88 DBMS_FLASHBACK

Using `DBMS_FLASHBACK`, you can flash back to a version of the database at a specified time or a specified system change number (SCN).

This chapter contains the following topics:

See Also:

For detailed information about `DBMS_FLASHBACK`:

- *Oracle Database Development Guide*
- *Oracle Database SQL Language Reference*.

DBMS_FLASHBACK Overview

`DBMS_FLASHBACK` provides an interface for the user to view the database at a particular time in the past, with the additional capacity provided by transaction back out features that allow for selective removal of the effects of individual transactions. This is different from a flashback database which moves the database back in time.

When `DBMS_FLASHBACK` is enabled, the user session uses the Flashback version of the database, and applications can execute against the Flashback version of the database.

You may want to use `DBMS_FLASHBACK` for the following reasons:

- Self-service repair: If you accidentally delete rows from a table, you can recover the deleted rows.
- Packaged applications such as email and voicemail: You can use Flashback to restore deleted email by re-inserting the deleted message into the current message box.
- Decision support system (DSS) and online analytical processing (OLAP) applications: You can perform data analysis or data modeling to track seasonal demand.

DBMS_FLASHBACK Security Model

To use the `DBMS_FLASHBACK` package, you must have the `EXECUTE` privilege on it.

DBMS_FLASHBACK Types

The following table describes the types used by `DBMS_FLASHBACK`.

Table 88-1 DBMS_FLASHBACK

Type	Description
`TXNAME_ARRAY`	Creates a `VARRAY` for holding Transaction Names or Identifiers (`XID`s)

DBMS_FLASHBACK Exceptions

DBMS_FLASHBACK creates the following error messages.

Table 88-2 DBMS_FLASHBACK Error Messages

Error	Description
`ORA-08180`	Time specified is too old
`ORA-08181`	Invalid system change number specified
`ORA-08182`	User cannot begin read-only or serializable transactions in Flashback mode
`ORA-08183`	User cannot enable Flashback within an uncommitted transaction
`ORA-08184`	User cannot enable Flashback within another Flashback session
`ORA-08185`	`SYS` cannot enable Flashback mode

DBMS_FLASHBACK Operational Notes

`DBMS_FLASHBACK` is automatically turned off when the session ends, either by disconnection or by starting another connection.

PL/SQL cursors opened in Flashback mode return rows as of the flashback time or SCN. Different concurrent sessions (connections) in the database can perform Flashback to different wall-clock times or SCNs. DML and DDL operations and distributed operations are not allowed while a session is running in Flashback mode. You can use PL/SQL cursors opened before disabling Flashback to perform DML.

Under Automatic Undo Management (AUM) mode, you can use retention control to control how far back in time to go for the version of the database you need. If you need to perform a Flashback over a 24-hour period, the DBA must set the `undo_retention` parameter to 24 hours. This way, the system retains enough undo information to regenerate the older versions of the data.

You can set the `RETENTION GUARANTEE` clause for the undo tablespace to ensure that unexpired undo is not discarded. `UNDO_RETENTION` is not in itself a guarantee because, if the system is under space pressure, unexpired undo may be overwritten with freshly generated undo. In such cases, `RETENTION GUARANTEE` prevents this. For more information, see the *Oracle Database Administrator's Guide*.

In a Flashback-enabled session, `SYSDATE` is not affected; it continues to provide the current time.

`DBMS_FLASHBACK` can be used within logon triggers to enable Flashback without changing the application code.

See Also:

Oracle Database Administrator's Guide for information on setting the minimum undo retention period.

DBMS_FLASHBACK Examples

The following example illustrates how Flashback can be used when the deletion of a senior employee triggers the deletion of all the personnel reporting to him. Using the Flashback feature, you can recover and re-insert the missing employees.

```
DROP TABLE employee;
DROP TABLE keep_scn;

REM -- Keep_scn is a temporary table to store scns that we are interested in

CREATE TABLE keep_scn (scn number);
SET ECHO ON
CREATE TABLE employee (
   employee_no   number(5) PRIMARY KEY,
   employee_name varchar2(20),
   employee_mgr  number(5)
      CONSTRAINT mgr_fkey REFERENCES EMPLOYEE ON DELETE CASCADE,
   salary        number,
   hiredate      date
);

REM -- Populate the company with employees
INSERT INTO employee VALUES (1, 'John Doe', null, 1000000, '5-jul-81');
INSERT INTO employee VALUES (10, 'Joe Johnson', 1, 500000, '12-aug-84');
INSERT INTO employee VALUES (20, 'Susie Tiger', 10, 250000, '13-dec-90');
INSERT INTO employee VALUES (100, 'Scott Tiger', 20, 200000, '3-feb-86');
INSERT INTO employee VALUES (200, 'Charles Smith', 100, 150000, '22-mar-88');
INSERT INTO employee VALUES (210, 'Jane Johnson', 100, 100000, '11-apr-87');
INSERT INTO employee VALUES (220, 'Nancy Doe', 100, 100000, '18-sep-93');
INSERT INTO employee VALUES (300, 'Gary Smith', 210, 75000, '4-nov-96');
INSERT INTO employee VALUES (310, 'Bob Smith', 210, 65000, '3-may-95');
COMMIT;

REM -- Show the entire org
SELECT lpad(' ', 2*(level-1)) || employee_name Name
FROM employee
CONNECT BY PRIOR employee_no = employee_mgr
START WITH employee_no = 1;

REM -- Sleep for a short time (approximately 10 to 20  seconds) to avoid
REM -- querying close to table creation

EXECUTE DBMS_LOCK.SLEEP(10);

REM -- Store this snapshot for later access through Flashback
```

```
DECLARE
I NUMBER;
BEGIN
I := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER;
INSERT INTO keep_scn VALUES (I);
COMMIT;
END;
/

REM -- Scott decides to retire but the transaction is done incorrectly
DELETE FROM EMPLOYEE WHERE employee_name = 'Scott Tiger';
COMMIT;

REM -- notice that all of scott's employees are gone
SELECT lpad(' ', 2*(level-1)) || employee_name Name
FROM EMPLOYEE
CONNECT BY PRIOR employee_no = employee_mgr
START WITH employee_no = 1;

REM -- Flashback to see Scott's organization
DECLARE
   restore_scn number;
BEGIN
   SELECT  scn INTO restore_scn FROM keep_scn;
   DBMS_FLASHBACK.ENABLE_AT_SYSTEM_CHANGE_NUMBER (restore_scn);
END;
/

REM -- Show Scott's org.
SELECT lpad(' ', 2*(level-1)) || employee_name Name
FROM employee
CONNECT BY PRIOR employee_no = employee_mgr
START WITH employee_no =
   (SELECT employee_no FROM employee WHERE employee_name = 'Scott Tiger');

REM -- Restore scott's organization.
DECLARE
   scotts_emp NUMBER;
   scotts_mgr NUMBER;
   CURSOR c1 IS
      SELECT employee_no, employee_name, employee_mgr, salary, hiredate
      FROM employee
      CONNECT BY PRIOR employee_no = employee_mgr
      START WITH employee_no =
         (SELECT employee_no FROM employee WHERE employee_name = 'Scott Tiger');
   c1_rec c1 % ROWTYPE;
BEGIN
   SELECT employee_no, employee_mgr INTO scotts_emp, scotts_mgr FROM employee
   WHERE employee_name = 'Scott Tiger';
   /* Open c1 in flashback mode */
   OPEN c1;
   /* Disable Flashback */
   DBMS_FLASHBACK.DISABLE;
 LOOP
   FETCH c1 INTO c1_rec;
   EXIT WHEN c1%NOTFOUND;
   /*
     Note that all the DML operations inside the loop are performed
     with Flashback disabled
   */
   IF (c1_rec.employee_mgr = scotts_emp) then
      INSERT INTO employee VALUES (c1_rec.employee_no,
         c1_rec.employee_name,
         scotts_mgr,
         c1_rec.salary,
         c1_rec.hiredate);
   ELSE
   IF (c1_rec.employee_no != scotts_emp) THEN
   INSERT INTO employee VALUES (c1_rec.employee_no,
         c1_rec.employee_name,
         c1_rec.employee_mgr,
         c1_rec.salary,
         c1_rec.hiredate);
      END IF;
    END IF;
 END LOOP;
END;
/

REM -- Show the restored organization.
select lpad(' ', 2*(level-1)) || employee_name Name
FROM employee
CONNECT BY PRIOR employee_no = employee_mgr
START WITH employee_no = 1;
```

Summary of DBMS_FLASHBACK Subprograms

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

Table 88-3 DBMS_FLASHBACK Package Subprograms

Subprogram	Description
DISABLE Procedure	Disables the Flashback mode for the entire session
ENABLE_AT_SYSTEM_CHANGE_NUMBER Procedure	Enables Flashback for the entire session. Takes an SCN as an Oracle number and sets the session snapshot to the specified number. Inside the Flashback mode, all queries return data consistent as of the specified wall-clock time or SCN
ENABLE_AT_TIME Procedure	Enables Flashback for the entire session. The snapshot time is set to the SCN that most closely matches the time specified in `query_time`
GET_SYSTEM_CHANGE_NUMBER Function	Returns the current SCN as an Oracle number. You can use the SCN to store specific snapshots
TRANSACTION_BACKOUT Procedures	Provides the mechanism to back out a transaction

DISABLE Procedure

This procedure disables the Flashback mode for the entire session.

Syntax

```
DBMS_FLASHBACK.DISABLE;
```

Examples

The following example queries the salary of an employee, Joe, on August 30, 2000:

```
EXECUTE dbms_flashback.enable_at_time('30-AUG-2000');
SELECT salary FROM emp where name = 'Joe'
EXECUTE dbms_flashback.disable;
```

ENABLE_AT_SYSTEM_CHANGE_NUMBER Procedure

This procedure takes an SCN as an input parameter and sets the session snapshot to the specified number.

In the Flashback mode, all queries return data consistent as of the specified wall-clock time or SCN. It enables Flashback for the entire session.

Syntax

```
DBMS_FLASHBACK.ENABLE_AT_SYSTEM_CHANGE_NUMBER (
   query_scn IN NUMBER);
```

Parameters

Table 88-4 ENABLE_AT_SYSTEM_CHANGE_NUMBER Procedure Parameters

Parameter	Description
`query_scn`	The system change number (SCN), a version number for the database that is incremented on every transaction commit.

ENABLE_AT_TIME Procedure

This procedure enables Flashback for the entire session.

The snapshot time is set to the SCN that most closely matches the time specified in query_time. It enables Flashback for the entire session.

Syntax

```
DBMS_FLASHBACK.ENABLE_AT_TIME (
   query_time   IN TIMESTAMP);
```

Parameters

Table 88-5 ENABLE_AT_TIME Procedure Parameters

Parameter	Description

`query_time`	This is an input parameter of type `TIMESTAMP`. A time stamp can be specified in the following ways:
	• Using the `TIMESTAMP` constructor
	`EXECUTE DBMS_FLASHBACK.ENABLE_AT_TIME(TIMESTAMP '2001-01-09 12:31:00').`
	Use the Globalization Support (NLS) format and supply a string. The format depends on the Globalization Support settings.
	• Using the `TO_TIMESTAMP` function:
	`EXECUTE DBMS_FLASHBACK.ENABLE_AT_TIME(TO_TIMESTAMP('12-02-2001 14:35:00', 'DD-MM-YYYY HH24:MI:SS'))`
	You provide the format you want to use. This example shows the `TO_TIMESTAMP` function for February 12, 2001, 2:35 PM.
	• If the time is omitted from query time, it defaults to the beginning of the day, that is, 12:00 A.M.
	• Note that if the query time contains a time zone, the time zone information is truncated.

GET_SYSTEM_CHANGE_NUMBER Function

This function returns the current SCN as an Oracle number datatype. You can obtain the current change number and store it for later use. This helps you retain specific snapshots.

Syntax

```
DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER
 RETURN NUMBER;
```

TRANSACTION_BACKOUT Procedures

This procedure provides a mechanism to back out a set of transactions. The user can call these procedures with either transaction names or transaction identifiers (`XIDS`).

The procedure analyzes the transactional dependencies, perform DMLs and generates an extensive report on the operation performed by the subprogram. This procedure does not commit the DMLs performed as part of transaction back out. However it holds all the required locks on rows and tables in the right form, so that no other dependencies can enter the system. To make the changes permanent you must explicitly commit the transaction.

A report is generated in the system tables `DBA_FLASHBACK_TRANSACTION_STATE` and `DBA_FLASHBACK_TRANSACTION_REPORT`.

Syntax

```
DBMS_FLASHBACK.TRANSACTION_BACKOUT
   numtxns            NUMBER,
   xids               XID_ARRAY,
   options            NUMBER default NOCASCADE,
   timeHint           TIMESTAMP default MINTIME);

DBMS_FLASHBACK.TRANSACTION_BACKOUT
   numtxns            NUMBER,
   xids               XID_ARRAY,
   options            NUMBER default NOCASCADE,
   scnHint            TIMESTAMP default 0   );

DBMS_FLASHBACK.TRANSACTION_BACKOUT
   numtxns           NUMBER,
   txnnames          TXNAME_ARRAY,
   options           NUMBER default NOCASCADE,
   timehint          TIMESTAMP MINTIME );

DBMS_FLASHBACK.TRANSACTION_BACKOUT
   numtxns           NUMBER,
   txnNames         TXNAME_ARRAY,
   options          NUMBER default NOCASCADE,
   scnHint          NUMBER 0);
```

Parameters

Table 88-6 TRANSACTION_BACKOUT Procedure Parameters

Parameter	Description
`numtxns`	Number of transactions passed as input
`xids`	List of transaction IDs in the form of an array
`txnnames`	List of transaction names in the form of an array
`options`	Back out dependent transactions: • `NOCASCADE` - No dependency is expected. If a dependency is found, this raises an error, with the first dependent transaction provided in the report. • `NOCASCADE_FORCE` - The user forcibly backs out the given transactions without considering the dependent transactions. The RDBMS executes the `UNDO SQL` for the given transactions in reverse order of their commit times. If no constraints break, and the result is satisfactory, the user can either `COMMIT` the changes or else `ROLL BACK`. • `NONCONFLICT_ONLY` - This option lets the user back out the changes to the nonconflicting rows of the given transactions. Note that a transaction dependency can happen due to a row conflict through either WAW or primary/unique key constraints. If the user chooses to back out only the nonconflicting rows, this does not cause any problem with database consistency, although transaction atomicity is lost. As this is a recovery operation, the user can correct the data. • `CASCADE` - This completely removes the given transactions including their dependents in a post order fashion (reverse order of commit times).
`timehint`	Time hint on the start of the transaction
`scnhint`	SCN hint on the start of the transaction

Usage Notes

Note:

For information about restrictions in using `TRANSACTION_BACKOUT`, see "Using Flashback Transaction" in the *Oracle Database Development Guide*.

- If transaction name is used, a time hint must be provided. The time hint should be a time before the start of all the given transactions to back out.
- If the SCN hint is provided, it must be before the start of the earliest transaction in the specified input set, or this raises an error and terminates. If it is not provided and the transaction has committed within undo retention, the database system is able to determine the start time.