Flashback Table LAB

Querying the past state of the table is achieved using the `AS OF` clause of the `SELECT` statement. As the SCOTT user, login to SQL*PLUS and delete the employee with id number 50 from the STAFF table..Assume that time is at 9:31:00 on September 23, 201x Then commit the query.

1). sqlplus scott/tiger

SCOTT (orcl)> delete from staff where id = 50;

SCOTT (orcl)> commit;

a). Record the time at which you deleted this user.

For example, the following query retrieves the state of the employee record for id 50 at 9:30AM, September 23, 2014:

```
SELECT * FROM STAFF AS OF TIMESTAMP
   TO_TIMESTAMP('201x-09-23 09:30:00', 'YYYY-MM-DD HH:MI:SS')
   WHERE ID = 50;
```

Restoring EMPLOYEE WITH ID 50'S information to the table STAFF requires the following update:

```
INSERT INTO STAFF
    (SELECT * FROM STAFF AS OF TIMESTAMP
     TO_TIMESTAMP('201x-09-23 09:30:00', 'YYYY-MM-DD HH:MI:SS')
     WHERE id = 50);
```

The missing row is re-created with its previous contents, with minimal impact to the running database.

Let's take a look at another scenario. The problematic year-end batch run might have affected only a few tables. For example, it may have updated only the table `EMP` with new COMMISSIONS (COMM). If this is the case, a DBA can use the flashback table feature, which reinstates a table to a point in the past.

There is no special setup necessary to perform the flashback table operation. The only requirement is that the table must have row movement enabled—either at table creation time or later using the `ALTER TABLE ACCOUNTS ENABLE ROW MOVEMENT` statement. The `FLASHBACK TABLE` statement reads the past images of the table from the undo segments and reconstructs the table rows using the flashback queries introduced in Oracle9*i*.

If a non-DBA user other than the schema owner performs a flashback table operation, he/she needs `SELECT, DELETE, INSERT, ALTER`, and `FLASHBACK` privileges on that table or the equivalent `ANY TABLE` system privileges. First create a user called john with the same password and grant him connect, resource roles

```
SIDPERS (orcl2)> create user john identified by password;

SIDPERS (orcl) > grant connect, resource to john;
```

Run the demobld script to create tables as the john user.

```
sqlplus john/password

SQL>  @demobld
```

. We also need to allow for flashback table row movement. For the emp, staff, org and dept tables we will alter them to allow for the row movement;

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

SQL> alter table emp enable row movement;

SQL> alter table dept enable row movement;

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

For John, the table `EMP` looks like this:

```
Name                  Null?          Type
--------------------- ------------   ------------
EMPNO                 NOT NULL       NUMBER(4)
ENAME                                CHAR(10)
JOB                                  CHAR(9)
MGR                                  NUMBER(4)
HIREDATE                             DATE
SAL                                  NUMBER(7,2)
COMM                                 NUMBER(7,2)
DEPTNO                               NUMBER(2)
```

Here are the steps John takes to use the flashback table feature:

1. He asks for an approximate point in time to which the database has to be retraced; the answer is about 11:00 p.m.
2. He defines the desired logical reference point to go back to. Here's what he sees when he queries the table now:
3. `select ENAME,COMM`
4. `from EMP`
5. `WHERE EMPNO BETWEEN 7300 AND 7500;`
6. `     ENAME   COMM`
7. `     ------  --------`
8. `      SMITH`
9. `      ALLEN   300`
10. The output shows that COMM has not been processed for all accounts yet (everyone gets a commission. The desired logical reference point should be where all the employees have commissions. We will know update the employees by giving each a 500.00 commission as a part of the standard ITHACA COLLEGE bonus given monthly and placed into the commission column.

11. select to_char(sysdate,'YYYY-MM-DD HH:MI:SS') FROM DUAL (OR EMP) to get the current time.

12.
    ```
    update emp
    set comm = nvl(comm,0) + 500;
    ```

13. He issues this statement to restore the table `emp table` to that prior update time:
14. `flashback table ACCOUNTS to timestamp`
15. `to_timestamp (<time before update,'mm/dd/yyyy hh24:mi:ss');`

Voilà! The entire table is reconstructed as of that time stamp. John can flash back to a point as far in the past as the remaining undo data in undo segments allow. Instead of the time stamp, John can also use the system change number (SCN) as follows:

```
flashback table emp
to SCN 9988653338;
```

This method could employ finding a suitable SCN number far enough back to accomplish the above task. First we would query the V$FLASHBACK_DATABASE_LOG table for the oldest scn number available.

```
select oldest_flashback_scn, oldest_flashback_time
 from v$flashback_database_log;
OLDEST_FLASHBACK_SCN        OLDEST_FL
------------------------------------        ----------------
  2659752                                   16-MAR-05
```

Second we would find the current SCN # as follows

```
select current_scn from v$database;
CURRENT_SCN
----------------------
   2663240
```

John could then flash back the table to a particular time 10:30 p.m.—and then rechecks the status:

```
16.  flashback table emp to timestamp
17.  to_timestamp ('09/24/2014 22:30:00','mm/dd/yyyy
   hh24:mi:ss');
```

Since the table is never dropped, all the dependent objects—such as indexes, constraints, triggers, and so on—remain intact. All independent objects referencing this table, such as procedures, also remain valid. Even global indexes on partitioned tables are maintained and remain valid.

If John wanted to flash back the table `dept` in addition to the table `emp`, he could have used many table names separated by a comma as follows:

```
flashback table john.dept, john.emp to scn 1234567;
```

The entire flashback table operation is done through a single, powerful SQL statement.

Let's examine another situation. Suppose that Laura accidentally drops a key lookup table, scott.`ORG`. Realizing the mistake, she asks John if he can reinstate the table. In prior versions of Oracle Database, this would have required a point-in-time recovery. In Oracle Database 10*g*, however, dropping a table renames the table and places it in a logical container known as the Recycle Bin.

To recover the table, John simply issues the following command:

```
flashback table john.org to before drop;
```

The table reappears instantly, without needing any kind of recovery. Note that unlike the flashback operation described earlier, this does not require data reconstruction through the undo segments; rather the table is merely moved back from the recycle bin.