTION_TYPE
   RETURN PLS_INTEGER;
```

LOCAL_TRANSACTION_ID Function

This function returns the local (to instance) unique identifier for the current transaction. It returns null if there is no current transaction.

Syntax

```
DBMS_TRANSACTION.LOCAL_TRANSACTION_ID (
   create_transaction BOOLEAN := FALSE)
  RETURN VARCHAR2;
```

Parameters

Table 205-4 LOCAL_TRANSACTION_ID Function Parameters

Parameter	Description
`create_transaction`	If true, then start a transaction if one is not currently active.

PURGE_LOST_DB_ENTRY Procedure

Procedure `PURGE_LOST_DB_ENTRY` purges entries that control database recovery from a local site.

When a failure occurs during commit processing, automatic recovery consistently resolves the results at all sites involved in the transaction. However, if the remote database is destroyed or re-created before recovery completes, then the entries used to control recovery in `DBA_2PC_PENDING` and associated tables are never removed, and recovery will periodically retry. Procedure `PURGE_LOST_DB_ENTRY` enables removal of such transactions from the local site.

Syntax

```
DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY (
   xid VARCHAR2);
```

Parameters

Table 205-5 PURGE_LOST_DB_ENTRY Procedure Parameters

Parameter	Description
`xid`	Must be set to the value of the `LOCAL_TRAN_ID` column in the `DBA_2PC_PENDING` table.

Usage Notes

WARNING:

PURGE_LOST_DB_ENTRY should *only* be used when the other database is lost or has been re-created. Any other use may leave the other database in an unrecoverable or inconsistent state.

Before automatic recovery runs, the transaction may show up in `DBA_2PC_PENDING` as state "collecting", "committed", or "prepared". If the DBA has forced an in-doubt transaction to have a particular result by using "commit force" or "rollback force", then states "forced commit" or "forced rollback" may also appear. Automatic recovery normally deletes entries in any of these states. The only exception is when recovery finds a forced transaction which is in a state inconsistent with other sites in the transaction; in this case, the entry is left in the table and the `MIXED` column has the value 'yes'.

However, under certain conditions, it may not be possible for automatic recovery to run. For example, a remote database may have been permanently lost. Even if it is re-created, it gets a new database ID, so that recovery cannot identify it (a possible symptom is `ORA-02062`). In this case, the DBA may use the procedure `PURGE_LOST_DB_ENTRY` to clean up the entries in any state other than "prepared". The DBA does not need to be in any particular hurry to resolve these entries, because they are not holding any database resources.

The following table indicates what the various states indicate about the transaction and what the DBA actions should be:

Table 205-6 PURGE_LOST_DB_ENTRY Procedure States

State of Column	State of Global Transaction	State of Local Transaction	Normal DBA Action	Alternative DBA Action
`Collecting`	`Rolled back`	`Rolled back`	`None`	`PURGE_LOST_DB_ENTRY (See Note 1)`
`Committed`	`Committed`	`Committed`	`None`	`PURGE_LOST_DB_ENTRY (See Note 1)`
`Prepared`	`Unknown`	`Prepared`	`None`	`FORCE COMMIT or ROLLBACK`
`Forced commit`	`Unknown`	`Committed`	`None`	`PURGE_LOST_DB_ENTRY (See Note 1)`
`Forced rollback`	`Unknown`	`Rolled back`	`None`	`PURGE_LOST_DB_ENTRY (See Note 1)`
`Forced commit (mixed)`	`Mixed`	`Committed`	`(See Note 2)`	
`Forced rollback (mixed)`	`Mixed`	`Rolled back`	`(See Note 2)`	

Note:

Use only if significant reconfiguration has occurred so that automatic recovery cannot resolve the transaction. Examples are total loss of the remote database, reconfiguration in software resulting in loss of two-phase commit capability, or loss of information from an external transaction coordinator such as a TP monitor.

Note:

Examine and take any manual action to remove inconsistencies; then use the procedure `PURGE_MIXED`.

PURGE_MIXED Procedure

This procedure deletes information about a given mixed outcome transaction

When in-doubt transactions are forced to commit or rollback (instead of letting automatic recovery resolve their outcomes), there is a possibility that a transaction can have a mixed outcome; some sites commit, and others rollback. Such inconsistency cannot be resolved automatically by Oracle. However, Oracle flags entries in `DBA_2PC_PENDING` by setting the `MIXED` column to a value of 'yes'.

Oracle never automatically deletes information about a mixed outcome transaction. When the application or DBA is certain that all inconsistencies that might have arisen as a result of the mixed transaction have been resolved, this procedure can be used to delete the information about a given mixed outcome transaction.

Syntax

```
DBMS_TRANSACTION.PURGE_MIXED (
   xid VARCHAR2);
```

Parameters

Table 205-7 PURGE_MIXED Procedure Parameters

Parameter	Description
`xid`	Must be set to the value of the `LOCAL_TRAN_ID` column in the `DBA_2PC_PENDING` table.

READ_ONLY Procedure

This procedure is equivalent to the SQL statement `SET TRANSACTION READ ONLY`.

Syntax

```
DBMS_TRANSACTION.READ_ONLY;
```

READ_WRITE Procedure

This procedure is equivalent to the SQL statement:

```
SET TRANSACTION READ WRITE
```

Syntax

```
DBMS_TRANSACTION.READ_WRITE;
```

ROLLBACK Procedure

This procedure is equivalent to the SQL statement `ROLLBACK`.

This procedure is included for completeness, the functionality being already implemented as part of PL/SQL.

Syntax

```
DBMS_TRANSACTION.ROLLBACK;
```

ROLLBACK_FORCE Procedure

This procedure is equivalent to the SQL statement `ROLLBACK FORCE <text>`.

Syntax

```
DBMS_TRANSACTION.ROLLBACK_FORCE (
   xid VARCHAR2);
```

Parameters

Table 205-8 ROLLBACK_FORCE Procedure Parameters

Parameter	Description
`xid`	Local or global transaction ID.

ROLLBACK_SAVEPOINT Procedure

This procedure is equivalent to the SQL statement `ROLLBACK TO SAVEPOINT <savepoint_name>`.

This procedure is included for completeness, the functionality being already implemented as part of PL/SQL.

Syntax

```
DBMS_TRANSACTION.ROLLBACK_SAVEPOINT (
   savept VARCHAR2);
```

Parameters

Table 205-9 ROLLBACK_SAVEPOINT Procedure Parameters

Parameter	Description
`savept`	Savepoint identifier.

SAVEPOINT Procedure

This procedure is equivalent to the SQL statement `SAVEPOINT <savepoint_name>` .

This procedure is included for completeness, the feature being already implemented as part of PL/SQL.

Syntax

```
DBMS_TRANSACTION.SAVEPOINT (
   savept VARCHAR2);
```

Parameters

Table 205-10 SAVEPOINT Procedure Parameters

Parameter	Description
`savept`	Savepoint identifier.

START_TRANSACTION Function

This function starts a local (vanilla) transaction or a Sessionless transaction depending on the `transaction_type` being `TRANSACTION_TYPE_LOCAL` or `TRANSACTION_TYPE_SESSIONLESS`.

If starting a local transaction, this function returns the `XID <usn>.<slot>.<wrap>` just like `DBMS_TRANSACTION.local_transaction_id()` does. If starting a Sessionless transaction, this function returns the hexadecimal representation of GTRID of the started transaction.

Syntax

```
DBMS_TRANSACTION.START_TRANSACTION(
   xid              in raw default null,
   transaction_type in pls_integer default TRANSACTION_TYPE_LOCAL,
   timeout          in pls_integer default 60,
   flag             in pls_integer default 0
) RETURN VARCHAR2;
```

The following table describes the parameters of the `DBMS_TRANSACTION.START_TRANSACTION` function.

Note:

The `xid`, `timeout`, and `flag` parameters apply only if the `TRANSACTION_TYPE` is `TRANSACTION_TYPE_SESSIONLESS`.

Table 205-11 START_TRANSACTION Function Parameters

Parameter	Description
`xid`	GTRID of the Sessionless transaction to be newly started or resumed. It can be raw byte array of size up to 64 bytes. If `NULL`, the server generates a GTRID.
`TRANSACTION_TYPE`	To start a local transaction, add `TRANSACTION_TYPE` as `TRANSACTION_TYPE_LOCAL` and to start a Sessionless transaction, add `TRANSACTION_TYPE` as `TRANSACTION_TYPE_SESSIONLESS`.
`timeout`	When the `TRANSACTION_NEW` flag is used, `timeout` denotes the transaction time-out value, which must be a positive number. It specifies the duration in seconds in which this transaction can be resumed after it is suspended. When the `TRANSACTION_RESUME` flag is used, `timeout` denotes the resume timeout value, which can be either 0 or a positive number. Within an instance, a Sessionless transaction can only be associated with one session at any given time. Thus, a Sessionless transaction cannot be resumed if it is associated with another session on the same instance. When this situation arises, the resume timeout value specifies how long (in seconds) this session waits for the other session to suspend the transaction, so that it can resume the transaction. If the other session does not suspend the transaction before the wait times out, an ORA-25351 error is raised. If the resume timeout value is 0, the error is immediately raised.
`flag`	The flag can be either `TRANSACTION_NEW` for starting a new transaction or `TRANSACTION_RESUME` for resuming a suspended transaction.

STEP_ID Function

This function returns local (to local transaction) unique positive integer that orders the DML operations of a transaction.

Syntax

```
DBMS_TRANSACTION.STEP_ID
   RETURN NUMBER;
```

SUSPEND_TRANSACTION Procedure

This procedure suspends the Sessionless transaction that is active within a session.

If the active transaction is not a Sessionless transaction, an `ORA-26202` error is raised. If there is no transaction active within the session, the suspend transaction operation is a no-op.

Syntax

```
DBMS_TRANSACTION.SUSPEND_TRANSACTION
```

USE_ROLLBACK_SEGMENT Procedure

This procedure is equivalent to the SQL statement `SET TRANSACTION USE ROLLBACK SEGMENT <rb_seg_name>`.

Syntax

```
DBMS_TRANSACTION.USE_ROLLBACK_SEGMENT (
   rb_name VARCHAR2);
```

Parameters

Table 205-12 USE_ROLLBACK_SEGMENT Procedure Parameters

Parameter	Description
`rb_name`	Name of rollback segment to use.