



Flashback

A quest for zero data loss.

Tommie Grove

MyDBA

January 2010

ORACLE FLASHBACK TECHNOLOGY

Oracle Flashback Technology brings some interesting and new features for DBA's when recovery operations is needed due to user errors. The “normal” point in time recovery would usually require the DBA to restore the database or tablespace from a full backup and roll forward changes to a specific point in time. We know that this would require quite some time and effort especially when restoring from tape. Flashback can replace RMAN recovery methods in certain recovery scenarios an it should be noted that none of the flashback technologies can repair physical damage or corruptions. However combining Flashback techniques with other Oracle features like Data Guard could yield a complete disaster recovery solution in any situation thinkable.

FLASHBACK OPTIONS

There are two methods to specify a point in time with Flashback Technologies. The first method is to use a system change number(SCN) which is the most accurate, the second method is to use the TO_TIMESTAMP where the actual point in time can vary approximately 3 seconds. Flashback Database can also use user created restore points.

All Flashback technology features has similar or different pre-requisites that is needed before it would work properly.

1. Flashback Query

Flashback Query can be used to view data as it was at a specific time in the past. It requires Automatic Undo Management to be enabled and thus relies on the current undo data available in the database. To use Flashback Query simple add the “AS OF {SCN | TIMESTAMP} exp” clause at the end of your query. Note that this feature is also available in Oracle 9i.

```
SQL> connect hr/hr
```

```
SQL> select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss') from dual;
```

```
TO_CHAR(SYSDATE,'DD
```

```
-----
```

```
22-12-2009 13:05:19
```

```
SQL> select count(*) from emp;
```

```
COUNT(*)
```

```
-----
```

```
107
```

```
SQL> delete from emp;
```

```
107 rows deleted.
```

```
SQL> commit;
```

```
Commit complete.
```

```
SQL> insert into emp
```

```
2 select * from emp
```

```
3 as of timestamp
```

```
4 to_timestamp('22-12-2009 13:05:00','dd-mm-yyyy hh24:mi:ss');
```

```
107 rows created.
```

```
SQL> commit;
```

```
Commit complete.
```

```
SQL> select count(*) from emp;
```

```
COUNT(*)
```

```
-----
```

```
107
```

2. Flashback with Export

Two parameters have been added to the Export utility that will allow you to do your export using the flashback query options. These new parameters are `flashback_scn` and `flashback_time`. Setting one of these command-line parameters will cause the EXP utility to export the database

as it looked at the flashback point. If Oracle is unable to generate an export consistent with the flashback request, then the export will fail. Here is an example of exporting a database using the flashback query option :

```
exp system/manager full=y file=yesterday.dmp flashback_time='01-SEP-2001'
```

3. Flashback Version Query

Flashback Version Query can be used to view different versions of data during the specified time. The version of a row or data is created whenever a commit statement is executed. It also relies on the undo data availability in the database. The VERSIONS_XID column is a pseudo column indicating the identifier of the transaction that made the change. The valid versions pseudo columns are shown below:

Pseudo Column	Description
VERSIONS_STARTTIME	Timestamp of the first version
VERSIONS_ENDTIME	Timestamp of the last version
VERSIONS_STARTSCN	SCN of the first version
VERSIONS_ENDSCN	SCN of the last version
VERSIONS_XID	Transaction ID of the transaction that created this version
VERSIONS_OPERATION	Returns 'I' if an INSERT, 'D' if a DELETE, or 'U' if an UPDATE caused the version

```
SQL> select salary from emp where employee_id = 100;
```

```

SALARY
-----
25000

```

```
SQL> select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss') from dual;
```

```

TO_CHAR(SYSDATE,'DD
-----
22-12-2009 13:31:27

```

```
SQL> update emp set salary = 30000 where employee_id = 100;
```

1 row updated.

```
SQL> commit;
```

Commit complete.

```
SQL> update emp set salary = 35000 where employee_id = 100;
```

1 row updated.

```
SQL> commit;
```

Commit complete.

```
SQL> select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss') from dual;
```

```
TO_CHAR(SYSDATE,'DD
```

```
-----
```

```
22-12-2009 13:32:26
```

```
SQL> select versions_xid,versions_starttime,versions_endtime,versions_operation,salary
```

```
2 from emp
```

```
3 versions between timestamp to_timestamp('22-12-2009 13:31:27','dd-mm-yyyy hh24:mi:ss')
```

```
4 and to_timestamp('22-12-2009 13:32:26','dd-mm-yyyy hh24:mi:ss')
```

```
5 where employee_id = 100;
```

```
VERSIONS_XID    VERSIONS_STARTTIME    VERSIONS_ENDTIME    V    SALARY
```

```
-----
```

```
01001300F3000000 02-JAN-10 01.32.19 PM                    U    35000
```

```
06001500FF000000 02-JAN-10 01.31.40 PM    02-JAN-10 01.32.19 PM    U    30000
```

```
                  02-JAN-10 01.31.40 PM                    25000
```

4. Flashback Transactional Query

Flashback Transactional Query can be used to view all changes made to database by its users at transactional level. Supplemental logging must be enabled and compatibility initialization parameter must be set to 10.0 or higher for Flashback Transactional Query to work. Only then will the view `FLASHBACK_TRANSACTION_QUERY` contain data related to each transaction(update, insert or delete) including who executed the transaction and sql code to undo each transaction. The pseudo column `VERSIONS_XID` can be used in conjunction with Flashback Version Query to retrieve only the specific transactions for certain version of data or rows.

```
SQL> select xid, operation, logon_user, undo_sql from flashback_transaction_query
2 where xid = '06001500FF000000';
```

XID	OPERATION	LOGON_USER

UNDO_SQL		

06001500FF000000	UPDATE	HR
update "HR"."EMP" set "SALARY" = '25000' where ROWID = 'AAAL0EAAEAAAABdAAS';		
06001500FF000000	BEGIN	HR

5. Flashback Table

Flashback Table enables you to recover a table to a specified point in time. Automatic undo management and row movement at table level is a pre-requisite for flashback table operations. It should be noted that it is not possible to flashback a table before row movement was enabled. After a flashback table operation the data in the original table is still available for recovery.

```
SQL> alter table emp enable row movement;
```

Table altered.

```
SQL> select to_char(sysdate,'dd-mm-yyyy hh24:mi:ss') from dual;
```

```
TO_CHAR(SYSDATE,'DD
```

```
-----
```

```
22-12-2009 13:45:26
```

```
SQL> delete from emp where employee_id = 100;
```

```
1 row deleted.
```

```
SQL> commit;
```

```
Commit complete.
```

```
SQL> flashback table emp
```

```
  2 to timestamp
```

```
  3 to_timestamp('22-12-2009 13:45','dd-mm-yyyy hh24:mi:ss');
```

```
Flashback complete.
```

```
SQL> select count(*) from emp where employee_id = 100;
```

```
  COUNT(*)
```

```
-----
```

```
      1
```

6. Flashback Drop

Flashback Drop undoes a DROP TABLE operation on a specific table. The default behavior when dropping a table in Oracle 10g is to place the dropped table in a so called “recycle bin” which makes it possible to recover from a DROP TABLE operation. To view objects in the recycle bin, you can use USER_RECYCLEBIN or DBA_RECYCLEBIN views. The recycle bin functionality can be enabled or disabled by the initialization parameter RECYCLEBIN. By default this parameter is enabled. It is also possible to query tables within the recycle bin.

```
SQL> drop table emp;
```

Table dropped.

```
SQL> show recyclebin
```

ORIGINAL NAME	RECYCLEBIN NAME	OBJECT TYPE	DROP TIME

EMP	BIN\$fc4gElnFAhDgQAB/AQAS9g==\$0	TABLE	2009-12-22:13:52:40

```
SQL> select first_name, salary from "BIN$fc4gElnGAhDgQAB/AQAS9g==$0" where employee_id = 100;
```

FIRST_NAME	SALARY

Steven	35000

```
SQL> flashback table emp to before drop;
```

Flashback complete.

```
SQL> show recyclebin
```

```
SQL> select first_name, salary from emp where employee_id = 100;
```

FIRST_NAME	SALARY

Steven	35000

Several purge options exist :

PURGE TABLE tablename; -- Specific table.

PURGE INDEX indexname; -- Specific index.

PURGE TABLESPACE ts_name; -- All tables in a specific tablespace.

PURGE TABLESPACE ts_name USER username; -- All tables in a specific tablespace for a specific user.

```
PURGE RECYCLEBIN; -- The current users entire recycle bin.
PURGE DBA_RECYCLEBIN;
```

Several restrictions apply relating to the recycle bin:
 Only available for non-system, locally managed tablespaces.
 There is no fixed size for the recycle bin. The time an object remains in the recycle bin can vary.
 The objects in the recycle bin are restricted to query operations only (no DDL or DML).
 Flashback query operations must reference the recycle bin name.
 Tables and all dependent objects are placed into, recovered and purged from the recycle bin at the same time.
 Tables with Fine Grained Access policies are not protected by the recycle bin.
 Partitioned index-organized tables are not protected by the recycle bin.
 The recycle bin does not preserve referential integrity.

7. Flashback Restore Points

Restore point is a user-defined name that can be substituted for an SCN or point in time when used in conjunction with Flashback Database, Flashback Table, and Recovery Manager (RMAN). Restore points eliminate the need to investigate the SCN or time of a transaction and provides users with the ability to bookmark a database transaction event. Guaranteed restore points ensure that sufficient flashback logs are always maintained to get back to that restore point. This means that flashback logs will not be deleted by the Flash Recovery Area, unless they are not needed for the current guaranteed restore points. These special restore points can be created before major database changes, such as a database batch job or schema upgrade, and used for flashback if the changes need to be undone.

```
SQL> create restore point before_update_no_guarantee;
```

Restore point created.

```
SQL> create restore point before_update_guarantee guarantee flashback database;
```

Restore point created.

```
SQL > select NAME, TIME, GUARANTEE_FLASHBACK_DATABASE from v$restore_point;
```

NAME	TIME	GUA
BEFORE_UPDATE_GUARANTEE	22-DEC-09 05.20.41.000000000 PM	YES
BEFORE_UPDATE_NO_GUARANTEE	22-DEC-09 05.20.14.000000000 PM	NO

8. Flashback Database

Flashback Database provides a faster alternative to point in time recovery by undoing all changes made to a database to the point in time specified. Flashback database can also be used at a standby site to easily recover from user errors and then restoring the standby to its original state. A Flash Recovery Area(FRA) needs to be setup before enabling the flashback database feature by setting the initialization parameters `db_recovery_file_dest` and `db_recovery_file_dest_size`. This is the only place where flashback logs can be created which is needed by flashback database feature to rollback all changes as required. The FRA space is automatically managed by the database which could limit how far back in time it would be possible to flashback a database...(more maybe more)

```
V$FLASHBACK_DATABASE_LOG
```

```
V$FLASHBACK_DATABASE_STAT
```

Re-instating a primary database after a failover...

Initiating failover on standby :

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH FORCE;
```

Database altered.

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;
```

Database altered.

```
SQL> ALTER DATABASE OPEN;
```

Database altered.

```
SQL> SELECT STANDBY_BECAME_PRIMARY_SCN FROM V$DATABASE;
```

```
STANDBY_BECAME_PRIMARY_SCN
```

```
-----
```

```
425940
```

On Old primary :

```
SQL> STARTUP MOUNT
```

```
SQL> FLASHBACK DATABASE TO SCN '425940';
```

Flashback complete.

```
SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
```

Database altered.

```
SQL> SHUTDOWN IMMEDIATE
```

```
SQL> STARTUP MOUNT
```

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE  
DISCONNECT;
```

Database altered.

On new Primary :

```
SQL > ALTER SYSTEM SWITCH LOGFILE;
```

....