Journal of Information and Operations Management
ISSN: 0976–7754 & E-ISSN: 0976–7762 , Volume 3, Issue 1, 2012, pp-119-123
Available online at http://www.bioinfo.in/contents.php?id=55

DATA RECOVERY IN OODBMS USING FLASHBACK

SAHEBRAO N. SHINDE[1] AND CHITRA G. DESAI[2]

[1]Computer Sci Dept., K.T.H.M. College, Nashik, India.
[2]Department of MCA, MIT Aurangabad, India.
*Corresponding Author: Email-[1]sns110@gmail.com,[2]chitrag_desai@yahoo.com

Received: December 12, 2011; Accepted: January 15, 2012

Abstract- Serialization and Deserialization maintain the object state but cannot retain the object state at any given point of time as it will not store the time when the object was created and removed. Retaining the state of object at any given time will be too much complicated in OODBMS as compared to flashback in ORDBMS. Flashback takes you to previous point of time to correct errors caused by users. Flashback is a feature which provides point-in-time recovery without restoring the backup. It makes possible to correct user errors like undropping tables, delete records and commited. This paper demonstrates the concept of serialization and deserialization using java.

Citation: Sahebrao N. Shinde and Chitra G. Desai (2012) Data recovery in OODBMS using flashback. Journal of Information and Operations Management ISSN: 0976–7754 & E-ISSN: 0976–7762, Volume 3, Issue 1, pp-119-123.

Introduction

Serialization is a process of maintain object state in the form of stream. Deserialization is the process of creating object from stream[1].

In case of Serialization and Deserialization you can maintain the object state but you cannot retain the object state at any given point of time as it will not store the time when the object was created and removed[1]. If you store the time also you have to write the code for getting the object state up to a particular time. In other words we can say, retaining the state of object at any given time will be too much complicated in OODBMS as compared to flashback in ORDBMS[3].

Serializing and Deserializing Objects

The serialization mechanism in Java provides the means for persisting objects beyond a single run of a Java program. To serialize an object, make sure that the declaring class implements the java.io.Serializable interface. Then obtain an ObjectOutputStream to write the object to and call the writeObject() method on the ObjectOutputStream. To deserialize an object, obtain an ObjectInputStream to read the object from and call the readObject() method on the ObjectInputStream. The code excerpt in figure 1 illustrate how an object of type OOTest1 is serialized and deserialized.

The opposite operation of the serialization is called deserialization i.e. to extract the data from a series of bytes is s known as deserialization which is also called inflating or unmarshalling.

The given program shows how to read any data or contents from the serialized object or file. It takes a file name and then converts into java object. If any exception occurs during reading the serialized file, it is caught in the catch block.

ObjectInputStream extends java.io.InputStream and implements ObjectInput interface. It deserializes objects, arrays, and other values from an input stream. Thus the constructor of ObjectInputStream is written as:

ObjectInputStream obj = new ObjectInputStream(new FileInputStream(f));

The code in figure1 of the program creates the instance of the ObjectInputStream class to deserialize that file which had been serialized by the ObjectInputStream class. The code in figure1 creates the instance using the instance of the FileInputStream

class which holds the specified file object which has to be deserialized because the ObjectInputStream is written as:
ObjectInputStream obj = new ObjectInputStream(new FileInputStream(f));
The code in figure1 of the program creates the instance of the ObjectInputStream class to deserialize that file which had been serialized by the ObjectInputStream class. The code in figure1 creates the instance using the instance of the FileInputStream class which holds the specified file object which has to be deserialized because the ObjectInputStream() constructor needs the input stream.
readObject()::
Method readObject( ) reads the object and restore the state of the object. This is the method of the ObjectInputStream class that helps to traverse the object.

```
import java.io.*;
class Employee implements Serializable
  {
    int empno;
    String ename;
    double sal;
    Employee(int a, String b, double c)
    {
      this.empno=a;
      this.ename=b;
      this.sal=c;
     }
 }

class OOTest
 {
   public static void main(String args[]) throws Exception
    {
     Employee emp=new Employee(101,"Sunil",123.45);
     ObjectOutputStream o=new ObjectOutputStream(new FileOutputStream("emp.txt"));
     o.writeObject(emp);
     o.close();
    }
  }

import java.io.*;
class OOTest1
{
public static void main(String args[]) throws Exception
   {
     ObjectInputStream o=new ObjectInputStream(new FileInputStream("emp.txt"));
     Object obj=o.readObject();
     Employee emp= (Employee) obj;
     System.out.println(emp.empno+"\t"+emp.ename+"\t"+emp.sal);
     o.close();
```

Flashback
Flashback takes you to previous point of time to correct errors caused by users[4]. It is a feature which provides point-in-time recovery without restoring the backup [5]. Flashback makes it possible to correct user errors like undropping tables, delete records and commited. Flashback must be used when there is corruption or lost of data and if you want to recover data in a quick and easily or by mistake if any rows are deleted and commited or updated with a wrong value and issued a commit statement.
Flashback database brings the database to previous point of time. Flashback table recovers a table to a point-in-time without restoring files from a backup[3]
Flashback Database
With Flashback Database you can bring quickly and easily database to the earlier point in time because you do not need to restore backups [3]. You can flashback the database to undo the changes that have resulted in data loss or logical data corruption. If there is physical corruption then you have to use traditional recovery methods. Flashback database is faster than the tradition point-in-time recovery because the time to restore the database files takes long time[5]. With flashback database the time to recover the database is proportional to the number of changes made and not to the size of the database To demonstrate the flashback mechanism in oracle 10g and above, consider for example the following DEPT table schema.
CREATE TABLE DEPT(DEPTNO NUMBER(5) PRIMARY KEY, DNAME
VARCHAR(20), LOC VARCHAR(20));
The flashback database cannot be used if :
  i) The controlfile has been restored.
  ii) If the tablespace is dropped.
To enable the flashback feature follow the steps:
1) Database must be configured in archivelog mode.
2) Set the following parameters:
    db_recovery_file_dest_size = 1G
    db_recovery_file_dest = C:\ORCL
    db_flashback_retention_target = 900
Let us know consider the scenario that we have deleted the records of table DEPT and commited.
To get the records of table DEPT there are two ways:
With LOGMINER
Logmining is nothing but getting the records by extracting the archive files. This is a lengthy process and the precondition is that archive files must be present (it means database must be running in archive log mode)[4].
With FLASHBACK
With we can recover the data much faster. This is a short way of recovering data which was delete and commited and also you can get back the dropped table faster.
Up to Oracle 9i there is no guarantee that you can get the records back i.e. till there is sufficient space in the undo tablespace for other transactions you can get back but if there is not enough space for the other transaction then the commited area will be overwritten and you cannot get the records back[3]. In Oracle 10g the retention is guaranteed.
Following is the example of getting the records back after deleting and issuing a commit statement.

```
15:52:54 SQL> EXEC DBMS_FLASHBACK.ENABLE_AT_TIME (TO_DATE('28-JUN-
                2011 18:17:16','DD-MON-YYYY HH24:MI:SS'));
```

```
DECLARE
    CURSOR C1 IS SELECT * FROM DEPT;
    REC C1%ROWTYPE;
BEGIN
   OPEN C1;
   DBMS_FLASHBACK.DISABLE;
   LOOP
     FETCH C1 INTO REC;
     EXIT WHEN C1%NOTFOUND;
     INSERT INTO DEPT VALUES
(REC.DEPTNO,REC.DNAME,REC.LOC);
     COMMIT;
   END LOOP;
   CLOSE C1;
END;
/
```

Flashback table

Flashback table recovers tables to a specific point in time without restoring a backup. A non-owner of a table can flashback a table if he has FLASHBACK ANY TABLE privilege[6]. If you drop a table, the table is not dropped permanently; it will be in the recyclebin.

If you drop a table, now if you want to get it back there are two ways:

- With Point-in-Time recovery. (You have to restore last backup database) 500ms
- With flashback command:

Flashback table DEPT to before drop

Flashback a table on time basis:

```
SQL> flashback table emp to timestamp to_timestamp('20-jun-
2011 11:11:11','dd-mon-
        yyyy hh24:mi:ss');
```

It should be however remembered that a Flashback table operations cannot be performed on system tables, external tables, views, temporary tables.

All existing index and also dependent objects are maintained. Flashback table generates undo and redo data. If we want to drop the table permanently i.e., also from the recyclebin we need to give the following command:

purge table emp

If we want to drop all the tables which are in recyclebin then issue:

```
SQL> purge recycle bin;
```

If we want to drop all the tables belonging to the tablespace users:

```
SQL> purge tablespace users;
```

If you want to drop all the tables belonging to the user scott which are in tablespace users:

```
SQL>purge tablespace users user scott;
```

Flashback on time basis

Flashback on time basis recovers data on the basis of time after commit.The recovery of data on the basis of time after commit is by flashback on time basis.

Flashback query:

Queries the data at a specified point in time.

```
SQL> select job,sal from emp as of timestamp to_timestamp('20-
jun-2011 11:11:11');
```

or

```
update emp set sal=(select sal from emp as of timestamp
to_timestamp('20-jun-2011 11:11:11');
```

Flashback Version Query:

It displays all the versions of rows between two times. Flashback Version Query is used to retrieve the different versions of specific rows that existed during a given time interval. A new row version is created whenever a COMMIT statement is executed.

Intially collect the Timestamp using following command:

```
SQL> select to_char(SYSTIMESTAMP,'YYYY-MM-DD HH:MI:SS')
from dual;
            TO_CHAR(SYSTIMESTAMP,'YYYYY)
            ---------------------------
            2011-06-9 20:30:43
```

Suppose a user creates a table emp and inserts a row into it and commits the row.

```
SQL> Create table emp (empno number(5), name varchar2(20),
sal  number(10,2));
        SQL> insert into emp values (101,'S1', 4000);
        SQL>commit;
```

At this time emp table has one version of one row.

Now a user sitting at another machine erroneously changes the Salary from 4000 to 2000 using Update statement

```
SQL> update emp set sal=sal-3000 where empno=101;
       SQL> commit;
```

The following example performs a FLASHBACK TABLE operation the table emp

```
FLASHBACK TABLE emp TO TIMESTAMP
        TO_TIMESTAMP('2011-06-9  09:30:00',  `YYYY-MM-DD
HH24:MI:SS');
```

The following command will give you the status of table emp before 5 minutes.

```
flashback table emp to timestamp(systimestamp - 5 /1440);
select count(*) from emp as of timestamp(sysdate - 5 / 1440);
flashback table emp to timestamp(systimestamp - 5 /1440);
select count(*) from emp as of timestamp(sysdate - 5 / 1440);
```

Flashback Drop (Recycle Bin)

In Oracle 10g the default action of a DROP TABLE command is to move the table to the recycle bin, rather than actually dropping it. The PURGE option can be used to permanently drop a table [6].

The recycle bin is a logical collection of previously dropped objects, with access tied to the DROP privilege. The contents of the recycle bin can be shown using the SHOW RECYCLEBIN command and purged using the PURGE TABLE command. As a result, a previously dropped table can be recovered from the recycle bin:

```
CREATE TABLE flash_tab (id  NUMBER(10));
                    INSERT INTO flash_tab (id) VALUES (1);
                    COMMIT;
                     DROP TABLE flash_tab;
                     SHOW RECYCLEBIN
ORIGINAL NAME   RECYCLEBIN NAME  OBJECT TYPE  DROP
TIME
---------------- ------------------------------ ------------ -------------------
FLASHBACK_DROP_T    BIN$TstgCMiwQA66fl5FFDTBgA==$0
TABLE  24-jun-2011:11:09:07
FLASHBACK TABLE flash_tab TO BEFORE DROP;
SELECT * FROM flashback_drop_test;
```

```
        ID
        ----------
       1
```

Tables in the recycle bin can be queried like any other table:

```
DROP TABLE flashback_drop_test;
SHOW RECYCLEBIN
ORIGINAL NAME    RECYCLEBIN NAME    OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
FLASHBACK_DROP_T BIN$TDGqmJZKR8u+Hrc6PGD8kw==$0 TABLE        2004-03-29:11:18:39EST
SELECT * FROM "BIN$TDGqmJZKR8u+Hrc6PGD8kw==$0";

        ID
----------
         1
```

If an object is dropped and recreated multiple times all dropped versions will be kept in the recycle bin, subject to space. Where multiple versions are present it's best to reference the tables via the RECYCLEBIN_NAME. For any references to the ORIGINAL_NAME it is assumed the most recent object is drop version in the referenced question. During the flashback operation the table can be renamed like:

```
FLASHBACK TABLE flashback_drop_test TO BEFORE DROP
RENAME TO flashback_drop_test_old;
```

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;                 -- The whole recycle bin.
```

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.
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.
Partitioned index-organized tables
are not protected by the recycle bin.
The recycle bin does not preserve referential integrity.
Flashback Database
The FLASHBACK DATABASE command is a fast alternative to performing an incomplete recovery. In order to flashback the database you must have SYSDBA privilege and the flash recovery area must have been prepared in advance.
If the database is in NOARCHIVELOG it must be switched to ARCHIVELOG mode:

```
CONN sys/password AS SYSDBA
ALTER SYSTEM SET
        log_archive_dest_1='location=d:\oracle\oradata\DB10G\archive\' SCOPE=SPFILE;
ALTER SYSTEM SET log_archive_format='ARC%S_%R.%T' SCOPE=SPFILE;
ALTER SYSTEM SET log_archive_start=TRUE SCOPE=SPFILE;
SHUTDOWN IMMEDIATE
STARTUP MOUNT
ARCHIVE LOG START
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

Flashback must be enabled before any flashback operations are performed:

```
CONN sys/password AS SYSDBA
SHUTDOWN IMMEDIATE
STARTUP MOUNT EXCLUSIVE
ALTER DATABASE FLASHBACK ON;
ALTER DATABASE OPEN;
```

With flashback enabled the database can be switched back to a previous point in time or SCN without the need for a manual incomplete recovery.
In the following example a table is created, the database is then flashbacked to a time before the table was created:

```
-- Create a flash_tab table.
                CONN scott/tiger
CREATE TABLE flash_tab ( id  NUMBER(10) );
-- Flashback 5 minutes.
        CONN sys/password AS SYSDBA
        SHUTDOWN IMMEDIATE
        STARTUP MOUNT EXCLUSIVE
        FLASHBACK DATABASE TO TIMESTAMP SYSDATE-(SYSDATE - 1/1440 * 5);
        ALTER DATABASE OPEN RESETLOGS;
-- Check that the table is gone.
        CONN scott/tiger
        DESC flash_tab
```

Some other variations of the flashback database command include:

```
FLASHBACK DATABASE TO TIMESTAMP d1(to_timestamp (date_time);
FLASHBACK DATABASE TO BEFORE TIMESTAMP d1;
FLASHBACK DATABASE TO SCN scnno;
FLASHBACK DATABASE TO BEFORE SCN scnno;
```

Conclusion

From above scenario flashback is helpful in recovering data up to a particular time due to small mistake of the user. Previous to the flashback feature if a table is dropped then the only way was to restore the backup and perform point-in-time recovery. It was wasting time to restore the backup due to a small user mistake. If you delete or update records due to mistake wrongly then there was only one option available was recovering data from logminer. Logminer is a tedious process because you have to start logmining activity and apply the archive files and then recover the data. But with flashback any user mistake can be rectified within notime by just executing simple SQL commands.

References

[1] Won Kim (1990) *Introduction to Object-Oriented Databases, The MIT press*.
[2] Kavin Loney (1990) *Tata McGraw-Hill publication second edition*.
[3] Donna Keesling (1999) *Oracle 9i Administration Fundamental Student Guide* vol 1.
[4] Kevin Loney, George Koch (2000) *Oracle 8i the complete reference: the most definitive resource*.
[5] Thomas Kyte (2003) *Expert-One-on-OracleApress*.
[6] Sam R. Alapati (2005) *Expert Oracle Database 10g Administration A press.*