Goodus 기술노트 [19 회]
Flashback

Author	권웅원, 나지혜
Creation Date	**2007-04-25**
Last Updated	**2007-04-25**
Version	**1.0**

Version	변경일자	변경자(작성자)	주요내용
1	2007-04-25	권웅원, 나지혜	문서 최초 작성
2			
3			

Goodus
Breathing Life into Technology

Flashback

Contents

1. Flashback

1.1. Flashback이란?

사용자 실수에 의한 손상된 데이터를 Database의 크기와 상관없이 복구를 할수 있는기능이다. 이 Flashback 기능은 일반적인 복구에서 우려되는 데이터베이스의크기를 걱정하지 않아도 된다.
보통의 사용자 실수는 커다란시스템 장애가수반되며, 이를 복구하기 위해서는많은 자원과 시간이 필요하다. 하지만 9i에서 지원되는 flashback query와 10g에서 지원하는 다양한 flashback을통하여 손쉽게 사용자실수를 손쉽게 복구한다.
Oracle 9i 부터는 AUM 환경하에서 Flashback 기능을 이용하여 잘못된 DML operation 으로 인한 복구를 쉽게 할 수 있다. 물론 이전까지 했던 방법인 Point in Time Recovery 또한 유효하다.

△ 9i : Flashback query
△ 10g : Flashback Database
Flashback Drop
Flashback Version Query
Flashback Transaction Query
Flashback Table
Oracle Flashback Feature는 10g Standard Edition에서는 지원하지 않는다.

Note : 여기서 한 가지 짚고 넘어갈 점은 Flashback table, Flashback Database, Flashback Drop, Flashback Version Query, Flashback Transaction Query는 아래의 표와 같이 각기 다른 영역을 사용한다는 점이다.

Flashback Technologies

Flashback Operation	Implementation
Flashback Database	Flashback logs + Redo logs
Flashback Drop	Recycle bin
Flashback Version Query	Undo
Flashback Transaction Query	Undo
Flashback Table	Undo

1.2. Flashback(9i)

1.2.1. Flashback Overview

- Oracle 9i New features
- Flashback 은 사용자가 Database 의 과거 시점의 Consistent view 를 볼 수 있게 해준다.
- 사용자들은 System time or SCN 를 기초로 Read-only view 를 생성할 수 있다.
- 그 시점의 Transaction committed 부분만 볼 수 있다.
- Self-service repair 를 가능하게 해준다.
- DDL 은 지원하지 않는다.
- Flashback 은 AUM (Automate Undo Management) 사용시만 가능하다.
- Undo 정보는 System level 의 Undo retention 기간 동안만 유지한다.
- Flashback 은 Session level 에서 Enabled 할 수 있다.
- Flashback 기능을 disable 하기 전에 open 된 PL/SQL cursor 를 이용하면 disable 시킨 후에는 DML 를 통해서 self-service repair 를 할 수 있다.

▲ Undo Retention 지정
SQL> connect /as sysdba
SQL> alter system set undo_retention = <seconds> ;

이 parameter 은 dynamic 하게 변경이 가능하며 initSID.ora 에 지정할 수 있다.
undo_retention 은 각 Site 별로 업무 성격 및 Undo Size 에 따라서 적절하게 산정해서 명시해 준다. 또한 undo_management=auto 인지 확인한다.

▲ 권한 부여

```
SQL> grant execute on dbms_flashback to scott;
```

1.2.2. 예제 1 (AS OF SCN)

▲ SCOTT session 에서 SYSTEMSTAMP 를 이용하여 현재 시간을 조회하시오.

```
SQL> conn soctt/tiger
Connected.

SQL> select systimestamp from dual;

SYSTIMESTAMP
---------------------------------------------------------------------------
07-AUG-05 07.40.22.644800 AM -07:00

SQL> conn /as sysdba
Connected.

SQL> grant execute on dbms_flashback to scott;

Grant succeeded.

SQL> conn scott/tiger
Connected.

SQL> select dbms_flashback.get_system_change_number() from dual;

DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER()
-----------------------------------------
                                   851258
```

▲ SCOTT 소유의 Table 에서 부서번호가 20 인 부서원, 부서정보를 모두 삭제, Commit 하시오.

```
SQL> delete from emp where deptno=20;

5 rows deleted.

SQL> commit;

Commit complete.
```

▲ 삭제된 Data 가 잘 못 삭제된 것을 알게 되었다. 삭제된 Data 를 다시 되살리고자 한다.

※ 5 분 후에

```
SQL> !date
Sun Aug   7 07:40:59 PDT 2005

SQL> select * from emp where deptno=20;

no rows selected

SQL> select * from emp as of scn 851258 where deptno=20;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000                    20
      7876 ADAMS      CLERK           7788 12-JAN-83       1100                    20
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20

SQL> insert into emp
  2  select * from emp as of scn 851258
  3  where deptno=20;

5 rows created.
```

```
SQL> commit;

Commit complete.

SQL> select * from emp
  2  where deptno=20;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80         800                    20
      7566 JONES      MANAGER         7839 02-APR-81        2975                    20
      7788 SCOTT      ANALYST         7566 09-DEC-82        3000                    20
      7876 ADAMS      CLERK           7788 12-JAN-83        1100                    20
      7902 FORD       ANALYST         7566 03-DEC-81        3000                    20
```

1.2.3. 예제 2 (AS OF TIMESTAMP)

▲ HR_TEST01 session 에서 SYSTEMSTAMP 를 이용하여 현재 시간을 조회하시오.

```
SQL> show user
USER is "HR_TEST1"

SQL> select systimestamp from dual;

SYSTIMESTAMP
---------------------------------------------------------------------------
02-AUG-05 03.09.52.609909 AM -05:00

SQL> delete emp_test1 where department_id=20;

2 rows deleted.

SQL> commit;

Commit complete.

SQL> select employee_id from emp_test1
  2  where department_id=20;

no rows selected
```

▲ 삭제된 Data 가 잘 못 삭제된 것을 알게 되었다. 삭제된 Data 를 다시 되살리고자 한다.

※ 5 분 후에

```
SQL>  select employee_id
  2  from emp_test1 as of timestamp(to_timestamp('02-AUG-05:03:09:52', 'DD-MON-YY:HH24:MI:SS'))
  3* where department_id=20;

EMPLOYEE_ID
-----------
        201
        202

SQL> select employee_id
  2  from emp_test1
  3  where department_id=20;

no rows selected

SQL> insert into emp_test1
  2  select * from emp_test1 as of timestamp(to_timestamp('02-AUG-05:03:09:52', 'DD-MON-YY:HH24:MI:SS'))
  3  where department_id=20;

2 rows created.

SQL> commit;
```

```
Commit complete.

SQL>   select employee_id
  2   from emp_test1
  3* where department_id=20;

EMPLOYEE_ID
-----------
        201
        202
```

1.2.4. 예제 3 (Package, SCN / timestamp)

▲ HR_TEST01 session 에서 SYSTEMSTAMP 를 이용하여 현재 시간을 조회하시오.

```
SQL> show user
USER is "HR_TEST1"

SQL> select systimestamp from dual;

SYSTIMESTAMP
---------------------------------------------------------------------------
02-AUG-05 03.31.21.752682 AM -05:00

SQL> conn /as sysdba
Connected.

SQL> select dbms_flashback.get_system_change_number() from dual;

DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER()
-----------------------------------------
                                   107270

SQL> conn hr_test1/oracle
Connected.

SQL> delete from emp_test1
  2   where department_id=20;

2 rows deleted.

SQL> delete from dept_test1
  2    where department_id=20;

1 row deleted.

SQL> commit;

Commit complete.
```

▲ 삭제된 Data 가 잘 못 삭제된 것을 알게 되었다. 삭제된 Data 를 다시 되살리고자 한다.

※ 5 분 후에

```
SQL>   conn /as sysdba

SQL>   grant execute on dbms_flashback to hr_test1;

Grant succeeded.

SQL> conn hr_test01/oracle

SQL> begin
dbms_flashback.enable_at_system_change_number(107270);
end;
/
```

```
PL/SQL procedure successfully completed.

SQL> select employee_id from emp_test1
where department_id=20;

EMPLOYEE_ID
-----------
        201
        202

SQL> select * from dept_test1
where department_id=20;

DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
           20 Marketing                             201        1800

SQL> begin
dbms_flashback.disable;
end;
/

PL/SQL procedure successfully completed.

SQL> declare
   cursor emp_curs is
      select * from emp_test1 where department_id = 20;
   cursor dept_curs is
      select * from dept_test1 where department_id = 20;

emp_rec emp_curs%ROWTYPE;

dept_rec dept_curs%ROWTYPE;

begin
dbms_flashback.enable_at_system_change_number(107270);
-- dbms_flashback.enable_at_time(to_timestamp(' 02-AUG-05:03.31:21','DD-MON-YY:HH24:MI:SS'));

open emp_curs;
open dept_curs;
dbms_flashback.disable;

loop
   fetch dept_curs into dept_rec;
   exit when dept_curs%NOTFOUND;
   insert into dept_test1 values(dept_rec.DEPARTMENT_ID,
      dept_rec.DEPARTMENT_NAME, dept_rec.MANAGER_ID,
      dept_rec.LOCATION_ID);
end loop;

loop
      fetch emp_curs into emp_rec;
      exit when emp_curs%NOTFOUND;
      insert into emp_test1 values(emp_rec.employee_id, emp_rec.first_name,emp_rec.last_name, emp_rec.email,
                                                   emp_rec.phone_number,emp_rec.hire_date, emp_rec.job_id,
emp_rec.salary,
                                                   emp_rec.commission_pct, emp_rec.manager_id,
emp_rec.department_id);
end loop;

end;
/

PL/SQL procedure successfully completed.
```

※ 또는, 위 내용을 Procedure 를 생성해서 복구 할 수도 있다.(flash.sql)

```
create or replace procedure exam_flash
as
        cursor tmp_curs is
                select * from emp where deptno = 20;
        emp_rec tmp_curs%ROWTYPE;

begin
        dbms_flashback.enable_at_time('03/08/21 10:47:32 ');
        open tmp_curs;
        dbms_flashback.disable;

        loop
                fetch tmp_curs into emp_rec;
                exit when tmp_curs%NOTFOUND;
                insert into emp values (
                        emp_rec.empno, emp_rec.ename, emp_rec.job,
                        emp_rec.mgr,emp_rec.hiredate, emp_rec.sal,
                        emp_rec.comm, emp_rec.deptno
                );
        end loop;
end;
/
```

Flashback 을 이용해 과거 데이터 복구
SQL> @flash /* exam_flash procedure 생성 */
SQL> exec exam_flash

SQL> select employee_id from emp_test1 where department_id = 20;

```
EMPLOYEE_ID
-----------
        201
        202
```

SQL> select * from dept_test1 where department_id=20;

```
DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
           20 Marketing                             201        1800
```

SQL> commit;

Commit complete.

1.3. Flashback(10g)

1.3.1. Flashback Database

Flashback Database 개요
Oracle Database 10g 이전까지는 transactional point-in-time recovery를 위해서는 backup용 file과 redo log file을 이용하여 원하는 시간까지의 복구를 하였었다. 그러나 이 방법은 backup용 file이 오래된 것이며, archive log가 많이 쌓여 있을 때는 많은 시간이 소요된다. Oracle Database 10g부터는 flashback database를 이용하여 좀 더 빠른 recovery가 가능하게 되었다.

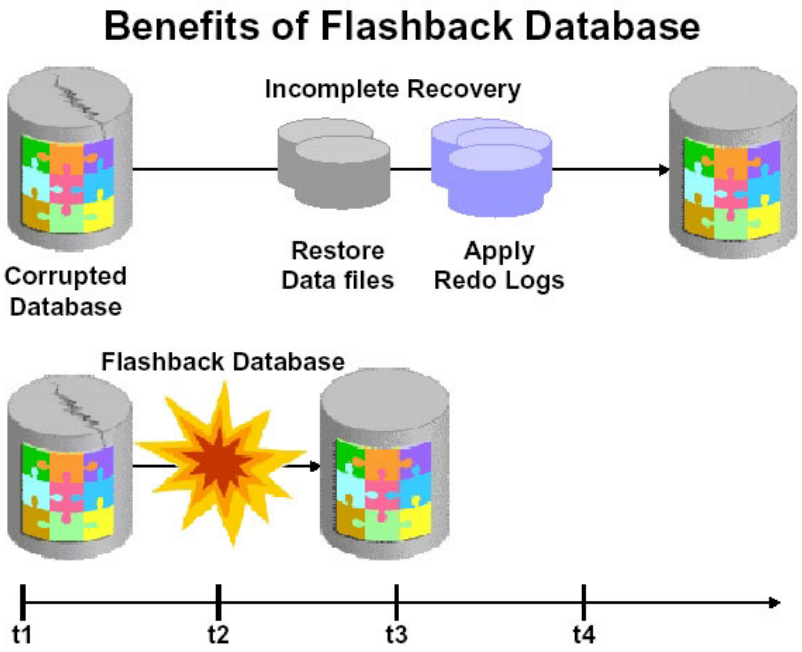
Flashback Database는 오라클 데이터베이스를 과거 시점으로 되돌리고, 논리적인 데이터 손상 또는 사용자 실수로 인해 발생한 문제를 해결할 수 있게 한다. Flashback Database는 데이터베이스를 위한 '되감기 버튼'과도 같다. 데이터베이스 백업본을 이용하여 복구 작업을 수행하지 않고도 데이터베이스를 과거의 시점으로 되돌릴 수 있다. 포인트-인-타임 복구 작업에는 테이프에 저장된 데이터베이스 백업을 복구하는 시간이 불필요하므로, 한층 신속한

복구가 가능하다.

Flashback Database 기능은 RMAN, SQL*Plus에서 FLASHBACK DATABASE 커맨드를 이용하여 실행되며, 그 효과면에서 일반적인 포인트-인-타임 복구 방식과 매우 유사하다. 이 기능을 이용하면 과거 특정 시점으로 데이터베이스의 상태를 되돌릴 수 있다. Flashback Database 기능을 활성화하려면, 먼저 Flash Recovery Area를 설정해야 한다. Flash Recovery Area는 Oracle Database 10g에 추가된 새로운 기능으로, 오라클 데이터베이스 복구 관련 파일 및 작업을 위한 통합적인 저장 공간으로 활용된다. 복구 영역에는 Flash Database 로그 이외에도 아카이브 리두 로그, RMAN 백업 등이 저장된다.

오라클은 Flash Recovery Area 내에 Flashback Log를 자동 생성/관리 한다. Flash Recovery Area에는 쿼타(quota)가 설정되며, 따라서 Flashback Log에는 공간 제한이 적용된다. Flashback Log의 사이즈는 로그 저장 기간 동안의 데이터베이스 변경 과정에서 발생한 읽기/쓰기 작업량에 따라 크게 달라진다. 오래된 블록 버전의 복사본은 Flashback Log에 기록된다. 하루 동안 10%의 데이터베이스 블록이 업데이트되었다면, 24 시간 동안의 Flashback Log 사이즈는 전체 데이터베이스 용량의 10 분의 1 수준이 될 것이다. 데이터베이스를 과거 시점으로 복구하는 과정에서 더 많은 디스크 공간이 필요한 경우, DBA는 디스크 쿼타를 다이내믹하게 확장할 수 있다.

Flashback database의 사용 용도는 logical data corruption이나 user error시 유용하다.
(Physical data corruption은 H/W 문제이기 때문에 Flashback database로 recovery는 불가능하다.) Flashback Database의 장점은 기존의 traditional point-in-time recovery에 비해 매우 빠른 recovery가 가능하다는 것이다. 이러한 빠른 성능을 낼 수 있는 이유는 flashback database는 database의 크기에 비례해서 recovery시간이 늘어나는 것이 아니라, 변경된 data의 양에 비례해서 recovery시간이 걸린다는 점이다.



위의 그림, 앞의 설명과 같이 Flashback Database는 매우 빠른 시간의 recovery를 가능하게 한다.

Flashback Database를 수행하기 위한 3가지 구성요소

1. Archive Mode
   Flashback Database 기능을 적용하기 위해서는 Archive Mode로 설정하여야 한다.

2. Flashback Log File
   Flashback Log File은 오라클 Database를 구성하는 Block(변경되기 이전의 이미지 Block)을 저장하는 로그 파일로서 10g에서 새롭게 소개되고 있는 데이터베이스 복구영역(database recovery area)에 생성되어진다.
   기존의 redo log와의 차이점
   - redo log의 경우에는 archive할 수 있는 기능이 함께 제공되었지만, Flashback Log는 archive 기능이 따로 제공될 필요가 없다.(db_recovery_file_dest, db_recovery_file_dest_size)
   - Flashback Log의 경우에는 물리적인 database 복구에는 사용될수 없다는 점이다.

3. RVWR Background Process
   Flashback Database 기능이 활성화 되어지면, rvwr이라는 background process가 시작된다.

• Flashback Database 테스트

Database 에 Flashback 기능이 ON 되어 있는지 확인한다.

```
SQL> select FLASHBACK_ON from v$database;

FLASHBACK_ON
------------------
NO

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount
ORACLE instance started.

Total System Global Area  394264576 bytes
Fixed Size                  2006632 bytes
Variable Size             218104216 bytes
Database Buffers          167772160 bytes
Redo Buffers                6381568 bytes
Database mounted.

SQL> alter database flashback on;

Database altered.

SQL> select FLASHBACK_ON from v$database;

FLASHBACK_ON
------------------
YES

SQL> alter database open;

Database altered.
```

Test Case 생성

```
SQL> conn scott/tiger
Connected.

SQL> create table flash as select * from user_objects;

Table created.

SQL> select count(*) from flash;

  COUNT(*)
----------
        65

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
   25833961
```

```
SQL> truncate table flash;

Table truncated.

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
   25833994

SQL> select count(*) from flash;

  COUNT(*)
----------
         0
```

Flashback Database 를 위해 Instance 를 종료시킨다.

```
SQL> conn /as sysdba
Connected.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Flashback Database 를 위해 Instance 를 Mount 시킨다.

```
SQL> startup mount
ORACLE instance started.

Total System Global Area 188743680 bytes
Fixed Size                   778036 bytes
Variable Size             162537676 bytes
Database Buffers           25165824 bytes
Redo Buffers                 262144 bytes
Database mounted.
```

원하는 시점으로 되돌아 가기 위해 조금전에 기록했던 SCN 으로 Flash Back 한다.

```
SQL> flashback database to scn 25833961;

Flashback complete.
```

```
alter_<SID>.log

Incomplete Recovery applied until change 25833975
Flashback Media Recovery Complete
Completed: flashback database to scn 25833961
```

Database 를 read only로 open 하여 Data를 확인 후에, Resetlogs 로 Open 하여 truncate 전의 데이터를 복구한다.

```
SQL> alter database open;
alter database open
*
ERROR at line 1:
ORA-01589: must use RESETLOGS or NORESETLOGS option for database open

SQL> alter database open read only;

Database altered.

SQL> select count(*) from scott.flash;
```

```
  COUNT(*)
----------
        65

SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup mount;
ORACLE instance started.

Total System Global Area  394264576 bytes
Fixed Size                  2006632 bytes
Variable Size             218104216 bytes
Database Buffers          167772160 bytes
Redo Buffers                6381568 bytes
Database mounted.

SQL> alter database open resetlogs;
Database altered.

SQL> select count(*) from scott.flash;

  COUNT(*)
----------
        65
```

관련 view
V$FLASHBACK_DATABASE_LOG;
V$FLASHBACK_DATABASE_STAT;

1.3.2. Flashback Drop

사용자와 DBA 모두에게 있어 실수로 오브젝트를 드롭(drop) 처리하는 경우는 흔하게 발생한다. 사용자들이 실수를 깨달았을 때에는 이미 때가 늦다. 과거에는 이렇게 드롭 처리된 테이블, 인덱스, 제약조건, 트리거 등을 쉽게 복구할 수 있는 방법이 없었다. Flashback Drop 은 Oracle Database 10g 환경의 오브젝트 드롭 작업을 위한 안전망을 제공한다. 사용자가 테이블을 드롭하면, 오라클은 드롭된 오브젝트를 Recyble Bin 에 보관한다.

△ 10g 에서 DROP TABLE 을 하게 되면 기본적으로 실제 그것을 DROP 하는 것보다 RECYCLE BIN 에 이동시키거나 이름을 바꾸게 된다.
△ Drop 된 Table을 복구한다.
△ Drop table이 완전 drop 되지 않고, window의 휴지통과 같은 recyclebin에 보관된다.
△ 이 drop된 table은 완전 삭제를 위해서는 purge 작업이 필요하며, space가 부족한 경우에는 자동 reuse된다.
△ Drop되어 recyclebin에 있는 bin$xxxxxx table에 대한 직접조회도 가능함.
△ 관련 view
- dba_recyclebin, user_recyclebin
△ 관련 parameter
_recyclebin = FALSE : recyclebin 기능을 사용하지 않는 경우 False로 지정
△ 제약사항 : table이 system tablespace에 있는 object는 복구 불가.
locally managed tablespace에 위치해 있는 table만 복구 가능.
Table이 복구되면 그 table의 index, trigger 등의 연관된 object도 함께 복구된다.
(bitmap join index제외)
Partioned index-organized table은 recycle bin에 의해 보호 받지 못한다.
recycle bin은 참조 무결성을 보장하지 않는다.

- **예제 1**
  1) Table을 drop 하기 (장애 만들기)

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
EMP                            TABLE
DEPT                           TABLE
BONUS                          TABLE
SALGRADE                       TABLE
DUMMY                          TABLE

SQL> drop table emp;

Table dropped.

SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BIN$Lr5d8GB+sLbgQz36Y80wtg==$0 TABLE
DEPT                           TABLE
BONUS                          TABLE
SALGRADE                       TABLE
DUMMY                          TABLE
```

Recycle Bin 보기

```
SQL> select object_name, original_name, type, DROPTIME, can_undrop from user_recyclebin
  2  order by droptime;

OBJECT_NAME                    ORIGINAL_NAME          TYPE                       DROPTIME            CAN
------------------------------ ---------------------- -------------------------- ------------------- ---
BIN$LumGwpeDSHHgRAAQgzW92A==$0 EMP                    TABLE                      2007-04-25:15:01:08 YES
```

2) Drop된 Table 복구하기 1

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

Flashback complete.

SQL> select object_name, original_name, type from user_recyclebin;

no rows selected

SQL> select table_name from user_tables;

TABLE_NAME
------------------------------
DUMMY
SALGRADE
BONUS
DEPT
EMP
```

3) Drop된 table 복구하기 2 (동일 이름의 table이 이미 있는 경우, 다른 이름으로 복구하기)

```
SQL> flashback table scott.emp to before drop rename to dropped_emp;
```

Drop된 table 완전 삭제하기

```
SQL> drop table scott.emp purge;                    -- drop 시 바로 purge하는 경우
SQL> purge recyclebin;
or
SQL> purge dba_recyclebin;
or
```

```
SQL> purge table scott.emp
```

아래는 몇가지 PURGE 옵션의 예 입니다.

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;	The whole recycle bin

- 예제 2

휴지통(recyclebin)에 같은 이름의 table 이 여러개 있을 때 PURGE and FLASHBACK TO BEFORE DROP 방법

같은 이름을 가지는 table 이 휴지통(recyclebin)에 하나 이상있을 경우 다루는 방법입니다.
table 을 PURGE 하는 경우 가장 오래된 table 이 휴지통에서 PURGE 되고 table 을 restore(FLASHBACK BEFORE DROP)하는 경우 가장 최근의 table 이 저장됩니다.

Example
========
5 개의 table 을 생성하고 drop 하자..

```
$ sqlplus scott/tiger
SQL> CREATE TABLE t1(a NUMBER);
SQL> DROP TABLE t1;
SQL> CREATE TABLE t1(a varchar2(10));
SQL> DROP TABLE t1;
SQL> CREATE TABLE t1(a date);
SQL> DROP TABLE t1;
SQL> CREATE TABLE t1(a varchar2(5));
SQL> DROP TABLE t1;
SQL> CREATE TABLE t1(a number);
SQL> DROP TABLE t1;

SQL> SELECT object_name,original_name,droptime,dropscn FROM recyclebin;

OBJECT_NAME                    ORIGINAL_NAME DROPTIME                 DROPSCN
------------------------------ ------------- ------------------------ --------------------------
BIN$14s6rPqoHQngMEWYESkRng==$0 T1            2004-04-08:17:40:21      2039154
BIN$14s6rPqmHQngMEWYESkRng==$0 T1            2004-04-08:17:40:19      2039107
BIN$14s6rPqnHQngMEWYESkRng==$0 T1            2004-04-08:17:40:21      2039133
BIN$14s6rPqpHQngMEWYESkRng==$0 T1            2004-04-08:17:40:21      2039231
BIN$14s6rPqqHQngMEWYESkRng==$0 T1            2004-04-08:17:40:22      2039252
```

만일 table t1 을 purge 한다면 dropscn=2039107 가 purge 될것이다.

```
SQL> PURGE TABLE t1;

Table purged.

SQL> SELECT object_name,original_name,droptime,dropscn FROM recyclebin;

OBJECT_NAME                    ORIGINAL_NAME  DROPTIME             DROPSCN
------------------------------ -------------- -------------------- ----------
BIN$14s6rPqoHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21  2039154
BIN$14s6rPqnHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21  2039133
BIN$14s6rPqpHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21  2039231
BIN$14s6rPqqHQngMEWYESkRng==$0 T1             2004-04-08:17:40:22  2039252
```

만일 table t1 을 restore 한다면 dropscn=2039252 이 restore 할 것이다.

```
SQL> FLASHBACK TABLE t1 TO BEFORE DROP ;

Flashback complete.

SQL> SELECT object_name,original_name,droptime,dropscn FROM recyclebin;

OBJECT_NAME                    ORIGINAL_NAME  DROPTIME            DROPSCN
------------------------------ -------------- ------------------- ----------
BIN$14s6rPqoHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21 2039154
BIN$14s6rPqnHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21 2039133
BIN$14s6rPqpHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21 2039231
```

=>이 문제를 해결하기 위해서..
==========
이 문제를 극복하기위해서 우리는 original 이름 대신에 drop된 object name을 사용하면 된다.
object name 은 unique 하므로 원하는 것을 purge 와 restore 할 수 있다.

Examples:

```
SQL> FLASHBACK TABLE "BIN$14s6rPqoHQngMEWYESkRng==$0" TO BEFORE DROP RENAME TO t2;

Flashback complete.

SQL> select tname from tab;

TNAME
------------------------------
DEPT
EMP
BIN$14s6rPqnHQngMEWYESkRng==$0
T1
BIN$14s6rPqpHQngMEWYESkRng==$0
T2

6 rows selected.


SQL> SELECT object_name,original_name,droptime,dropscn FROM recyclebin;

OBJECT_NAME                    ORIGINAL_NAME  DROPTIME            DROPSCN
------------------------------ -------------- ------------------- ----------
BIN$14s6rPqnHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21 2039133
BIN$14s6rPqpHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21 2039231
```

비슷한 방법으로 purge 할수 있다.

```
SQL> PURGE TABLE "BIN$14s6rPqnHQngMEWYESkRng==$0";

Table purged.

SQL> SELECT object_name,original_name,droptime,dropscn FROM recyclebin;

OBJECT_NAME                    ORIGINAL_NAME  DROPTIME            DROPSCN
------------------------------ -------------- ------------------- ----------
BIN$14s6rPqpHQngMEWYESkRng==$0 T1             2004-04-08:17:40:21 2039231
```

1.3.3. Flashback Versions Query

△ 과거의 어떤시점의 정보를 시간과 SCN(SystemChange Number)를 이용하여 Query하는 기능.
△ 9i 부터지원된 Flashback Query가 있으며, 10g에서는 그 기능이 확장되어 Versions between을 이용해서 일정시점이 아닌 시간간격의 데이터를 조회할 수 있는 기능.
△ Flashback versions query에 의해 추출된 row들은 transaction에 의해 변화된 row들의 history를 보여줌. 이 기능은 data가 어떻게 바뀌었는지 auditing 기능을 가능하게 하며 commit된 데이터만 추출 함.
△ Flashback versions query를 통해서 알수 있는 transaction id를 통하여 더 추가적인 정보를 Flashback Transaction Query를 통해 얻을수 있다.
△ DDL이 수행되어 table의 구조가 바뀌면 사용불가.
△ Flashback versions query는 undo를 이용하여 과거 데이터를 읽어오는 것은 undo_retention 값과 undo size에 의해 자동으로 관리됨. 만약 undo_retention이 아주 크다고 하더라도, undo size가 작아서 undo를 보관하지 않고 재사용하게 되면 flashback versions query가 수행되지 않을 수 있음.
△ Versions between은 시간과 SCN으로 지정 할 수 있음
△ 이기능을 지원하기 위해 scn_to_timestamp 와 timestamp_to_scn function이 지원된다.

• **과거의 시점에 대한 SCN 확인.**

```
SQL> select timestamp_to_scn(to_timestamp('20060213 171201', 'yyyymmdd hh24miss')) SCN from dual;
        SCN
        ------------
        8268801520810
```

• **과거의 SCN을 이용하여 Time 확인.**

```
SQL> select scn_to_timestamp(8268801520810) TIME from dual;
        TIME
        --------------------------
        2006/02/13 17:12:01.000000000
```

△ Versions Query의 Pseudo column (Select절에 사용할 수 있음)
- Versions_startscn,
- Versions_starttime
- Versions_endscn
- Versions_endtime
- Versions_xid
- Versions_operation

△ 주의 : undo retention 보다 이전의 version을 query하면 ora-30052 : invalid lower limit snapshot expression 발생함.

• **예제**

Data 변경(Update) 하기

```
SQL> conn scott/tiger
Connected.
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE           SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ----------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80          800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81         1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81         1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81         2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81         1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81         2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81         2450                    10
      7788 SCOTT      ANALYST         7566 09-DEC-82         3000                    20
      7839 KING       PRESIDENT            17-NOV-81         5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81         1500          0         30
      7876 ADAMS      CLERK           7788 12-JAN-83         1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81          950                    30
      7902 FORD       ANALYST         7566 03-DEC-81         3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82         1300                    10

14 rows selected.
```

```
SQL> update emp
  2  set comm=1111
  3  where empno=7934;

1 row updated.

SQL> commit;

Commit complete.

SQL> update emp
  2  set comm=2222
  3  where empno=7934;

1 row updated.

SQL> rollback;

Rollback complete.

SQL> update emp
  2  set comm=2222
  3  where empno=7934;

1 row updated.

SQL> commit;

Commit complete.

SQL> update emp
  2  set comm=3333
  3  where empno=7934;

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7788 SCOTT      ANALYST         7566 1982-12-09 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7876 ADAMS      CLERK           7788 1983-01-12 00:00:00       1100                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300       3333         10
```

```
14 rows selected.
```

2007 2월 25일 15시 50분 ~ 16시 00분 까지 empno가 7934인 data가 변한 내역조회

```
SQL> col start_time format a21
SQL> col end_time format a21
SQL> set lines 200

SQL> SELECT empno, ename, sal, comm, deptno,
  2  versions_xid AS XID,
  3  versions_operation as operation,
  4  versions_startscn AS START_SCN,
  5  versions_starttime as start_time,
  6  versions_endscn AS END_SCN,
  7  versions_endtime as end_time
  8  FROM emp VERSIONS BETWEEN TIMESTAMP
  9      TO_TIMESTAMP('2007-04-25 15:50:00','YYYY-MM-DD HH24:MI:SS')
 10  AND TO_TIMESTAMP('2007-04-25 16:00:00','YYYY-MM-DD HH24:MI:SS')
 11  WHERE empno=7934;

 EMPNO ENAME       SAL    COMM   DEPTNO XID              O  START_SCN START_TIME            END_SCN    END_TIME
------ -------- ------ ------- -------- ---------------- - ---------- --------------------- ---------- ---------------------
  7934 MILLER     1300    3333       10 0009000000000988 U   25837411 25-APR-07 03.58.31 PM
  7934 MILLER     1300    2222       10 0003000D00000994 U   25837379 25-APR-07 03.57.01 PM  25837411 25-APR-07 03.58.31 PM
  7934 MILLER     1300    1111       10 0006001C00000985 U   25837354 25-APR-07 03.56.04 PM  25837379 25-APR-07 03.57.01 PM
  7934 MILLER     1300               10                                                     25837354 25-APR-07 03.56.04 PM

SQL> SELECT empno, ename, sal, comm, deptno,
  2  versions_xid AS XID,
  3  versions_operation as operation,
  4  versions_startscn AS START_SCN,
  5  versions_starttime as start_time,
  6  versions_endscn AS END_SCN,
  7  versions_endtime as end_time
  8  FROM emp VERSIONS BETWEEN TIMESTAMP
  9      systimestamp - interval '10' minute and systimestamp - interval '1' minute
 10  WHERE empno=7934;

 EMPNO ENAME       SAL    COMM   DEPTNO XID              O  START_SCN START_TIME            END_SCN    END_TIME
------ -------- ------ ------- -------- ---------------- - ---------- --------------------- ---------- ---------------------
  7934 MILLER     1300    3333       10 0009000000000988 U   25837411 25-APR-07 03.58.31 PM
  7934 MILLER     1300    2222       10 0003000D00000994 U   25837379 25-APR-07 03.57.01 PM  25837411 25-APR-07 03.58.31 PM
  7934 MILLER     1300    1111       10 0006001C00000985 U   25837354 25-APR-07 03.56.04 PM  25837379 25-APR-07 03.57.01 PM
  7934 MILLER     1300               10                                                     25837354 25-APR-07 03.56.04 PM
```

조회결과로 Null ➔ 1111➔ 2222 ➔ 3333 으로 변경된 내역을 볼 수 있다.

1.3.4. Flashback Query

Oracle9i 에서 처음 소개된 Flashback Query 는 과거 시점의 데이터를 조회하는 기능을 제공한다. 기본적으로 데이터베이스는 가장 최근에 커밋된 데이터를 기준으로 작업을 수행한다. 하지만 과거의 특정 시점을 기준으로 데이터베이스를 조회하고자 한다면, Flashback Query 기능을 이용할 수 있다. Flashback Query 는 특정 시점 또는 SCN(System Change Number)을 기준으로, 해당 시점에 커밋된 데이터를 조회할 수 있게 한다. Flashback Query 메커니즘은 Automatic Undo Management 를 이용하는 경우 가장 효과적으로 동작한다.

오라클 데이터베이스는 언두(undo)를 중요한 데이터베이스 오브젝트로 관리한다. 언두 데이터는 영구적으로 저장/관리되며 데이터베이스 시스템에 크래쉬, 셧다운이 발생하는 경우에도 유지된다. 또 다른 데이터베이스 오브젝트와 함께 데이터베이스 버퍼 캐시를 공유하므로 성능 보장이 가능하다. 오라클 데이터베이스는 트랜잭션이 커밋된 이후에도 언두 데이터를 관리하고, 필요한 경우 논리적 손상으로부터 복구할 수 있게 한다.

Flashback

오라클 데이터베이스의 관리자는 보존할 언두 데이터의 양을 명시적으로 지정할 수 있다. 시스템은 새로운 트랜잭션의 언두 데이터를 생성하기 위해 만료된 언두 데이터를 자동으로 삭제한다. 언두 데이터의 보존 기간은 롱-러닝(long-running) 쿼리의 실행 시간 또는 논리적 손상에 대한 복구 요구사항에 따라 다르게 설정된다. 또는 언두 보존 기간을 설정하지 않고 데이터베이스가 알아서 최적의 보존 정책을 관리하도록 할 수도 있다. 그러면 데이터베이스는 롱-러닝 쿼리에 대한 실행 시간과 논리적 손상의 복구를 최대한 보장할 수 있는 방안을 자동으로 적용한다. 디폴트 상태에서 언두 데이터의 보존은 보장되지 않는다. 시스템은 현재 진행 중인 트랜잭션의 언두 데이터 기록을 위해 필요한 경우, 언제든 오래된 언두 데이터를 만료 처리할 수 있다.

10g R1 부터는 UNDO_RETENTION 이 5 일 이상으로 지정되어 있는 경우, 5 일 또는 그 이상 경과한 과거의 데이터를 쿼리할 수 있는 기능이 추가되었다. 오라클은 Undo Tablespace 데이터파일에 충분한 공간이 남아 있는 한, 언두 데이터를 유지한다. 데이터베이스에서 Flashback Query 와 기타 언두 데이터 관련 플래시백 기능을 활성화하기 위한 방법이 아래와 같다:

1. 데이터베이스가 Undo Tablespace 를 사용하고 있는지 확인한다. Undo Tablespace 를 사용하려면 UNDO_MANAGEMENT 초기화 매개변수를 AUTO 로 설정해 놓아야 한다.
2. 가장 긴 실행 시간을 갖는 쿼리를 성공적으로 복구할 수 있는 시간, 또는 사용자 에러로부터 복구하기에 충분한 시간으로 UNDO_RETENTION 초기화 매개변수를 설정한다.
3. 만료되지 않은 언두 데이터가 덮어씌워지지 않도록, 언두 테이블스페이스에 RETENTION GUARANTEE 조건을 추가한다.

Flashback Query 기능을 이용하면 과거의 데이터 시점의 데이터를 확인할 수 있을 뿐 아니라 데이터를 처리하는 방법을 선택하는 것도 가능하다. 분석 작업을 수행한 후에 모든 변경 작업을 취소하거나, 변경 데이터를 캡처하여 다른 작업에 활용할 수도 있다. Flashback Query 메커니즘은 다양한 상황에서 활용될 수 있는 유연성을 제공한다. 몇 가지 활용 예가 아래와 같다:

- 과거 시점의 데이터를 조회.
- 현재 데이터와 과거 데이터를 비교. (개별 로우를 비교하거나 Intersection, Union 등의 조건을 이용하여 복잡한 비교 작업을 수행할 수도 있다.)
- 삭제, 변경된 데이터의 복구.

△ Oracle9i에서 부터 지난 시점의 데이터를 질의 하기 위한 DBMS_PACKAGE를 제공 했으며 10g에서는 훨씬 기능을 유연하게 발전 시켰다.
△ Flashback Query는 AS OF 절을 사용하여 해당 시점에서의 데이터 값에 대한 질의가 가능하며, 이 기능은 DBMS_FLASHBACK 패키지의 기능과 유사하다.
△ Flashback versions query는 과거의 일정 시간구간에서 조회하는 것에 비해 Flashback query는, 과거의 일정한 시간에서 query를 하는 것.
△ Database는 현재의 시간이지만, 수행하는 SQL은 혼자 과거의 정보를 보게 됨.

• 예제

```
-- Data 삭제(장애 만들기)

SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01 00:00:00	2850		30
7782	CLARK	MANAGER	7839	1981-06-09 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	1982-12-09 00:00:00	3000		20
7839	KING	PRESIDENT		1981-11-17 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	1983-01-12 00:00:00	1100		20
7900	JAMES	CLERK	7698	1981-12-03 00:00:00	950		30
7902	FORD	ANALYST	7566	1981-12-03 00:00:00	3000		20
7934	**MILLER**	**CLERK**	**7782**	**1982-01-23 00:00:00**	**1300**	**3333**	**10**

14 rows selected.

```
SQL> delete emp
  2  where empno=7934;

1 row deleted.

SQL> commit;

Commit complete.

SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01 00:00:00	2850		30
7782	CLARK	MANAGER	7839	1981-06-09 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	1982-12-09 00:00:00	3000		20
7839	KING	PRESIDENT		1981-11-17 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	1983-01-12 00:00:00	1100		20
7900	JAMES	CLERK	7698	1981-12-03 00:00:00	950		30
7902	FORD	ANALYST	7566	1981-12-03 00:00:00	3000		20

13 rows selected. (1건이 삭제 되었음)

-- 1시간 전 Data를 구하기

```
SQL> select * from emp as of timestamp ( systimestamp - interval '1' hour);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17 00:00:00	800		20
7499	ALLEN	SALESMAN	7698	1981-02-20 00:00:00	1600	300	30
7521	WARD	SALESMAN	7698	1981-02-22 00:00:00	1250	500	30
7566	JONES	MANAGER	7839	1981-04-02 00:00:00	2975		20
7654	MARTIN	SALESMAN	7698	1981-09-28 00:00:00	1250	1400	30
7698	BLAKE	MANAGER	7839	1981-05-01 00:00:00	2850		30
7782	CLARK	MANAGER	7839	1981-06-09 00:00:00	2450		10
7788	SCOTT	ANALYST	7566	1982-12-09 00:00:00	3000		20
7839	KING	PRESIDENT		1981-11-17 00:00:00	5000		10
7844	TURNER	SALESMAN	7698	1981-09-08 00:00:00	1500	0	30
7876	ADAMS	CLERK	7788	1983-01-12 00:00:00	1100		20
7900	JAMES	CLERK	7698	1981-12-03 00:00:00	950		30
7902	FORD	ANALYST	7566	1981-12-03 00:00:00	3000		20
7934	**MILLER**	**CLERK**	**7782**	**1982-01-23 00:00:00**	**1300**	**3333**	**10**

14 rows selected.

-- 1분 전 Data를 구하기

※ delete 후 바로 조회하면 아직 delete 되지 않은 것으로 보인다.

```
SQL> select * from emp as of timestamp ( systimestamp - interval '1' minute);
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

```
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7788 SCOTT      ANALYST         7566 1982-12-09 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
      7876 ADAMS      CLERK           7788 1983-01-12 00:00:00       1100                    20
      7900 JAMES      CLERK           7698 1981-12-03 00:00:00        950                    30
      7902 FORD       ANALYST         7566 1981-12-03 00:00:00       3000                    20
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300       3333         10

14 rows selected.


-- 1시간전 Data와 현재 Data의 차이를 알고 싶을때.
-- 즉, 1시간전과 같지 않은 데이터를 모두 찾는다.
SQL> select * from emp as of timestamp ( systimestamp - interval '1' hour)
  2  minus
  3  select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------------------- ---------- ---------- ----------
      7934 MILLER     CLERK           7782 1982-01-23 00:00:00       1300       3333         10

※ 1시간 전의 Table을 Backup 해 놓을수 있다.
SQL> create table emp_back
  2  as select * from emp as of timestamp ( systimestamp - interval '1' hour);

Table created.

-- 급하게 복구를 해야 할때. 약 1시간전에 많은 건수를 삭제한 경우.
SQL> insert into emp
  2  select * from emp as of timestamp ( systimestamp - interval '1' hour)
  3  minus
  4  select * from emp;

1 row created.

SQL> commit;

Commit complete.

SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------------------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 1980-12-17 00:00:00        800                    20
      7499 ALLEN      SALESMAN        7698 1981-02-20 00:00:00       1600        300         30
      7521 WARD       SALESMAN        7698 1981-02-22 00:00:00       1250        500         30
      7566 JONES      MANAGER         7839 1981-04-02 00:00:00       2975                    20
      7654 MARTIN     SALESMAN        7698 1981-09-28 00:00:00       1250       1400         30
      7698 BLAKE      MANAGER         7839 1981-05-01 00:00:00       2850                    30
      7782 CLARK      MANAGER         7839 1981-06-09 00:00:00       2450                    10
      7788 SCOTT      ANALYST         7566 1982-12-09 00:00:00       3000                    20
      7839 KING       PRESIDENT            1981-11-17 00:00:00       5000                    10
      7844 TURNER     SALESMAN        7698 1981-09-08 00:00:00       1500          0         30
```

```
      7876 ADAMS      CLERK      7788 1983-01-12 00:00:00      1100                 20
      7900 JAMES      CLERK      7698 1981-12-03 00:00:00       950                 30
      7902 FORD       ANALYST    7566 1981-12-03 00:00:00      3000                 20
      7934 MILLER     CLERK      7782 1982-01-23 00:00:00      1300       3333      10

14 rows selected.
```

1.3.5. Flashback transaction query

테이블의 데이터 변경 작업이 잘못 수행되었음을 나중에야 발견하는 경우가 있다. 변경 내역을 조사하기 위해, DBA 는 플래시백 쿼리를 실행하여 특정 시점의 로우 데이터를 조회할 수 있다. 또는 좀 더 효율적인 방법으로, Flashback Versions Query 기능을 이용하여 일정 기간 동안의 로우 변경 내역과 트랜잭션 ID 를 한꺼번에 확인할 수도 있다. 이때 DBA 는 SELECT 구문에 VERSIONS BETWEEN 절을 적용하고, SCN 또는 타임스탬프를 기준으로 일정 기간의 로우 변경 히스토리를 조회한다.

문제가 되는 트랜잭션을 발견했다면, 다시 Flashback Transaction Query 기능을 이용하여 해당 트랜잭션에 의해 수행된 다른 변경 사항을 확인한다. 그리고 변경 사항을 복구하기 위한 언두(undo) SQL 을 요청한다. 이때 트랜잭션 히스토리와 언두 SQL 을 확인하기 위해 사용되는 것이 바로 FLASHBACK_TRANSACTION_QUERY 뷰이다.

잘못 실행된 트랜잭션을 완전히 취소하기 위해, 언두 SQL 구문을 수동으로 실행하고 사용자/애플리케이션 에러를 쉽게 복구할 수 있다. Flashback Transaction Query 는 데이터베이스의 온라인 진단 범위를 확장하고, 분석 및 트랜잭션 감사 환경을 개선할 수 있게 한다.

△ VERSIONS_XID 값이 트랜잭션의 ID라고 했는데, 이 값을 FLASHBACK_TRANSACTION_QUERY의 인자 값으로 줘서 쿼리를 실행 하면 해당 트랜잭션에 대한 정보를 볼 수 있다.
  예를 들면 어떤 DML을 이용했으며 어떠한 SQL이 실행 되었는지 하는 것이 확인 가능하다.
△ Flashback transaction query는 Transaction level에서 Data의 변경사항을 추적하기 위한 기능
△ Transaction의 분석과 진단을 하는 기능 임.
△ 변경사항 뿐만 아니라, Undo SQL을 생성할 수 있으며, 이 SQL을 이용하여 Transaction level의 작업을 rollback할 수 있음
△ undo data를 index access 방식으로 조회하므로 logminor
  주의 : xid column에 조건을 줄 때 반드시 hextoraw function을 사용해야 만 fixed view의 index를 사용함.
△ Flashback versions query와 마찬가지로 undo data를 이용함.
△ Flashback Transaction query를 사용하기 위해서는 Database level에 logging이 enable되어야 한다.
  alter database add supplemental log data;
  확인방법 : select supplemental_log_data_min from v$database (YES 가정상)
△ 필요 권한 : grant select any transaction to XXX;
△ 기본적으로 flashback_transaction_query 라는 view table을 이용하여 query한다.
△ flashback_transaction_query columns.

XID	RAW(8)	Transaction identifier
START_SCN	NUMBER	Transaction start system change number (SCN)
START_TIMESTAMP	DATE	Transaction start timestamp
COMMIT_SCN	NUMBER	Transaction commit system change number (null for active transactions)
COMMIT_TIMESTAMP	DATE	Transaction commit timestamp (null for active transactions)
LOGON_USER	VARCHAR2(30)	Logon user for the transaction
UNDO_CHANGE#	NUMBER	Undo system change number (1 or higher)
OPERATION	VARCHAR2(32)	Forward-going DML operation performed by the transaction: D - Delete I - Insert U - Update B - UNKNOWN
TABLE_NAME	VARCHAR2(256)	Name of the table to which the DML applies
TABLE_OWNER	VARCHAR2(32)	Owner of the table to which the DML applies
ROW_ID	VARCHAR2(19)	Rowid of the row that was modified by the DML

• Database level에 logging이 enable되어있는지 확인 한다.

```
SQL> select supplemental_log_data_min from v$database;
```

```
SUPPLEME
--------
NO

SQL> alter database add supplemental log data;

Database altered.

SQL> select supplemental_log_data_min from v$database;

SUPPLEME
--------
YES
```

• emp와 dept를 각각 수정한 후, 이에 대한 transaction query를 하는 예제.

```
SQL> conn scott/tiger

SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE            SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------ ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80           800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81          1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81          1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81          2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81          1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81          2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81          2450                    10
      7788 SCOTT      ANALYST         7566 09-DEC-82          3000                    20
      7839 KING       PRESIDENT            17-NOV-81          5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81          1500          0         30
      7876 ADAMS      CLERK           7788 12-JAN-83          1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81           950                    30
      7902 FORD       ANALYST         7566 03-DEC-81          3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82          1300                    10

14 rows selected.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL>
SQL> update emp set sal = 9999, job='2030' where empno=7934;

1 row updated.

SQL> update dept set dname = 'ADMIN' where deptno = 40;

1 row updated.

SQL> commit;

Commit complete.
```

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE            SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ------------ ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80           800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81          1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81          1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81          2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81          1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81          2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81          2450                    10
      7788 SCOTT      ANALYST         7566 09-DEC-82          3000                    20
      7839 KING       PRESIDENT            17-NOV-81          5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81          1500          0         30
      7876 ADAMS      CLERK           7788 12-JAN-83          1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81           950                    30
      7902 FORD       ANALYST         7566 03-DEC-81          3000                    20
      7934 MILLER     2030            7782 23-JAN-82          9999                    10

14 rows selected.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 ADMIN          BOSTON

-- flashback versions query를 이용하여 xid를 찾는다.
SQL> col versions_starttime format a30
SQL> col versions_endtime format a30

SQL> select versions_starttime, versions_endtime, versions_xid, versions_operation, empno, sal
  2  from scott.emp versions between timestamp minvalue and maxvalue
  3  where empno = 7934
  4  order by VERSIONS_STARTTIME;

VERSIONS_STARTTIME             VERSIONS_ENDTIME               VERSIONS_XID     V     EMPNO        SAL
------------------------------ ------------------------------ ---------------- - ---------- ----------
25-APR-07 05.13.51 PM                                         000A002700000076B U       7934       9999
                               25-APR-07 05.13.51 PM                                    7934       1300

SQL> select versions_starttime, versions_endtime, versions_xid, versions_operation, deptno, dname
  2  from scott.dept versions between timestamp minvalue and maxvalue
  3  where deptno = 40
  4  order by VERSIONS_STARTTIME;

VERSIONS_STARTTIME             VERSIONS_ENDTIME               VERSIONS_XID     V     DEPTNO DNAME
------------------------------ ------------------------------ ---------------- - ---------- --------------
25-APR-07 05.13.51 PM                                         000A002700000076B U         40 ADMIN
                               25-APR-07 05.13.51 PM                                      40 OPERATIONS

SQL> col OPERATION format a12
SQL> col LOGON_USER format a12
SQL> col UNDO_SQL format a100

SQL> SELECT xid, operation,logon_user, undo_sql
```

```
  2  FROM flashback_transaction_query
  3  WHERE xid = HEXTORAW('000A00270000076B');            -- hextoraw를 사용하지 않으면,
                                                             undo tablespace의 크기에 따라 10분이상 걸림.

XID              OPERATION    LOGON_USER   UNDO_SQL
---------------- ------------ ------------ ------------------------------------------------------------------------------------------------------------
000A00270000076B UPDATE       SCOTT        update "SCOTT"."DEPT" set "DNAME" = 'OPERATIONS' where ROWID = 'AAAQLHAAHAAACfeAAD';
000A00270000076B UPDATE       SCOTT        update "SCOTT"."EMP" set "JOB" = 'CLERK', "SAL" = '1300' where ROWID = 'AAAQLGAAHAAACfWAAN';
000A00270000076B BEGIN        SCOTT

-- 해당 Transaction을 rollback하기 위해서는 아래와 같이 undo_sql을 수행한다.
SQL> update "SCOTT"."DEPT" set "DNAME" = 'OPERATIONS' where ROWID = 'AAAQLHAAHAAACfeAAD';
1 row updated.

SQL> update "SCOTT"."EMP" set "JOB" = 'CLERK', "SAL" = '1300' where ROWID = 'AAAQLGAAHAAACfWAAN';
1 row updated.

SQL> commit;

SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE           SAL       COMM     DEPTNO
---------- ---------- --------- ---------- ----------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80          800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81         1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81         1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81         2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81         1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81         2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81         2450                    10
      7788 SCOTT      ANALYST         7566 09-DEC-82         3000                    20
      7839 KING       PRESIDENT            17-NOV-81         5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81         1500          0         30
      7876 ADAMS      CLERK           7788 12-JAN-83         1100                    20
      7900 JAMES      CLERK           7698 03-DEC-81          950                    30
      7902 FORD       ANALYST         7566 03-DEC-81         3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82         1300                    10

14 rows selected.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

1.3.6. Flashback Table

Oracle9i Database에는 Flashback 질의 옵션 개념이 도입되어 데이타를 과거의 시점에서부터 검색하지만, 테이블 삭제 같은 DDL 작업을 순간적으로 되돌릴 수는 없습니다. 이 경우 유일한 수단은 다른 데이타베이스에서 테이블스페이스 적시 복구를 사용한 다음, 엑스포트/임포트 또는 기타 메서드를 사용해 현재 데이타베이스에 테이블을 다시 생성하는 것입니다. 이 프로시저를 수행하려면 복제를 위해 다른 데이타베이스를 사용하는 것은 물론, DBA의 많은 노력과 귀중한 시간이 요구됩니다.

하지만 Oracle Database 10g의 Flashback 테이블 기능으로 들어가면 몇 개의 문만 실행하여 삭제된 테이블을 간단히 검색할 수 있습니다.

△ Flashback Table은 잘못된 데이터 처리를 한 경우, 작업전의 시점으로 빠르게 돌려주기 위한기능. (SCN or

시간)
△ Flashback Table 명령을 통해 개별적인 테이블에 대해 시간에 준한 복구를 위해서는 아래에 있는 조건을 만족해야 합니다. 테이블의 데이터에 대해 과거 시점으로 돌아가서 값들을 확인 하는 것이 가능 합니다.
△ Flashback any table 또는 해당 Table에 대한 Flashback object privilege를 가지고 있어야 합니다.
△ 테이블에 대한 SELECT, INSERT, DELETE, ALTER 권한이 있어야 합니다.
△ ROW MOVEMENT의 경우 테이블에 대해 ALTER TABLE tablename ENABLE ROW MOVEMENT;가 설정 되어 있어야 합니다.

△ Backup의 restore없이 Table을 지정한 시점까지 되돌려 줌.
△ Table의 데이터만을 과거시점의 데이터로 돌려주며, Table과 관련한 모든 object (index, constrains, trigger)등은 현재시점으로 유지됨
△ Table이 Flashback 하는 동안에는 exclusive lock을 잡게됨.
△ Flashback 한후, 다시 현재 시점의 Data로 돌아올 수 있음. 그러나 현재의 SCN을 알고 있어야 함.
SELECT current_scn FROM v$database; -- 현재 SCN알기
△ 다음의 Object들에는 Flashback table 안됨.
Cluster, Mview, AQ tables, static data dictionary, system tables, remote tables
△ Undo Data를 이용함.
△ undo retention 이전의 데이터는 복구 안됨.
△ flashback versions query로부터 원하는 SCN을 찾아서 flashback table을 할 수 있음.
(VERSIONS_STARTSCN, VERSIONS_ENDSCN)
△ 필요 권한 : flashback object, flashback any table, 해당 table에 대한 select, insert, update, delete, alter table 권한.
△ flashback table을 하기 위해서는 row movement 를 enable해 주어야 함.
alter table XXXX enable row movement;
△Table에 DDL의 변경 작업이 있었다면, flashback 불가 (moving, truncate, add, modify, drop,merging, split, coalescing)

• **Flashback Table 예제: SCN을 이용한 과거시점으로 Flashback 하기**

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

  COUNT(*)
----------
        14

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
   25842839

SQL> delete from emp where rownum < 5;                    -- 잘못된 transaction을 수행함.

4 rows deleted.

SQL> commit;

Commit complete.

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
   25842867

SQL> alter table emp enable row movement;                 -- flashback table을 하기 위해 enable 시킴.

Table altered.

SQL> flashback table emp   to scn 25842839;
```

```
Flashback complete.

SQL> select count(*) from emp;                    ➔ Flashback후, delete전의 데이터가 됨.

  COUNT(*)
----------
        14
```

1.3.7. Flashback Use Case

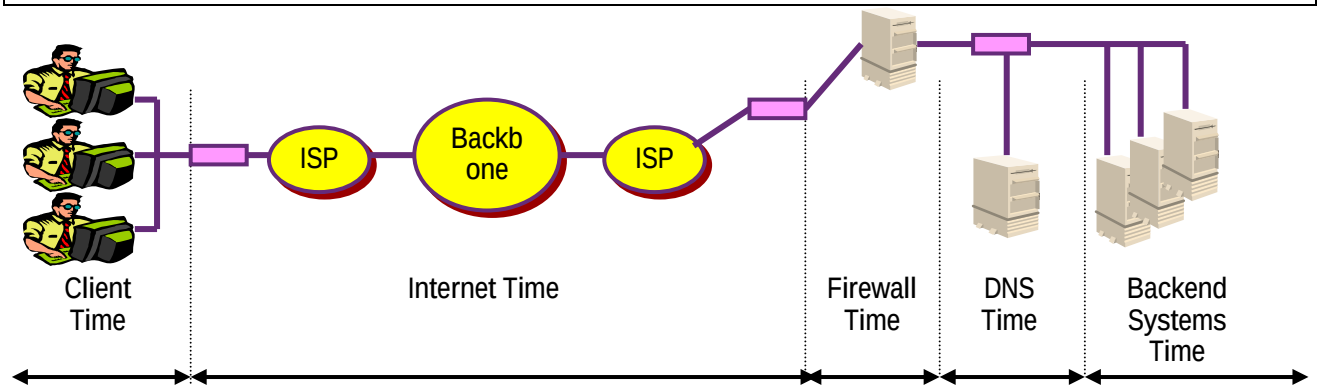
• 장애의경우에 따라 Use Case를사용하여 신속히 복구한다.

장애 Case	Case 상세	복구 방법
Table이 Drop된경우		Recyclebin을 조회하여 drop한 table의 복구가능성을 확인 한다. Flashback Drop을 이용하여 복구 한다.
Table에 데이터를 잘못 변경하고 commit한 경우	많은 데이터 변경시	변경시점으로 Table을 flashback하는 방법.
	적은 데이터 변경시	Table에 대해 Version query를 이용하여 해당data의 변경 tx를 찾는 방법.
Program이 잘못 수행되어 여러개의 table에 변경되었을 경우.	Commit이 한번일 경우	하나의 Table에서, 변경된 Data에 대한 Versions query를 하여 Transaction을 찾은 후, Transaction에 대한 undo를 뽑아 복구.
	Commit이 여러번인 경우	Flashback query를 통해 여러 Table을 Select하여 backup본 구성.
데이터에 대한 변경이력 추적시		Flashback Version Query를 이용하여 변경 이력 추적

Pro-Active Tuning Service 소개

1. 실제 사용자(End-User) 관점의 응답시간 튜닝

Pro-active tuning service 는 사용자 관점의 모니터링 및 분석을 통하여 실제 End-User 가 느끼는 응답시간(Response Time)을 튜닝합니다. APM(Application Performance Management) 툴을 이용하여 End-User 의 Request 결과를 반환받기까지의 모든 구간(Client PC, Internet 구간, FireWall, DNS, Web Server, WAS, DBMS) 을 분석하여 가장 Delay Time 이 많이 소요된 구간을 찾아 냅니다.



2. 최상의 성능 상태로 비즈니스 고가용성을 유지

Pro-Active Tuning Service

- 매 업무 단위 프로젝트 마다 참여하여 업무 적용(Open) 前 문제 요소를 분석하여 튜닝.
- 단위 업무 적용(Open) 후 매 3 개월(데이터량 갱신 주기) 마다 튜닝 포인트를 설정, 성능 둔화 요소를 해결.
- 전사적으로 새롭게 추가되는 업무 단위 프로젝트의 모든 SQL 쿼리를 검토 및 튜닝.
- 다양한 대용량 데이터베이스 관리/튜닝 기법을 도입하여 최적의 DB 상태를 1 년 내내 상시 유지.
- 전략적 튜닝 Factor 를 분석, 투자 대비 효율이 높은 Targeting 기법 적용. (비중도 높은 SQL 을 튜닝함)

[Pro-Active Tuning Service Schedule]

Performance Drop Point 마다 적절한 튜닝을 실시함으로써 항상 최적의 성능을 유지한다. !!

3. Knowledge Transfer

Pro-Active Tuning Service 는 고객의 Business Process 를 이해하고 시스템을 분석한 후 튜닝하는 것으로 완료되지 않습니다. 실제로 고객사 환경에서 튜닝한 내용을 그대로 실무자들에게 전수하여 내부 임직원의 역량을 제고시킵니다. 또한, Oracle RDBMS 신 버젼의 New Features 를 교육함으로써, 이용자(관리자 및 개발자)가 스스로 개발 업무의 효율 및 생산성을 향상시킬 수 있도록 지원합니다. 이외에도 DBMS 관리자를 위한 관리 노하우(고급 Trouble-Shooting, 대용량 DB 처리, 병렬 처리 등)를 전수함으로써, 최상의 시스템을 최고의 기술로 유지할 수 있도록 지원합니다.

UAS (User Adapted Seminar) 진행 사례 및 내용 (Contents)

- **개발자를 위한 SQL 튜닝 실무 사례 세미나**
  G 쇼핑몰 업체 튜닝 후 실제 고객사의 튜닝 사례를 개발자들에게 전수하여 개발자들이 성능을 고려한 SQL 을 작성할 수 있도록 내부 역량을 제고시킴.
- **Oracle 10g New Features 세미나**
  S Global 전자 기업 : Oracle 10g 버전으로 업그레이드 하기 전, 신 버전의 새로운 기능과 주의 사항을 전파함으로써, 업그레이드 후 발생할 수 있는 문제점의 사전 제거와 개발자들이 새로운 기능을 이용함으로써, 개발 생산성을 향상시킴.
- **K 국가기관 DBMS 관리 노하우 세미나**
  내부 관리자 (DBA,SE)들을 대상으로 DBMS 관리자들이 흔히 겪을 수 있는 상황에 대한 Administration Know-How 와 고급 Trouble-Shooting 사례를 소개함으로써, 관리 기술력을 향상시킴.

Goodus
Breathing Life into Technology