OCP 认证培训教材

FlashBack

腾科 ORACLE 教学部

Oracle Flashback 闪回

【学习目标】

Flashback Database 功能非常类似与RMAN的不完全恢复，它可以把整个数据库回退到过去的某个时点的状态，这个功能依赖于Flashback log日志。比RMAN 更快速和高效。因此Flashback Database可以看作是不完全恢复的替代技术。

本章主要内容是描述 FLASHBACK 的使用。

【本章要点】

- 开启、配置闪回
- 使用闪回恢复数据

【关键术语】

Flashback 闪回	RecycleBin 回收站
Timestamp 时间戳	System Change Number 系统更改号

Oracle 的闪回技术是一种数据恢复技术，具有恢复时间快，不使用备份文件的特点，使得数据库可以回到过去的某个状态，可以满足用户逻辑错误的快速恢复，通常用于快速恢复数据库中出现的认为误操作等逻辑错误。可以诊断错误在数据库中是如何造成的，然后可以修复损坏。需要注意，闪回技术仅仅对逻辑恢复有效，如果是数据文件损坏必须使用介质恢复（备份还原）。

在没有闪回技术之前，数据库的逻辑错误恢复都是采用基于时间点的恢复，通过备份恢复数据库到过去指定的时间点，这种恢复方式需要使用备份并使用适当的归档日志完成，恢复时间取决于备份文件的复制时间和日志的应用时间，如果是很大的数据库可能会恢复很长时间，采用闪回技术，则可以更快速，便捷的恢复用户错误或数据库逻辑错误。

Flashback 分为以下成员： Flashback Database，Flashback Drop，Flashback Query(分 Flashback Query,Flashback Version Query，Flashback Transaction Query 三种) Flashback Table 和 Flashback Data Archive。

理解闪回级别：

从闪回的方式可以分为基于数据库级别闪回、表级别闪回、事务级别闪回

数据库级闪回：允许将整个数据库恢复到过去的某个时间点。数据库级的恢复在以下几种情况下使用。当误删除一个用户或者误截断一个表时可以采用数据库级的闪回恢复。

表级闪回：表级闪回可以将闪回到过去的某个时间点或恢复到过去的某个 SCN，而闪回删除通过 DROP 指令删除的表。

事务级闪回：该级闪回显示了用户事务的每个 DML 操作，并给出闪回的 DML 指令，比如 DELETE 操作对应的就是 INSERT 操作，通过对一个事务的闪回，可以恢复用户误操作的事务。

根据闪回对数据的影响程度又可以分为闪回恢复，闪回查询。闪回恢复将修改数据，闪回点之后的数据将全部丢失。而闪回查询则可以查询数据被 DML 的不同版本，也可以在此基础之上确定是否进行恢复。

一、 Flashback Version Query

所谓版本指的是每次事务所引起的数据行的变化情况，每一次变化就是一个版本。Oracle 提供了闪回版本查询，让我们可以看到数据行的整个变化过程。变化指的是已经提交的事务引起的变化，没有提交的事务引起的变化不会显示，闪回版本查询使用的是 undo 表空间里记录的 undo 数据，当 UNDO 段的数据由于空间压力而被清除，则产生无法闪回的情况。

Flashback Version 多用于查看某条特定记录所有已提交的版本，包括每个版本的创建时间以及结束时间。Flashback Version Query 可以看到过去某个时间段内，记录是如何发生变化的。根据这个历史，DBA 就可以快速的判断数据是在什么时点发生了错误，进而恢复到之前的状态。

通过 versions between 能够查看指定时间段内 undo 表空间中记录的不同版本（注意，只包括被提交的记录）。只需要在标准查询后面附加 versions between timestamp[/scn] t1 and t2 即可。记录在版本查询中可能会是一对多的关系，比如某些记录如果被修改过多次，并分别提交，那么你在查询的时候，如果修改的操作是在你指定的时间段(或 scn)，则记录每次修改的结果都会被选择出来，这比较有利于我们做数据的对比，比如看看数据究竟是怎么变化的，主要应用场景：审计某个时间做了什么，即审计查询。

闪回版本查询语法，使用 VERSIONS BETWEEN 关键字

```
基于 SCN 的版本查询
SELECT <columns>
FROM <schema_name.table_name>
VERSIONS BETWEEN SCN <minimum_scn> AND <maximum_scn>
[WHERE <column_filter>]
[GROUP BY <non-aggregated_columns>]
[HAVING <group filter>
[ORDER BY <position_numbers_or_column_names>]
基于 TIMESTAMP 的版本查询
SELECT <columns>
FROM <schema_name.table_name>
VERSIONS BETWEEN timestamp to_timestamp('start_timestamp') and to_timestamp('end_timestamp')
[WHERE <column_filter>]
[GROUP BY <non-aggregated_columns>]
[HAVING <group filter>
[ORDER BY <position_numbers_or_column_names>]
```

查询过程中提供了多个伪列如下：

VERSIONS_STARTSCN VERSIONS_STARTTIME
该记录操作时的 scn 或时间，如果为空，表示该行记录是在查询范围外创建的。

VERSIONS_ENDSCN VERSIONS_ENDTIME
该记录失效时的 scn 或时间，如果为空，说明记录当前时间在当前表内存在，或者已经被删除了，可以配合着 VERSIONS_OPERATION 列来看，如果 VERSIONS_OPERATION 列值为 D，说明该列已被删除，如果该列为空，则说明记录在这段时间无操作。

VERSIONS_OPERATION
对该行执行的操作：I 表示 insert，D 表示 delete，U 表示 update。提示：对于索引键的 update 操作，版本查询可能会将其识别成两个操作：DELETE 和 INSERT。

VERSIONS_XID 该操作的事务 ID

例子：

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
SQL> alter database add supplemental log data;
SYS@orcl> create user test1 identified by test1;
SYS@orcl> grant connect,resource to test1;
SYS@orcl> grant select on scott.emp to test1;
SYS@orcl> conn test1/test1;
TEST1@orcl> create table tb1 tablespace users as select empno,ename,job,deptno
from scott.emp;
TEST1@orcl> insert into tb1 values(1000,'Jack','Clerk',20);
TEST1@orcl> commit;
TEST1@orcl> update tb1 set job='Manager' where empno=1000;
TEST1@orcl> commit;
TEST1@orcl> delete from tb1 where empno=1000;
TEST1@orcl> commit;
TEST1@orcl> insert into tb1 values(1000,'Jack','President',20);
TEST1@orcl> commit;
SQL> update tb1 set job='ww' where empno=1000;
SQL> commit;
```

对 empno=1000 所作的不同的修改被全部记录下来，可以通过使用 versions between 关键字来查询对该表中的某条特定记录修改的不同版本。

```
TEST1@orcl> select empno,ename,job,versions_xid xid,versions_startscn
v_stcn,versions_endscn v_edcn,versions_operation v_ops from tb1 versions
between scn minvalue and maxvalue where empno=1000;
SQL> conn / as sysdba
SQL> select commit_scn,commit_timestamp,operation,undo_sql from
flashback_transaction_query where  xid in(select versions_xid from test1.tb1
versions between scn minvalue and maxvalue where empno=1000);
SQL> conn / as sysdba
SQL> select dbms_flashback.get_system_change_number from dual;
GET_SYSTEM_CHANGE_NUMBER
------------------------
         1051730
SQL> delete from test1.tb1 where empno=1000;
SQL> commit;
SQL> select dbms_flashback.get_system_change_number from dual;
GET_SYSTEM_CHANGE_NUMBER
------------------------
         1051754
SQL> select empno,ename,job,versions_xid xid,versions_startscn
v_stcn,versions_endscn v_edcn,versions_operation v_ops from test1.tb1 versions
between scn 1051730 and 1051754;
SQL> Select commit_scn,commit_timestamp,operation,undo_sql from
flashback_transaction_query where  xid in(select versions_xid from test1.tb1
versions between scn 1051730 and 1051754 where empno=1000);
SQL> select table_name,table_owner,undo_sql from flashback_transaction_query
where xid=hextoraw('04000D00D0020000');
SQL>select to_char(sysdate,'yyyy-mm-dd hh:mi:ss') from dual;
SQL> update test1.tb1 set ename='wl' where empno=7934;
SQL> commit;
SQL>select to_char(sysdate,'yyyy-mm-dd hh:mi:ss') from dual;
SQL> set linesize 300
SQL> col versions_starttime format a22
SQL> col versions_endtime format a22
SQL> select
versions_starttime,versions_startscn,versions_endtime,versions_endscn,versions_
```

```
xid,versions_operation,empno,ename from test1.tb1 versions between timestamp
minvalue and maxvalue;
SQL> select
versions_starttime,versions_startscn,versions_endtime,versions_endscn,versions_
xid,versions_operation,empno,ename from test1.tb1 versions between timestamp
to_date('2013-09-09 20:50:25','YYYY-MM-DD HH24:MI:SS') and to_date('2013-09-09
20:51:16', 'YYYY-MM-DD HH24:MI:SS');
```

二、Flashback Transaction Query

闪回事务查询就是对过去某段时间内所完成事务的查询和撤销，通过闪回事物分析，可以识别在一个特定的时间段内所发生的所有变化，也可以对数据库表进行事物级恢复。前面提到可以审计一个事务到底做了什么，现在可以获得事务的历史操作进行撤销。Flashback Transaction Query 利用 UNDO 表空间的 undo 数据来实现。利用这个功能可以查看某个事务执行的所有变化。

闪回事务查询是对闪回版本查询的扩展。从某种程度上来说，闪回版本查询通常用于更细粒度的查询，如针对特定的记录。而闪回事务则是针对某一事务进行闪回，是基于事务级别的。闪回事务查询通过查询视图 flashback_transaction_query 来获得某个或多个特定事务信息，同时可以根据该视图中提供的 undo_sql 列中的语句来反转事务，从而保证数据的完整性，这个视图的 XID 列代表事务 ID，利用这个 ID 可以区分特定事务发生的所有数据变化。在 DBMS__FLASHBACK 包上授予适当权限，将select any transaction权限授予将要使用闪回事务查询的用户，默认情况下 sys 用户和 DBA 角色具有该权限。

```
grant execute on dbms_flashback to scott;
grant select any transaction to scott;
```

使用闪回事务查询前，必须启用重做日志流的其它日志记录。重做日志流数据是撤消表空间中记录的信息的补充。闪回事务查询既需要增强的重做信息，也需要撤消信息。

使用 altet database 命令启用对 DML 更改引用的列值和主键值的日志记录：

```
alter database add supplemental log data;
alter database add supplemental log data (primary key) columns;
```

例如

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
Oracle 11g 禁用了 supplemental logging,开启后恢复正常：
SQL> alter database add supplemental log data;
SYS@orcl> create user test1 identified by test1;
SYS@orcl> grant connect,resource to test1;
```

SYS@orcl> grant select on scott.emp to test1;

SYS@orcl> conn test1/test1;

TEST1@orcl> create table tb2 tablespace users as select empno,ename,sal,deptno from scott.emp;

TEST1@orcl> insert into tb2 select 9999,'wl',3000,50 from dual;

TEST1@orcl> commit;

TEST1@orcl> select * from tb2 where empno=9999;

TEST1@orcl> update tb2 set sal=sal+500 where empno=9999;

TEST1@orcl> commit;

TEST1@orcl> update tb2 set deptno=20 where empno=9999;

TEST1@orcl> commit;

TEST1@orcl> select empno,ename,sal,deptno,versions_xid,versions_operation from tb2 versions between scn minvalue and maxvalue where empno=9999;

根据事务号获得一个反转该事务的 DML 语句

TEST1@orcl> conn / as sysdba

SYS@orcl>desc flashback_transaction_query

SYS@orcl> select operation,undo_sql from flashback_transaction_query where xid=hextoraw('07000C00EC000000');

从查询中得到反转事务的 DML 语句，直接执行相应的反转语句，即可将事务变更到特定的状态。

SYS@orcl> update "TEST1"."TB2" set "SAL" = '3000' where ROWID = 'AAAM0cAAEAAAAGHAAA';

SYS@orcl> select * from test1.tb2;

SYS@orcl> commit;

SQL> select table_name,operation,row_id,undo_sql from flashback_transaction_query where table_name='TB2';

或者

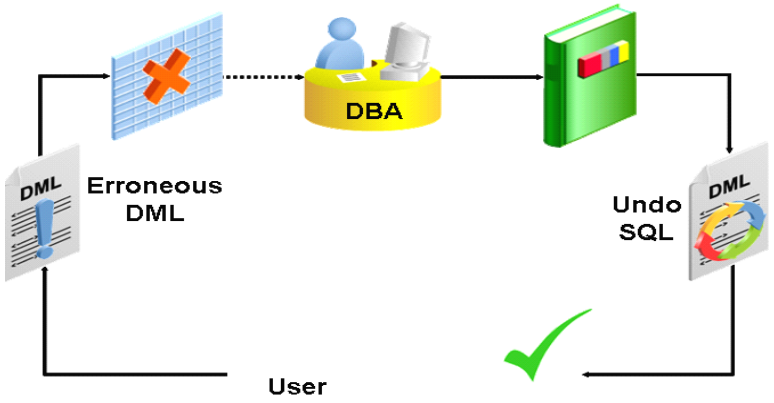
SQL> select operation,undo_sql from flashback_transaction_query where xid in(select versions_xid from test1.tb2 versions between scn minvalue and maxvalue);

运行上面找出的 sql 语句，即可将以前删除的数据恢复回来。

三、 Flashback query

闪回查询是查询该表过去某个时刻的数据情况，依赖于 undo 表空间的 undo 数据，一旦确认某个时刻的数据满足我们的需求以后，可以根据这个时间执行闪回表，闪回查询先查询，等确认后再闪回到需求时刻。

利用 oracle 查询多版本一致的特点，实现从回滚段中读取表一定时间内操作过的数据，可用来进行数据比对，或者修正意外提交造成的错误数据，该项特性也被称为 Flashback Query。



Flashback Query 利用多版本读一致性的特性从 UNDO 表空间读取操作前的记录数据！什么是多版本读一致性

Oracle 采用了一种非常优秀的设计，通过 undo 数据来确保写不堵塞读，简单的讲，不同的事务在写数据时，会将数据的前映像写入 undo 表空间，这样如果同时有其它事务查询该表数据，则可以通过 undo 表空间中数据的前映像来构造所需的完整记录集，而不需要等待写入的事务提交或回滚。

flashback query 有多种方式构建查询记录集，记录集的选择范围可以基于时间或基于 scn，要通过 flashback query 查询 undo 中的撤销数据，只需要在标准查询语句的表名后面跟上 as of timestamp(基于时间)或 as of scn(基于 scn)即可。

一、闪回查询(Flashback Query)语法

```
使用 as of scn
SELECT <column_name_list>
FROM <table_name>
AS OF <SCN>
[WHERE <filter_conditions>]
[GROUP BY <unaggregated columns>]
[HAVING <group_filter>]
[ORDER BY <column_positions_or_name>]
使用 as of timestamp
SELECT <column_name_list>
FROM <table_name>
AS OF <TIMESTAMP>
[WHERE <filter_conditions>]
```

```
[GROUP BY <unaggregated columns>]
[HAVING <group_filter>]
[ORDER BY <column_positions_or_name>]
```

1. As of timestamp

```
SQL> alter session set nls_date_format='YYYY-MM-DD hh24:mi:ss';
```

会话已更改。

```
SQL> select sysdate from dual;
```

SYSDATE

2009-10-15 19:04:16

```
SQL> select * from A;
```

模拟用户误操作，删除数据

```
SQL> delete from A;
```

已删除 4 行。

```
SQL> commit;
```

提交完成。

```
SQL> select * from A;
```

未选定行

查看删除之前的状态：

假设当前距离删除数据已经有 5 分钟左右的话：

```
SQL> select * from A as of timestamp sysdate-5/1440;
```

或者：

```
SQL>select * from A as of timestamp to_timestamp('2009-10-15 19:04:16','YYYY-MM-DD
hh24:mi:ss');
```

用 Flashback Query 恢复之前的数据：

```
SQL>insert into A select * from A as of timestamp to_timestamp('2009-10-15
19:04:16','YYYY-MM-DD hh24:mi:ss');
```

已创建 4 行。

```
SQL> COMMIT;
```

提交完成。

```
SQL> select * from A;
```

上例所表示的 as of timestamp 的确非常易用，但是在某些情况下，我们建议使用 as of scn 的方式执行 flashback query，比如需要对多个相互有主外键约束的表进行恢复时，

如果使用 as of timestamp 的方式，可能会由于时间点不统一的缘故,造成数据选择或插入失败，通过 scn 方式则能够确保记录的约束一致性。

2. As of scn

查看 SCN:

```
SQL>SELECT dbms_flashback.get_system_change_number FROM dual;
```

或

```
SQL> SELECT CURRENT_SCN FROM V$DATABASE;
CURRENT_SCN
-----------
1095782
```

删除数据：

```
SQL> delete from A;
```

已删除 4 行。

```
SQL> commit;
```

提交完成。

查看删除之前的状态：

```
SQL> select * from A as of scn 1095782;
```

用 Flashback Query 恢复之前的数据：

```
SQL> insert into A select * from A as of scn 1095782;
```

已创建 4 行。

```
SQL> commit;
```

提交完成。

```
SQL> select * from A;
```

二、举例

1、as of timestamp 闪回查询

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
SQL> create table test as select * from scott.dept;
SQL> select * from test;
查询当前的时间
SQL>select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;
SQL> delete from test;
SQL> commit;
SQL> select * from test;
SQL> select * from test as of timestamp to_timestamp('2012-10-27
09:29:18','yyyy-mm-dd hh24:mi:ss');
或
SQL> select * from test as of timestamp sysdate-3/1440;
或
SQL> select * from test as of timestamp(to_date('2012-10-27 09:29:18','yyyy-mm-
dd hh24:mi:ss'));
SQL> insert into test select * from test as of timestamp to_timestamp('2012-10-
27 09:29:18','YYYY-MM-DD hh24:mi:ss');
SQL> select * from test;
```

2、as of scn 闪回查询

```
SQL> SELECT CURRENT_SCN FROM V$DATABASE;
SQL> delete from test;
SQL> commit;
SQL> select * from test as of scn 497953;
SQL> insert into test select * from test as of scn 497953;
```

3、伪列 ORA_ROWSCN

所谓伪列，就是假的，不存在的数据列，用户创建表时虽然没有指定，但是 Oracle 为了维护而添加的一些内部字段。最熟悉的伪列就是 ROWID。ORA_ROWSCN 是 Oracle 10g 新增的，暂且把它看作是记录最后一次被修改时的 SCN。Flashback Version Query 就是通过这个伪列来跟踪出记录的变化历史。

举个例子：

```
SQL> create table A(id int);
SQL> select * from A;
SQL> insert into A values(5);
已创建 1 行。
SQL> select * from A;
SQL> commit;
提交完成。
SQL> insert into A values(6);
SQL> commit;
SQL> select ora_rowscn,id from A;
ORA_ROWSCN      ID
---------- ----------
   1055198       5
   1055198       6
```

获取更多的历史信息

```
SQL>select
versions_xid,versions_startscn,versions_endscn,DECODE(versions_operation,'I','I
nsert','U','Update','D','Delete','Original') "Operation",id from A versions
between scn minvalue and maxvalue;
```

或者

```
SQL> alter database add supplemental log data;
SQL> select current_scn from v$database;
```

```
CURRENT_SCN
-----------
    1054967
SQL> update a set id=8;
SQL> commit;
SQL> select current_scn from v$database;
CURRENT_SCN
-----------
    1054976
SQL>select xid,commit_scn,commit_timestamp,operation,undo_sql from
flashback_transaction_query q where q.xid in(select versions_xid from A
versions between scn 1054967 and 1054976);
```

Flashback Version Query 技术其实有很多伪列，但是 ORA_ROWSCN 是最重要。它记录的是最后一次被修改时的 SCN，注意是被提交的修改。如果没有提交，这个伪列不会发生变化。ORA_ROWSCN 缺省是数据块级别的，也就是一个数据块内的所有记录都是一个 ORA_ROWSCN，数据块内任意一条记录被修改，这个数据库块内的所有记录的 ORA_ROWSCN 都会同时改变。上例的查询结果可以证明。

可以在建表时使用关键字 rowdependencies，可以改变这种缺省行为，使用这个关键字后，每条记录都有自己的 ORA_ROWSCN。

举例：

```
SQL> create table B (id number(2)) rowdependencies;
表已创建。
SQL> insert into B values(1);
已创建 1 行。
SQL> insert into B values(2);
已创建 1 行。
SQL> insert into B values(3);
已创建 1 行。
SQL> commit;
提交完成。
SQL> select ora_rowscn, id from B;
ORA_ROWSCN ID
---------- ----------
1100560 1
```

```
1100560 2
1100560 3
```

此处 SCN 一样，一定很奇怪，这正好说明是最后一次被修改时的 SCN，如果没有提交，是不会变的，我们重做一下就清楚了。

```
SQL> analyze table B compute statistics;
表已分析。
SQL> select ora_rowscn, id from B;
ORA_ROWSCN ID
---------- ----------
1100560 1
1100560 2
1100560 3
SQL> delete from B;
已删除 4 行。
SQL> select ora_rowscn, id from B;
未选定行
SQL> insert into B values(1);
已创建 1 行。
SQL> commit;
提交完成。
SQL> insert into B values(2);
已创建 1 行。
SQL> commit;
提交完成。
SQL> select ora_rowscn, id from B;
ORA_ROWSCN ID
---------- ----------
1100723 1
1100729 2
```

四、 闪回表 Flashback Table

闪回表就是将表里的数据回退到历史上的某个时间点，例如回退到用户误删除数据之前的时间点，从而将误删除的数据恢复回来。在这个过程中，数据库仍然可用。闪回表利用 undo 表空间里记录的数据旧映像，如果闪回表所需要的 undo 数据由于保留的时间超过了初始化参数 undo_retention 所指定的值，从而导致该 undo 数据块被其它事务覆盖，就

不能恢复到指定的时间点。参数 undo_retention 的作用是 undo 中的数据至少保留的时间，在这段时间内不能覆盖，这样就可以保证将表恢复到这个时间段内的某个时间点。可以指定 undo 表空间的 retention guarantee 选项，来保证闪回的成功。

闪回表的操作会修改表里的数据，从而可能引起数据行的移动，比如某一行数据当前在 A 数据块里，而在表闪回到以前的某个时间点上时，在那个时间点上该行数据位于 B 数据块里面，因此在闪回表之前必须启用数据行的移动特性。

闪回表是一种能够恢复表或设置表到过去某个特定的时间点而又不需要进行不完全恢复的闪回技术。使用闪回表时，所有的相关对象都能得到恢复。

Oracle11g flashback table 特性：

1、在线操作；

2、恢复到指定时间点或 SCN 的任何数据；

3、自动恢复相关属性、如索引、触发器等；

4、满足分布式的一致性；

5、满足数据一致性，所有相关对象将自动一致；

6、闪回表技术是基于回滚数据(undo data)来实现的，因此，要想闪回表到过去的某个时间上，必须确保与回滚表空间有关的参数设置合理。

表闪回的几种方式

```
基于 SCN 的表闪回
FLASHBACK TABLE <schema_name.table_name> TO SCN <scn_number> [<ENABLE | DISABLE> TRIGGERS]
基于 TIMESTAMP 的表闪回
FLASHBACK TABLE <schema_name.table_name> TO TIMESTAMP <timestamp> [<ENABLE | DISABLE> TRIGGERS]
基于 RESTORE POINT 的表闪回
FLASHBACK TABLE <schema_name.table_name> TO RESTORE POINT <restore_point> [<ENABLE | DISABLE>
TRIGGERS]
```

注意：对于表闪回，可以多次使用同一类型的闪回方式，可以往前闪回，一旦往前闪回之后，也可以往后进行闪回。SYS 模式中的表不能使用表闪回技术

举例：

```
SQL>delete from test where id=7766;
SQL>commit;
```

进行恢复

```
SQL>alter table test enable row movement;
SQL>flashback table test to timestamp to_timestamp( '2007-10-10
15:33:30' ,' yyyy-mm-dd hh24:mi:ss' )
```

局限性

FLASHBACK TABLE 命令作为单独一个事务执行，获取 DML 锁，统计信息不会闪回；当前索引及依赖对象被维护；闪回表有如下特性：

1） 不能对系统表做闪回操作，SYS 用户不支持闪回；

2） 在执行 DDL 操作后不能做闪回操作；

3） 闪回操作命令写入 alert 日志文件；

4） 闪回操作会产生 undo 和 redo 数据；

Flashback Table 也是使用 UNDO tablespace 的内容来实现对数据的回退。注意：如果想要对表进行 flashback，必须允许表的 row movement.

```
Alter table table_name row movement;
```

要查看某表是否启用 row movement，可以到 user_tables、all_tables、dba_tables 查询。

```
SQL> select row_movement from user_tables where table_name='C';
ROW_MOVE
--------
ENABLED
```

要启用或禁止某表 row movement，可以通过下列语句：

```
--启用
SQL> ALTER TABLE table_name ENABLE ROW MOVEMENT;
--禁止
SQL> ALTER TABLE table_name DISABLE ROW MOVEMENT;
```

举例：

```
SQL> create table C (id number(2));
表已创建。
SQL> insert into C values(1);
已创建 1 行。
SQL> insert into C values(2);
已创建 1 行。
SQL> commit;
提交完成。
SQL> select * from c;
SQL> alter session set nls_date_format="yyyy-mm-dd hh24:mi:ss";
会话已更改。
SQL> select sysdate from dual;
SYSDATE
-------------------
2009-10-15 21:17:47
```

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

CURRENT_SCN
-----------
1103864
```

删除数据并恢复

```
SQL> delete from C;
已删除 2 行。
SQL> commit;
提交完成。
SQL> alter table c enable row movement;
表已更改。
SQL> flashback table c to scn 1103864;
闪回完成。
```

或者：

```
SQL> flashback table c to timestamp to_timestamp('2009-10-15 21:17:47','yyyy-
mmdd
hh24:mi:ss');
SQL> select * from c;
```

Flashback table 命令支持同时操作多个表，表名中间以逗号分隔即可，如果你执行一条 flashback table 命令时同时指定了多个表，要记住单个 flashback table 是在同一个事务中，因此这些表的恢复操作要么都成功，要么都失败。如：

```
flashback table a,b ,c to scn 1103864;
```

一些注意事项

1、基于 undo 的表恢复，需要注意 DDL 操作的影响，修改并提交过数据之后，对表做 DDL 操作，包括：drop/modify 列， move 表， drop 分区(如果有的话),truncate table/partition，这些操作会另 undo 表空间中的撤销数据失效，对于执行过这些操作的表应用 flashback query 会触发 ORA-01466 错误。

另外一些表结构修改语句虽然并不会影响到 undo 表空间中的撤销记录，但有可能因表结构修改导致 undo 中重做记录无法应用的情况，比如对于增加了约束，而 flashback query 查询出的 undo 记录已经不符合新建的约束条件，这个时候直接恢复显然不可能成功，你要么暂时 disable 约束，对恢复的数据进行处理之后，再执行恢复。

flashback query 对 v$tables,x$tables 等动态性能视图无效，不过对于 dba_*,all_*,user_*等数据字典是有效的。

2、基于 undo 的表恢复，flashback table 实际上做的也是 dml 操作(会在被操作的表上加
dml 锁)，因此还需要注意 triggers 的影响，默认情况下，flashback table to scn/timestamp 在执行时会自动 disable 掉与其操作表相关的 triggers，如果你希望在此期间 trigger 能够继续发挥做用，可以在 flashback table 后附加 ENABLE TRIGGERS 子句。

补充

什么是 Automatic Undo Management(自动撤销管理表空间)提到自动撤销管理表空间，就不得不提手动管理的回滚段。在 9i 之前，回滚段的管理和监控是需要 dba 手工介入的，创建合适的回滚段是件非常耗费 dba 精力的事情，你可能需要不断关注 oracle 运行状况很长一阵子时间后，通过不断的调整才能基本确认一段时期内回滚段的大小，一旦回滚段创建的不合适，就极有可能引起性能问题甚至错误。

9i 之后呢，oracle 为了清晰它的整个概念，取消了回滚段这个说法(实际上并未取消回滚段)，而完全以 undo 来代替，这也正好与 redo 相对应，一个撤销，一个重做。回滚段可以不再由 dba 手工介入，完全由它自己在运行时自动分配，这在一定程度上即解放了 dba，也确实起到了提高性能的作用，比如采用自动管理表空间就可以最大程序的降低 ora-1555 发生的机率(注意是降低，不是避免)是否起用自动管理的撤销表空间由二个初始化参数决定：UNDO_MANAGEMENT：值为 AUTO 表示使用了自动撤销管理表空间，MANUAL 则表示手动管理，UNDO_TABLESPACE：当 UNDO_MANAGEMENT 值为 AUTO 时，该参数用来指定当前的 undo 表空间名称。

undo 表空间的大小，直接影响到 flashback query 的查询能力，因为多版本查询所依赖的 undo 数据都存储在 undo 表空间中，该表空间越大，所能够存储的 undo 数据自然也越多，如果该表空间可用空间非常小，别说 flashback 了，恐怕正常查询都有可能出错。

初始化参数 UNDO_RETENTION，该参数用来指定 undo 记录保存的最长时间，以秒为单位，是个动态参数，完全可以在实例运行时随时修改，通常默认是 900 秒，也就是 15 分钟。一定要注意，undo_retention 只是指定 undo 数据的过期时间，并不是说 undo 中的数据一定会在 undo 表空间中保存 15 分钟，比如说刚一个新事务开始的时候，如果 undo 表空间已经被写满，则新事务的数据会自动覆盖已提交事务的数据，而不管这些数据是否已过期，因此，当你创建一个自动管理的 undo 表空间时，还要注意其空间大小，要尽可能保证 undo 表空间有足够的存储空间。

同时还要注意，并不是说，undo_retention 中指定的时间一过，已经提交事务中的数据就立刻无法访问，它只是失效，只要不被别的事务覆盖，它仍然会存在，并可随时被 flashback 特性引用。如果你的 undo 表空间足够大，而数据库又不是那么繁忙，那么其实

undo_retention 参数的值并不会影响到你，哪怕你设置成 1，只要没有事务去覆盖 undo 数据，它就会持续有效。因此，这里还是那句话，要注意 undo 表空间的大小，保证其有足够的存储空间。

只有在一种情况下，undo 表空间能够确保 undo 中的数据在 undo_retention 指定时间过期前一定有效，就是为 undo 表空间指定 retention guarantee，指定之后，oracle 对于 undo 表空间中未过期的 undo 数据不会覆盖。

```
SQL> Alter tablespace undotbs1 retention guarantee;
```

如果想禁止 undo 表空间 retention guarantee，

```
SQL> Alter tablespace undotbs1 retention noguarantee;
```

举例

1、TIMESTAMP 的表闪回

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
SQL>  create user test1 identified by test1;
SQL> grant connect,resource to test1;
SQL> conn test1/test1
SQL> create table t1 as select * from all_objects;
SQL> create table t2 as select * from t1;
SQL> select count(*) from t1;
SQL> select count(*) from t2;
SQL> create index inx_test1 on T1 (object_name);
SQL> create index inx_test2 on T1 (object_id);
SQL> select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;
SQL> drop index inx_test1;
SQL> delete from t1;
SQL> commit;
SQL> truncate table t2;
```

查看 row_movement 状态

```
SQL> select table_name,row_movement from user_tables where table_name='T1';
TABLE_NAME                     ROW_MOVE
------------------------------ --------
T1                             DISABLED
SQL> select table_name,row_movement from user_tables where table_name='T2';
TABLE_NAME                     ROW_MOVE
------------------------------ --------
```

T2 DISABLED

SQL> ALTER TABLE t1 ENABLE ROW MOVEMENT;

SQL> ALTER TABLE t2 ENABLE ROW MOVEMENT;

SQL> flashback table t1 TO TIMESTAMP to_timestamp('2013-11-27 05:15:08','yyyy-mm-dd hh24:mi:ss');

SQL> flashback table t2 TO TIMESTAMP to_timestamp('2013-11-27 05:15:08','yyyy-mm-dd hh24:mi:ss');

出错

ERROR at line 1:

ORA-01466: unable to read data - table definition has changed

执行 delete 操作的表是可以恢复，执行 truncate 操作的表是不可以恢复，flashback table 利用 undo。

验证

SQL> select count(*) from t1;

SQL> select count(*) from t2;

SQL> select t.index_name from user_indexes t where t.table_name='T1';

索引“inx_test1”无法恢复，因为 drop 不记录 undo，所以 drop 的索引无法恢复。

2、SCN 的表闪回

[oracle@opc ~]$ uniread sqlplus / as sysdba

SQL> create user test1 identified by test1;

SQL> grant connect,resource to test1;

SQL> conn test1/test1

SQL> create table test2 as select owner from all_objects where 1=0;

无插入数据前的 SCN：1055072

SQL> conn / as sysdba

SQL> select current_scn,systimestamp from v$database;

SQL> insert into test1.test2 values(1);

SQL> commit;

第一次插入数据的 SCN：1055141

SQL> select current_scn,systimestamp from v$database;

SQL> insert into test1.test2 values(2);

SQL> commit;

第二次插入数据的 SCN：1055633

SQL> select current_scn,systimestamp from v$database;

```
SQL> insert into test1.test2 values(3);
SQL> commit;
```

第三次插入数据的 SCN：1055648

```
SQL> select current_scn,systimestamp from v$database;
SQL> alter table test1.test2 enable row movement;
```

验证

```
SQL> flashback table test1.test2 to scn  497109;
SQL> select * from test1.test2;
SQL> flashback table test1.test2 to scn  497165;
```

3、RESTORE POINT 的表闪回

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
SQL> create table scott.tb_emp enable row movement as select
empno,ename,job,deptno from scott.emp where 1=0;
SQL> create restore point zero;
SQL> insert into scott.tb_emp select empno,ename,job,deptno from scott.emp
where deptno=10;
SQL> commit;
SQL> create restore point one;
SQL> insert into scott.tb_emp select empno,ename,job,deptno from scott.emp
where deptno=20;
SQL> commit;
SQL>  create restore point two;
SQL> insert into scott.tb_emp select empno,ename,job,deptno from scott.emp
where deptno=30;
SQL> commit;
SQL> select deptno,count(*) from scott.tb_emp group by deptno order by 1;
SQL> flashback table scott.tb_emp to restore point two;
SQL> drop restore point one;
```

存在参照关系的表闪回

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
SQL> create user test1 identified by test1;
SQL> grant connect,resource to test1;
SQL> grant select on scott.emp to test1;
SQL> grant select on scott.dept to test1;
```

```
SQL> conn test1/test1
create table tb1 enable row movement as select * from scott.emp ;
create table tb2 enable row movement as select * from scott.dept;
alter table tb1 add constraint tb1_empno_pk primary key(empno);
alter table tb2 add constraint tb2_deptno_pk primary key(deptno);
alter table tb1 add constraint tb1_tb2_deptno_fk foreign key(deptno) references
tb2(deptno);
insert into tb2 select 50,'Customer','Landon' from dual;
insert into tb1(empno,ename,job,deptno) select 8000,'W1','Clerk',50 from dual;
commit;
conn / as sysdba
select current_scn from v$database;    497543
conn test1/test1
delete from tb1 where empno=8000;
delete from tb2 where deptno=50;
commit;
flashback table tb1 to scn 497543;    //收到错误
flashback table tb1,tb2 to scn 497543;
```

雇员被闪回，部门也被闪回

```
select empno,ename,deptno,dname from tb1 inner join tb2  using(deptno) where
deptno=50;
```

一个事务执行必须获得足够的 undo 空间来存储事务产生的 undo 信息，如果事务不提交，则 undo 信息被保留，如果事务提交了，则 undo 信息不一定会保留，因为 undo 循环使用。

例：一定能闪回查询 1 天之内发生的变化，即要求 undo 保留 1 天。需启动 guaranteeing undo retention 强制保留属性。

```
SQL> show parameter undo_retention
```

900 秒=15 分钟

```
SQL> alter system set undo_retention=86400;
```

强制保留 undo 类型表空间

```
SQL> desc dba_TABLESPACEs
SQL> select tablespace_name,retention from dba_tablespaces where
contents='UNDO';
```

设置强制保留，undo 信息在 undo 表空间至少存储 1 天，不到 1 天，是不能被覆盖的，所以一定可以完成 1 天闪回查询

```
SQL> alter tablespace undotbs1 retention guarantee;
```

如果设置强制的时候，undo 表空间不够呢?事务如果拿不到足够的空间，又不能去重用，则事物会失败。

例子

```
SQL>create undo tablespace undo02 datafile
'/u01/app/oracle/oradata/orcl/undo02.dbf' size 5M;
SQL>alter system set undo_tablespace=undo02;
SQL>alter tablespace undo02 retention guarantee;
SQL>select tablespace_name,retention from dba_tablespaces where contents='UNDO';
SQL>show parameter undo
SQL>conn scott/scott
SQL>create table big as select * from all_objects;
SQL>delete from big;
```

提示错误：ORA-30036: unable to extend segment by 8 in undo tablespace 'UNDO02'

分批删除：

```
SQL>delete from big where rownum<=500;
```

失败--》如果拿不到足够的空间，事务无法执行

```
SQL>delete from big where rownum<=100;
```

成功删除

还原 undo

```
SQL>alter system set undo_tablespace=undotbs1;
SQL>delete from scott.big;
```

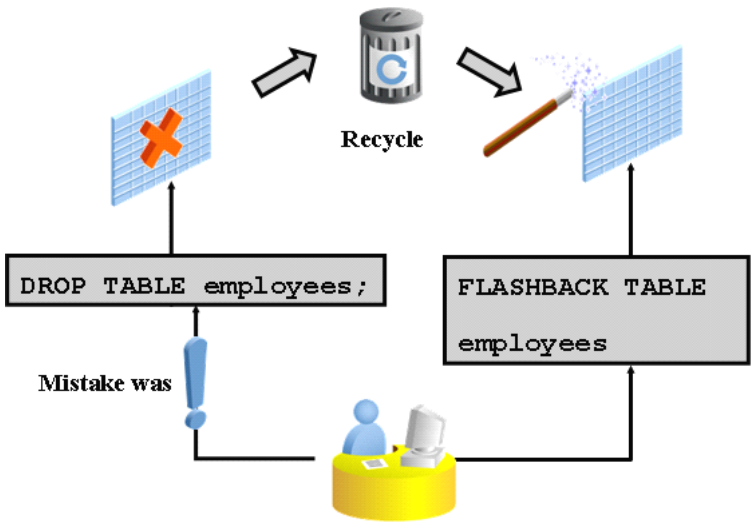
成功删除

五、 闪回删除（Flashback Drop）

闪回删除的目的是防止用户错误地删除表、索引等数据库对象，在没有闪回技术前，只能使用传统的数据恢复方式从备份中恢复，此时需要备份文件以及归档日志文件，这样的恢复往往涉及那些不需要恢复的对象，显然这样的恢复不具有针对性，而使用闪回删除就可以直接恢复想恢复的对象。

如果使用 drop table 删除表，该表不会从数据库中立即删除，而是保持原表的位置，但是将删除的表重新命名，并将删除的表信息存储在回收站中，回收站记录了被删除表的新名字和原名字，显然此时被删除的表所占用的空间没有被立即释放，变成数据库可以使用的潜在空间，记录在回收站中的信息会保留一段时间，直到回收站空间不足或使用 purge 指令删除回收站中的记录。

回收站是一个逻辑结构，不具有物理数据结构，只要删除的表信息记录在回收站中就可以通过闪回技术恢复删除的表。每个表空间都有一个叫作回收站的逻辑区域，Oracle 回收站将用户所做的 DROP 语句操作记录在一个系统表里，即将被删除的对象写到一个数据字典中，当不再需要被删除对象时，可以使用 PURGE 命令对回收站空间进行清除。闪回删除是指将被删除的数据库对象及其相应对象的拷贝保存在回收站中，以便在必要时能够及时恢复这些对象。在回收站被清空以前，被删除的对象并没有从数据库中删除，这就使得数据库能够恢复被意外或误操作而删除的表。



如删除 emp 表，然后该表对应的物理数据块空间就成为备用的可用空间，该删除记录保存记录在回收站中，回收站为删除的表重新命名并记录该表的原始名，这样用户就很容易查询删除的表以及对应的原始表名。

FLASHBACK DROP 允许在不丢失任何数据的情况下将指定的表恢复至被删除的时间点，并保持数据库为当前状态。闪回删除并不是真正的删除表，而是把该表重命名并放入回收站，类似于 Windows 的回收站一样。当某个活动对象需要使用该表所占用的空间时，该表才会被真正删除。只要空间未被复用，该表即可恢复。

将先前删除的表恢复到删除之前的状态，恢复该表的索引以及触发器，授权；恢复该表的约束，包括唯一约束、主键约束、非空约束，外键约束不可恢复。可以实现基于系统和基

于会话的 flash drop 操作，drop table 命令并不真正删除表,在内部被映射为 rename 命令,即是将其重命名之后放入回收站。

```
alter system set recyclebin = on | off;
alter session set recyclebin = on | off;
```

举例说明：

SQL>drop table test;

查询 user_recyclebin 回收站

SQL>select object_name,original_name from user_recyclebin;

利用回收站中的记录，使用 flashback 恢复表 test

SQL>flashback table test to before drop;

回收站空间清除，可以使用 PURGE 命令清除回收站对象从而释放空间,有以下几种方式：

1、使用 PURGE TABLE original_table_name; 这里的 original_table_name 表示未 drop 以前的名称

2、使用 PURGE TABLE recyclebin_object_name; 这里的 recyclebin_object_name 表示回收站中的对象名称

3、使用 PURGE TABLESPACE tablespace_name 从指定的表空间中清除所有的丢弃对象

4、使用 PURGE TABLESPACE tablespace_name USER user_name 从回收站中清除属于某个特定用户的所有丢弃对象。

5、DROP USER user_name cascade 直接删除指定用户及其所属的全部对象，也就是说，DROP USER 命令会绕过回收站进行直接删除。

6、使用 PURGE RECYCLEBIN 命令清除用户自己的回收站

7、PURGE DBA_RECYCLEBIN 从所有用户的回收站清除所有对象

SQL> conn / as sysdba

SQL> PURGE DBA_RECYCLEBIN;

局限性，闪回丢弃对如下表不生效：

存放在 SYSTEM 表空间上的表；

建在字典管理表空间上的表；

已经被手工清除或自动清除的表（即表被 DROP 后使用了 PURGE 操作）

如下依赖对象不受保护：

位图索引

物化视图日志

外键一致性约束

初始化参数 recyclebin 控制是否启用 recyclebin 功能，缺省是 ON，可以使用 OFF 关闭。

```
SQL> show parameter recycle
```

禁用该功能：

```
SQL> alter system set recyclebin=off;
SQL> alter session set recyclebin=off;
```

禁用后删除的对象将直接删除，不会写到 Recycle 中。如在删除时，指定 purge 参数，表也将直接删除，不会写到 recyclebin 中。

```
SQL> drop table name purge;
```

查看 recyclebin 中的对象列表：

```
SQL> select * from A;
SQL> drop table A;
```

表已删除。

```
SQL> show recyclebin
ORIGINAL NAME RECYCLEBIN NAME OBJECT TYPE DROP TIME
--------------- ------------------------------ ------------ -------------------
A BIN$RWXQQcTPRde0ws4h9ewJcg==$0 TABLE 2009-10-15:12:44:33
```

查看 recyclebin 中对象：

```
SQL> select original_name,object_name from recyclebin;
ORIGINAL_NAME OBJECT_NAME
-------------------------------- ------------------------------
A BIN$RWXQQcTPRde0ws4h9ewJcg==$0
```

查看 recyblebin 对象里的内容：

```
SQL> select * from "BIN$RWXQQcTPRde0ws4h9ewJcg==$0";
```

表空间的 Recycle Bin 区域只是一个逻辑区域，而不是从表空间上物理的划出一块区域固定用于回收站，因此 Recycle Bin 和普通对象共用表空间的存储区域，或者说是 Recycle Bin 的对象要和普通对象抢夺存储空间。当发生空间不够时，Oracle 会按照先入先出的顺序覆盖 Recycle Bin 中的对象。也可以手动的删除 Recycle Bin 占用的空间。

```
SQL> select original_name,object_name from recyclebin;
ORIGINAL_NAME OBJECT_NAME
-------------------------------- ------------------------------
A BIN$RWXQQcTPRde0ws4h9ewJcg==$0
SQL> flashback table a to before drop;
```

闪回完成。

SQL> select * from a;

当我们删除表 A 后，在新建表 A，这时在恢复的时候就会报错，此时我们在闪回时，对表重命名就可以了：

SQL> drop table a;

表已删除。

SQL> create table a (id number(1));

表已创建。

SQL> flashback table a to before drop ;

flashback table a to before drop

*

第 1 行出现错误:

ORA-38312: 原始名称已被现有对象使用

SQL> flashback table a to before drop rename to B;

闪回完成。

SQL> select * from B;

当我们删除表 A，在新建表 A，在删除它，这是在 Recycle Bin 中就会有 2 个相同的表明，

此时恢复我们就要指定 object_name 才行.

SQL> select * from B;

SQL> drop table B;

表已删除。

SQL> create table B(name varchar(20));

表已创建。

SQL> drop table B;

表已删除。

SQL> select original_name,object_name from recyclebin;

ORIGINAL_NAME OBJECT_NAME

-------------------------------- ------------------------------

B BIN$vYuv+g9fTi2exYP9X2048Q==$0

B BIN$geQ9+NekSjuRvzG+TqDVWw==$0

SQL> flashback table "BIN$vYuv+g9fTi2exYP9X2048Q==$0" to before drop;

闪回完成。

SQL> select * from B;

一旦完成闪回恢复，Recycle Bin 中的对象就消失了。Flashback Drop 需要注意的地方：

1)只能用于非系统表空间和本地管理的表空间

2)对象的参考约束不会被恢复，指向该对象的外键约束需要重建。

3)对象能否恢复成功，取决与对象空间是否被覆盖重用。

4)当删除表时，信赖于该表的物化视图也会同时删除，但是由于物化视图并不会被放入 recycle bin，因此当你执行 flashback table to before drop 时，也不能恢复依赖其的物化视图，需要 dba 手工介入重新创建。

5)对于 Recycle Bin 中的对象，只支持查询.

例子

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
SYS@orcl> conn scott/scott
SCOTT@orcl> create table tb_emp as select * from emp;
SCOTT@orcl> alter table tb_emp add constraint empno_pk primary key(empno);
SCOTT@orcl>  alter table tb_emp add constraint ename_uk  unique(ename);
SCOTT@orcl> alter table tb_emp add constraint sal_ck check(sal>0);
SCOTT@orcl> alter table tb_emp modify job constraint job_nn not null;
SCOTT@orcl> alter table tb_emp add constraint dept_fk foreign key(deptno)
references dept(deptno) on delete cascade;
SCOTT@orcl> select constraint_name,constraint_type from user_constraints where
table_name='TB_EMP';
SYS@orcl> conn / as sysdba
```

表 tb_emp 所在文件的 id,块的起始 id,大小

```
SYS@orcl> select file_id,block_id,bytes from dba_extents where
segment_name='TB_EMP';
```

查看表 tb_emp 的对象 ID

```
SYS@orcl> select object_name,object_id from dba_objects  where object_name =
'TB_EMP';
```

对表进行重命名

```
SCOTT@orcl> alter table tb_emp rename to tb_employees;
```

重命名后所在文件的 id,块的起始 id,大小没有发生变化

```
SYS@orcl> select file_id,block_id,bytes from dba_extents where
segment_name='TB_EMPLOYEES';
```

重命名后对象 ID 没有发生变化

```
SYS@orcl> select object_name,object_id from dba_objects where object_name =
'TB_EMPLOYEES';
```

重命名后索引和约束也没有发生变化

```
SCOTT@orcl> select index_name,index_type from user_indexes where
table_name='TB_EMPLOYEES' union all select constraint_name,constraint_type from
user_constraints where table_name='TB_EMPLOYEES';
```

从上例得出，对于表的重命名仅仅是修改了表名，对于表对象的 ID，以及表存放的位置，块的起始，大小等并未发生变化。

1、删除表 tb_employees 并查看回收站的信息

```
SCOTT@orcl> drop table tb_employees;
SCOTT@orcl> select object_name,original_name,can_undrop,base_object from
user_recyclebin;
```

使用回收站名来访问对象，但要对对象加双引号

```
SCOTT@orcl> select count(1) from "BIN$7DA24BRU64PgQAB/AQAelw==$0";
```

2、闪回并查看闪回后的情况

```
SCOTT@orcl> flashback table tb_employees to before drop;
SCOTT@orcl> select count(1) from tb_employees;
```

查看闪回后索引，约束的情况，发现名称仍然为 BIN$名称，由系统生成的名字，可以将其改回，但外键约束已经不存在。

```
SCOTT@orcl> select index_name,index_type from user_indexes where
table_name='TB_EMPLOYEES' union all  select constraint_name,constraint_type
from user_constraints where table_name='TB_EMPLOYEES';
```

对表进行 DML 操作

```
SCOTT@orcl> insert into tb_employees(empno,ename,job,sal,deptno) select
9999,'W1','DBA',3000,50 from dual;
```

将 BIN 开头的索引改回原来的名字

```
SCOTT@orcl> alter index "BIN$7DA24BRS64PgQAB/AQAelw==$0" rename to EMPNO_PK;
```

回收站的管理，每个用户都拥有自己的回收站，且可以查看在自己模式中删除的表，使用 purge 命令可以永久删除对象。purge 命令的几种常用形式：

```
drop table tbname purge     --直接删除表，而不置于回收站
purge table tbname          --清除回收站中的 tbname 表
purge index idx_name        --清除回收站中的索引 idx_name
```

purge tablespace tablespace_name --清除该表空间中所有已删除的对象

purge tablespace tablespace_name user user_name --清除表空间中指定用户删除的对象

purge user_recyclebin --清除指定用户已删除的所有对象

purge dba_recyclebin --清除所有已删除的对象

回收站的视图

dba_recyclebin

user_recyclebin

表空间不足时无法闪回表删除

SYS@orcl> create tablespace test datafile

'/u01/app/oracle/oradata/orcl/test.dbf' size 1m;

表空间可用空间

SYS@orcl> select tablespace_name,sum(bytes/1024/1024) ||' M' from

dba_free_space where tablespace_name='TEST' group by tablespace_name;

表空间不能自动扩展

SYS@orcl> select tablespace_name,autoextensible from dba_data_files where

tablespace_name ='TEST';

SYS@orcl> create table tb1 tablespace test as select * from dba_objects where

rownum < 6000;

SYS@orcl> select tablespace_name,sum(bytes/1024/1024) ||' M' from

dba_free_space where tablespace_name='TEST' group by tablespace_name;

SYS@orcl> drop table tb1;

SYS@orcl> show recyclebin;

空间显示的可用空间已返还为 1M ,但并不是真正为 1M,在需要表空间时,将自动清除回收站最老的对象,以满足当前空间需求

SYS@orcl> select tablespace_name,sum(bytes/1024/1024) ||' M' from

dba_free_space where tablespace_name='TEST' group by tablespace_name;

SYS@orcl> create table tb2 tablespace test as select * from dba_objects where

rownum < 6000;

回收站中原来的表 tb1 记录被自动清除

SYS@orcl> show recyclebin;

无内容显示

表 tb1 不能被闪回

```
SYS@orcl> flashback table tb1 to before drop;
```

报错：

```
ERROR at line 1:
ORA-38305: object not in RECYCLE BIN
```

通过对上述表的删除及空间分配情况，总结如下：

1、表的删除被映射为将表的重命名，然后将其置于回收站。

2、表的索引,触发器,授权,闪回后将不受到影响。索引、触发器名字可以更改回原来名称。

3、对于约束，如果是外键约束，表删除之后将不可恢复，其余的约束不受影响。

4、查询回收站中的对象，对象名使用双引号括起来。

5、对于表空间不足时，系统会自动清除回收站中最老的对象，从而导致闪回失败

6、闪回表的常用方法

```
flashback table tbname to before drop ;
flashback table tbname to before drop rename to newtbname;
```

第二条语句用于被删除的表名已经被再次重用,故闪回之前必须将其改名为新表名,schema 不变化

7、如回收站中存在两个相同的原表名，则闪回时总是闪回最近的版本，如果闪回特定的表，需要指定该表在回收站中的名称。如

```
flashback table "BIN$k1zC3yEiwZvgQAB/AQBRVw==$0" to before drop;
```

8、flashback drop 不能闪回 truncate 命令截断的表，只能恢复 drop 之后的表。

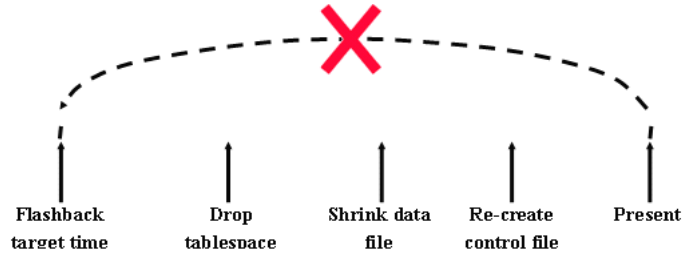
9、flashback drop 不能闪回 drop user scott cascade 删除方案的操作,此只能用 flashback database

10、在 system 表空间中存储的表无法启用 flashback drop，且这些表会被立即删除。

六、 Flashback Database

闪回数据库技术是一种快速的数据库恢复方案，这种恢复是基于用户的逻辑错误，比如对表中的数据做了错误的修改，插入了大量错误数据，删除了一个用户等，此时往往需要将数据库恢复到修改之前的某个时间点。如果是传统的恢复技术，则首先需要复原备份的数据文件，再使用归档日志恢复到以前的某个时间点，显然这样的恢复涉及很多不必要的数据恢复，并且恢复的时间主要取决于库的大小。Oracle 的闪回数据库技术解决了这个问题，它使用闪回日志来恢复用户的逻辑错误，这种恢复只针对用户逻辑错误的恢复，而不涉及整个数据库的恢复，恢复更具有针对性，且恢复时间大大减少。

闪回日志由oracle 自动创建，并存储在闪回恢复区中，由闪回恢复区管理。既然闪回日志由



闪回恢复区管理，那么就不能保证闪回日志被保存的数据的可靠性，因为一旦闪回恢复区空间不足，会自动删除旧的闪回日志文件以腾出空间。在 oracle 中闪回恢复区中备份文件的存储优先，所以，为了保存尽可能多的闪回日志数据，最好将闪回恢复区设置的大些。

闪回缓冲区（flashback buffer）中的变化数据将按照一定的时间间隔，顺序的被写入闪回日志（flashback logs）,比如用户对表删除或增加了数据，此时变化的前像操作就记录在闪回日志中，一旦闪回，将利用相反的操作恢复修改，注意将闪回缓冲区中的数据写入闪回日志的操作由 rvwr 进程负责，一旦启动了闪回数据库，该进程会自动启动。

```
SQL> SELECT name FROM v$bgprocess WHERE paddr!='00';
[oracle@oracle Desktop]$ ps -ef|grep rvwr
```

如果数据库中绝大多数是查询操作，此时的闪回数据库开销很小，因为它不需要做什么，所以闪回数据库的系统开销取决于是否有密集型的写操作。对于闪回数据库而言，闪回日志不会被归档。

flashback database 闪回到过去的某一时刻，使用 resetlogs 创建新的场景并打开数据库。flashback database 用于 truncate table、多表发生意外错误等，使用闪回日志来实现数据库闪回，闪回点之后的数据将丢失。

Oracle11g 闪回数据库特点：

1、使用闪回数据库不能解决媒介故障；

2、如果控制文件已被重建，不能使用闪回数据库；

3、不能完成删除表空间的恢复。

闪回数据库局限性：如下情形不能进行闪回数据库操作：

1、控制文件被恢复或重新创建

2、表空间被 drop 掉

3、数据文件被压缩

flashback database 的组成

闪回缓冲区:当启用 flashback database,则 sga 中会开辟一块新区域作为闪回缓冲区,大小由系统分配，启用新的 rvwr 进程，rvwr 进程将闪回缓冲区的内容写入到闪回日志中，注意闪回日志不同于联机重做日志，闪回日志在联机重做日志基础之上生成，是完整数据块映像的日志。联机日志则是变化的日志。闪回日志不能复用，也不能归档。闪回日志使用循环写方式。

flashback database 的配置

flashback database 要求数据库必须处于归档模式，且闪回之后必须使用 resetlogs 打开数据库。Oracle 默认不启动闪回数据库，如果需要启动闪回数据库特性必须将数据库设置为归档模式，并启用闪回恢复区，因为闪回日志文件存放在闪回恢复区中。

a. 查看数据库的归档模式及闪回是否启用（FLASHBACK_ON 为 NO，表示闪回特性尚未启用）

```
SQL> select log_mode,open_mode,flashback_on from v$database;
```

当前的数据库处于归档模式，使用 db_recovery_file_dest 参数指定的目录作为储存目录，该参数值即为快速恢复区。

b. 查看及设置闪回目录、闪回目录空间大小等

使用 alter system set db_recovery_file_dest 来设置新路径

使用 alter system set db_recovery_file_dest_size 来设定新的大小

```
SQL> show parameter db_recovery
```

参数 db_flashback_retention_target 是一个以 分钟为单位的数字，将数据库闪回到过去的时间，即从当前开始计算最大可以把数据库闪回到过去的时间。我们可以修改这个参数，使得闪回数据库可以闪回更长时间的数据，比如可以闪回到过去二天内的数据，即 24*60*2=2880

c. 设置闪回保留目标生存期（缺省为分钟，即小时）

```
SQL> show parameter db_flashback
SQL> alter system set db_flashback_retention_target=30;
```

d. 在 mount 状态下来启用 flashback，如在 open 状态下则出现下列错误提示

```
SQL>startup mount;
SQL> select status from v$instance;
SQL> alter database flashback on;
增加了后台进程 rvwr
SQL> ho ps -ef | grep rvwr
查看闪回区分配的大小，闪回分钟以内的数据需要的空间大小，oldest_flashback_time 说明了允许返回的最早的时间点
SQL> select oldest_flashback_scn old_flhbck_scn,oldest_flashback_time
old_flhbck_tim,retention_target rete_trgt,flashback_size/1024/1024
flhbck_siz,estimated_flashback_size/1024/1024 est_flhbck_size from v$flashback_database_log;
查看闪回
SQL> select * from v$flashback_database_stat;
查看 sga 中分配的闪回空间大小
SQL> select * from v$sgastat where name like 'flashback%';
查看生成的闪回日志
```

```
SQL> ho ls -hlt $ORACLE_BASE/flash_recovery_area/ORCL/flashback
```

关闭闪回数据库，一旦关闭闪回数据库特性，闪回日志将自动删除。

```
alter tablespace users flashback off;
```

查询表空间是否已经不被闪回保护。

```
select name,flashback_on from v$tablespace;
```

四、使用 flashback database 闪回数据库

步骤：

1）关闭数据库

2）启动数据库到 mount 状态

3）闪回至某个时间点，SCN 或 log sequence number

4）使用 resetlogs 打开数据库

使用 sqlplus 实现闪回的几种方法，使用一个时间标记或一个系统改变号

基于 SCN 闪回：FLASHBACK [STANDBY] DATABASE [<database_name>] TO [BEFORE] SCN <system_change_number>

基于时间戳闪回：FLASHBACK [STANDBY] DATABASE [<database_name>] TO [BEFORE] TIMESTMP <system_timestamp_value>

基于时点闪回：FLASHBACK [STANDBY] DATABASE [<database_name>] TO [BEFORE] RESTORE POINT <restore_point_name>

配置 Flash Recovery Area 要想使用 Flashback Database， 必须使用 Flash Recovery Area，因为 Flashback Database Log 只能保存在这里。要配置的 2 个参数如下，一个是大小，一个是位置。如果数据库是 RAC，flash recovery area 必须位于共享存储中。数据库必须处于 archivelog 模式。

1) 更改闪回区大小：

```
SQL>ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE=20G SCOPE=BOTH;
```

2) 更改闪回区路径：

```
SQL>ALTER SYSTEM SET DB_RECOVERY_FILE_DEST='/u01/app/oracle/flashback/orcl/'
SCOPE=BOTH;
```

对于 Flash Recovery Area，Oracle 是这样建议的，flash recovery area 设置的越大，flashback database 的恢复能力就越强，因此建议 flash recovery area 能够放的下所有的数据文件，增量备份，以及所有尚未备份的归档文件，当然还有它自己产生的 flashback logs。

在数据库运行过程中，oracle 自动向该区域写入文件，当剩余空间不足 15%的时候，它就会在 alert 中增加警告，提示你空间不足。但此时不会影响数据库的正常运转，直到所有空间统统被用掉之后，oracle 首先尝试删除寻些过期的文件，冗余文件或备份过的文件，

如果这些做完了，还是没有空闲空间的话，数据库就被 hang 住了。

启用数据库 Flashback 功能

1)数据库启动到 mount 状态

```
SQL> startup mount;
```

2)检查 Flashback 功能， 缺省时功能是关闭的。

```
SQL> select name, current_scn, flashback_on from v$database;

NAME CURRENT_SCN FLASHBACK_ON
-------- ----------- ------------------

DBA       945715       NO
```

3)启动 Flashback 功能

```
SQL> alter database flashback on;
```

数据库已更改。

4)设置初始化参数：DB_FLASHBACK_RETENTION_TARGET：

```
SQL>alter system set db_flashback_retention_target=1440 scope=both;
```

Tips:该参数用来控制 flashback log 数据保留的时间，或者说，你希望 flashback database 能够恢复的最早的时间点。默认值是 1440，单位是 minute，即 24 小时，需要注意的是该参数虽然未直接指定 flash recovery area 大小，但却受其制约，举个例子假如数据库每天有 10%左右的数据变动的话，如果该初始化参数值设置为 1440，则 flash recovery area 的大小至少要是当前数据库实际容量的 10%，如果该初始化参数设置为 2880，则 flash recovery area 的大小就至少是数据库所占容量的 20%。

5)启动数据库

```
SQL>alter database open;
```

Flashback Database 操作示例

Tips：做操作前先备份数据库

1. 检查是否启动了 flash recovery area:

```
SQL>show parameter db_recovery_file

NAME                    TYPE              VALUE
--------------------- ---------- ------------------------------

db_recovery_file_dest tring D:\oracle/flash_recovery_area
db_recovery_file_dest_size big integer 1G
```

2. 检查是否启用了归档

```
SQL>archive log list;
```

3. 检查是否启用了 flashback database

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

FLASHBACK_ON
------------------

YES
```

4. 查询当前的 scn

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

```
CURRENT_SCN
-----------
    947921
```

5. 查询当前的时间

```
SQL> select to_char(sysdate,'yy-mm-dd hh24:mi:ss') time from dual;
TIME
-----------------
09-10-14 14:37:05
```

6. 删除表 A

```
SQL> select * from A;
ID            NAME
---------- ----------
1            tianle
2             dave
SQL> drop table A;
表已删除。
SQL> commit;
```

7. Flashback Database 实际是对数据库的一个不完全恢复操作，因为需要关闭数据库重启到 mount 状态。

```
SQL> shutdown immediate
SQL> startup mount
```

8. 执行恢复：分 timestamp 或者 SCN 两种

```
SQL> flashback database to timestamp to_timestamp('09-10-14 14:37:05','yy-mm-dd hh24:mi:ss');
```

闪回完成。

或者：

```
SQL> Flashback database to scn 947921;
```

闪回完成。

9. 打开数据库：

alter database open resetlogs 打开数据库，当然，指定 scn 或者 timestamp 时间点之后产生的数据统统丢失。

```
SQL> alter database open resetlogs;
```

数据库已更改。

验证数据：

```
SQL> select * from A;
ID NAME
---------- ----------
1            tianle
2            dave
```

和 Flashback Database 相关的视图

1. V$database

这个视图可以查看是否启用了 Flashback database 功能

```
SQL> select flashback_on from v$database;
FLASHBACK_ON
------------------
   YES
```

2. V$flashback_database_log

Flashback Database 所能回退到的最早时间，取决与保留的 Flashback Database Log 的多少，该视图就可以查看许多有用的信息。

Oldest_flashback_scn/Oldest_flashback_time ： 这两列用来记录可以恢复到最早的时点

Flashback_size: 记录了当前使用的 Flash Recovery Area 空间的大小

Retention_target: 系统定义的策略

Estimated_flashback_size: 根据策略对需要的空间大小的估计值

```
SQL> select oldest_flashback_scn os,to_char(oldest_flashback_time,'yy-mm-dd
hh24:mi:ss') ot, retention_target rt,flashback_size fs,
estimated_flashback_size es from v$flashback_database_log;
OS              OT          RT       FS      ES
---------- ----------------- ---------- ---------- ----------
946088 09-10-14 13:49:59 1440 16384000 350920704
```

3. V$flashback_database_stat

这个视图用来对 Flashback log 空间情况进行更细粒度的记录和估计。这个视图以小时为单位记录单位时间内数据库的活动量，Flashback_Data 代表 Flashback log 产生数量，DB_Date 代表数据改变数量，Redo_Date 代表日志数量，通过这 3 个数量可以反映出数据的活动特点，更准确的预计 Flash Recovery Area 的空间需求

```
SQL> alter session set nls_date_format='hh24:mi:ss';
会话已更改。
SQL> select * from v$flashback_database_stat;
BEGIN_TI END_TIME FLASHBACK_DATA DB_DATA REDO_DATA ESTIMATED_FLASHBACK_SIZE
```

```
-------- -------- ------------- --------- --------- --------------------
14:43:10 15:15:28 6455296 29310976 3898368 0
```

示例：

```
SQL> flashback database to timestamp('2010-10-24 13:04:30','yyyy-mm-dd hh24:mi:ss');

SQL> flashback database to scn 918987;
SQL> flashback database ro restore point b1_load;
```

a. 基于时间戳闪回

```
[oracle@opc ~]$ uniread sqlplus / as sysdba
检查是否开启闪回
SYS@orcl> create user test1 identified by test1;
SYS@orcl> grant connect,resource to test1;
SYS@orcl> conn test1/test1
TEST1@orcl> create table t1 as select * from all_objects;
TEST1@orcl> select count(*) from t1;
获得系统当前的时间
TEST1@orcl> select to_char(sysdate,'yyyy-mm-dd hh24:mi:ss') tm from dual;
删除帐户 test1，同时删除 test1 下所有对象
TEST1@orcl> conn / as sysdba
SYS@orcl> drop user test1 cascade;
SYS@orcl> conn scott/scott;
SCOTT@orcl> create table tb_emp as select * from emp;
SYS@orcl> shutdown immediate;
SYS@orcl> startup mount;
闪回
SYS@orcl> flashback database to timestamp to_timestamp('2010-10-24 13:04:30','yyyy-mm-dd hh24:mi:ss');
SYS@orcl> alter database open resetlogs;
验证 test1 及对象 t1 被成功闪回
SQL> select count(*) from test1.t1;
闪回点之后修改的数据全部丢失
SQL> select count(1) from scott.tb_emp;
```

b. 基于 SCN 号闪回

```
获得当前的 SCN 号
SYS@orcl> select current_scn from v$database;
SYS@orcl> drop table test1.t1;
手动执行检查点
```

```
SYS@orcl> alter system checkpoint;
SYS@orcl> select file#,checkpoint_change# from v$datafile;
SYS@orcl>shutdown abort;
SYS@orcl>startup mount;
SYS@orcl>flashback database to scn 918987;
SYS@orcl> alter database open resetlogs;
SYS@orcl>select count(*) from test1.t1;
```

c.基于时点闪回

```
SYS@orcl> create table t(id int,col varchar2(20));
SYS@orcl> insert into t values(1,'ABC');
SYS@orcl> insert into t values(2,'DEF');
SYS@orcl> commit;
创建闪回点
SYS@orcl> create restore point aa;
SYS@orcl> insert into t values(3,'GHI');
SYS@orcl> commit;
SYS@orcl> select ora_rowscn,id,col from t;
SYS@orcl> shutdown immediate;
SYS@orcl> startup mount;
SYS@orcl> flashback database to restore point aa;
SYS@orcl> alter database open resetlogs;
闪回成功后，闪回点之后的数据丢失
SYS@orcl> select * from t;
```

七、 闪回数据归档

Oracle 引入闪回技术使得一些逻辑误操作不再需要利用归档日志和数据库备份进行时间点恢复，在前面介绍的诸多闪回技术中，Flashback Database 依赖于闪回日志，flashback drop 依赖 recycle bin，其它闪回技术都是依赖于 Undo 撤销数据，与数据库初始化参数 undo_retention 密切相关，所以就存在一个限制就是 undo 中的信息不能被覆盖。

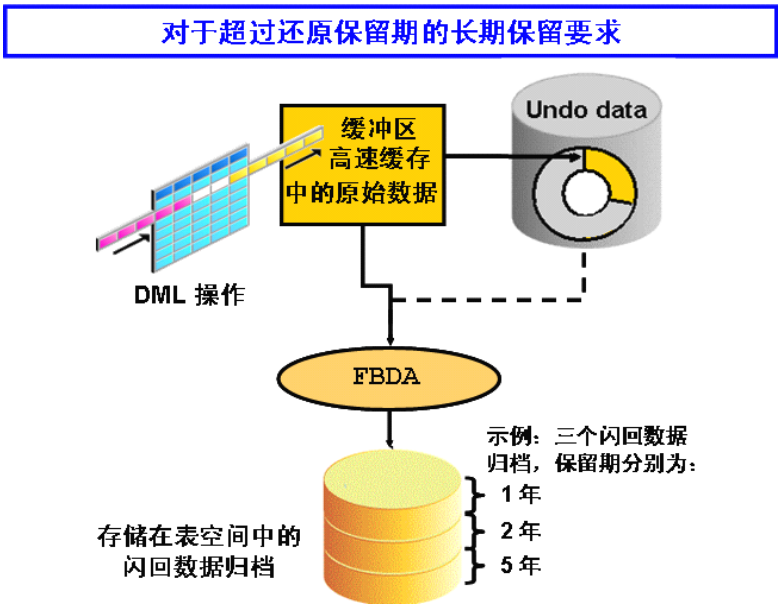
Oracle11g 中的闪回数据归档(Flashback Data Archive)与前面所说的闪回技术实现机制不同，它是通过将变化数据存储到创建的闪回归档区中，与 undo 区别开。同时 Flashback Data Archive 并不是记录数据库的所有变化，而只是记录了指定表的数据变化，是 Flashback database 的补充。

闪回数据归档是 oracle 11g 新特性。闪回查询需要用到撤销数据的闪回，会受到撤销数据保留时间的限制。闪回数据归档是将原本保存在 undo 表空间的撤销数据以一种历史表的形式保存在指定的普通表空间中。并且不像 undo_retention 那样影响整个数据库的设置，闪回数据归档可以指定存储某个表的数据，为特定表服务。

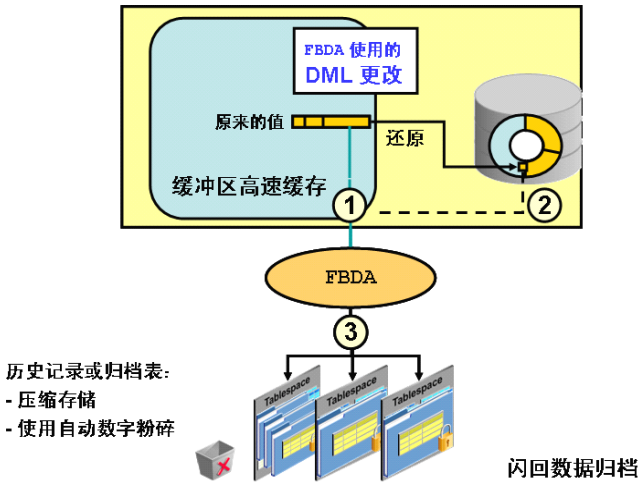
闪回数据归档区是闪回数据归档的历史数据存储区域，在一个系统中，可以有一个默认的闪回数据归档区，也可以创建其他许多的闪回数据归档区域。每一个闪回数据归档区都可以有一个唯一的名称。同时，每一个闪回数据归档区都对应了一定的数据保留策略。例如可以配置归档区 FLASHBACK_DATA_ARCHIVE_1 中的数据保留期为 1 年，而归档区 FLASHBACK_DATA_ARCHIVE_2 的数据保留期为 2 天或者更短。 以后如果将表放到对应的闪回数据归档区，则就按照该归档区的保留策略来保存历史数据。

闪回数据归档区是一个逻辑概念，是从一个或者多个表空间中拿出一定的空间，来保存表的修改历史，这样就摆脱了对 Undo 撤销数据的依赖，不利用 undo 就可以闪回到归档策略内的任何一个时间点上。

	闪回数据归档	闪回数据库
主要优点	访问任何时间点的数据而不会更改当前数据	使整个数据库实际回退到某个 时间点
操作	联机操作，启用跟踪，使用最少的资源	脱机操作，需要预配置和资源
粒度	表	数据库
访问时间点	每个表任意数目	每个数据库一个



体系结构



2．配置闪回数据归档

Oracle 11g 数据库使用 FBDA 功能是相当简单的，只需要经过几个简单的步骤即可：

(1) 创建或指定一个或多个表空间用于 FBDA 保留历史数据

(2) 随意指派一个 FBDA 作为数据库的默认 FBDA

(3) 指派一个用户账户作为 FBDA 管理员，授予它 FLASHBACK ARCHIVE ADMINISTER 系统权限

(4) 授予 FBDA 权限给适合的用户账号

(5) 授予 FLASHBACK 和 SELECT 权限给合适的 FBDA 表用户

(6) 为 FBDA 用户授予 DBMS_FLASHBACK 存储过程 EXECUTE 权限

3．访问历史记录数据的基本工作流：

创建闪回数据归档：

1． 创建闪回数据归档：

```
CREATE FLASHBACK ARCHIVE fla1 TABLESPACE tbs1 QUOTA 10G RETENTION 5 YEAR;
```

2． 对 FLA1 归档中的表启用历史记录跟踪：

```
ALTER TABLE inventory FLASHBACK ARCHIVE fla1;
```

3． 查看历史记录数据：

```
SELECT product_number, product_name,count FROM inventory AS OF TIMESTAMP
TO_TIMESTAMP('2007-01-01 00:00:00','YYYY-MM-DD HH24:MI:SS');
```

使用默认闪回归档：

1． 创建默认闪回数据归档：

```
CREATE FLASHBACK ARCHIVE DEFAULT fla2 TABLESPACE tbs1 QUOTA 10G RETENTION 2
YEAR;
```

2． 对表启用历史记录跟踪：

ALTER TABLE stock_data FLASHBACK ARCHIVE;

注：因为使用了默认的闪回数据归档，所以不需要提供闪回数据归档名称。

3. 禁用历史记录跟踪：

ALTER TABLE stock_data NO FLASHBACK ARCHIVE;

闪回数据归档填满后，会发生什么？

1）空间使用率达到 90%

2）引发错误：

ORA-55623 "Flashback Archive \"%s\" is blocking and tracking on all tables is suspended"

ORA-55617 "Flashback Archive \"%s\" runs out of space and tracking on \"%s\" is suspended"

3）生成预警日志条目

4）挂起跟踪

4. 维护闪回数据归档

1）增加空间：

ALTER FLASHBACK ARCHIVE fla1 ADD TABLESPACE tbs3 QUOTA 5G;

2）更改保留时间：

ALTER FLASHBACK ARCHIVE fla1 MODIFY RETENTION 2 YEAR;

3）清除数据：

ALTER FLASHBACK ARCHIVE fla1 PURGE BEFORE TIMESTAMP(SYSTIMESTAMP - INTERVAL '1' day);

4）删除闪回数据归档：

DROP FLASHBACK ARCHIVE fla1;

示例

1. 创建：

CREATE FLASHBACK ARCHIVE tax7_archive TABLESPACE tbs1 RETENTION 7 YEAR;

2. 访问历史记录数据：

SELECT symbol, stock_price FROM stock_data AS OF TIMESTAMP TO_TIMESTAMP ('2006-12-31 23:59:00', 'YYYY-MM-DD HH24:MI:SS')

3. 恢复数据：

INSERT INTO employees SELECT * FROM employees AS OF TIMESTAMP TO_TIMESTAMP('2007-06-12 11:30:00','YYYY-MM-DD HH24:MI:SS') WHERE name = 'JOE';

DELETE FROM hr.applicants WHERE application_date <=TO_DATE('11-10-2008','dd-mm-yyyy');

COMMIT;

INSERT INTO applicants SELECT * FROM applicants VERSIONS BETWEEN TIMESTAMP TO_TIMESTAMP('2008-12-04 10:00','yyyy-mm-dd hh24:mi') AND MAXVALUE WHERE VERSIONS_OPERATION = 'D';

COMMIT;

限制

对启用了闪回数据归档功能的表使用下述任一 DDL 语句都会导致错误 ORA-55610：

1） 执行以下任一操作的 ALTER TABLE 语句：

（1）删除、重命名或修改列

（2）执行分区或子分区操作

（3）将 LONG 列转换为 LOB 列

（4）包括 UPGRADE TABLE 子句（不管有无 INCLUDING DATA 子句）

2） DROP TABLE 语句

3） TRUNCATE TABLE 语句

4） RENAME TABLE 语句

查看结果： 视图名称说明

*_FLASHBACK_ARCHIVE 显示有关闪回数据归档的信息

*_FLASHBACK_ARCHIVE_TS 显示闪回数据归档的表空间

*_FLASHBACK_ARCHIVE_TABLES 显示有关启用了闪回归档的表的信息

一、创建闪回数据归档

1、为了创建闪回数据归档，必须拥有 DBA 角色或拥有系统权限 flashback archive administer。

sys@orcl> select * from dba_sys_privs where privilege like '%FLASH%';

创建表空间

sys@orcl> create tablespace flash_tbs1 datafile '/u01/app/oracle/oradata/orcl/flash_tbs1.dbf' size 200M;

sys@orcl> create tablespace tp1 datafile '/u01/app/oracle/oradata/orcl/tp1.dbf' size 200M;

sys@orcl> CREATE USER wl IDENTIFIED BY wl DEFAULT TABLESPACE flash_tbs1 TEMPORARY TABLESPACE temp;

sys@orcl> grant CONNECT,RESOURCE ,flashback archive administer to wl;

2、创建闪回归档

sys@orcl> create flashback archive flash1 tablespace flash_tbs1 quota 100M retention 5 year;

sys@orcl> CREATE FLASHBACK ARCHIVE DEFAULT fla2 TABLESPACE tp1 QUOTA 10G RETENTION 2 YEAR;

二、更改闪回数据归档

sys@orcl> alter flashback archive flash1 set default;

添加表空间

SQL> col FLASHBACK_ARCHIVE_NAME format a25
SQL> select FLASHBACK_ARCHIVE_NAME,STATUS from dba_FLASHBACK_ARCHIVE;
sys@orcl> alter flashback archive flash1 add tablespace tp1;
SQL>select TABLESPACE_NAME,FLASHBACK_ARCHIVE_NAME from dba_FLASHBACK_ARCHIVE_TS;
SQL>ALTER FLASHBACK ARCHIVE fla1 ADD TABLESPACE tbs3 QUOTA 5G;
删除表空间
sys@orcl> alter flashback archive flash1 remove tablespace tp1;
添加配额
sys@orcl> alter flashback archive flash1 modify tablespace flash_tbs1 quota 150M;
SQL> select TABLESPACE_NAME,FLASHBACK_ARCHIVE_NAME,QUOTA_IN_MB from dba_FLASHBACK_ARCHIVE_TS;
sys@orcl> alter flashback archive flash1 modify retention 3 year;
SQL> select FLASHBACK_ARCHIVE_NAME,RETENTION_IN_DAYS from dba_FLASHBACK_ARCHIVE;
清除所有
sys@orcl> alter flashback archive flash1 purge all;
清除 2 天前的
sys@orcl> alter flashback archive flash1 purge before timestamp (systimestamp - interval '2' day);
sys@orcl> alter flashback archive flash1 purge before scn 123344;

三、启用和禁用闪回数据归档

sys@orcl> **conn wl/wl**
1、在建表的同时就启用表的闪回日志
wl@orcl> create table t1(id int,name varchar2(10)) flashback archive flash1;
2、建表后，启用表的闪回日志
为表启用闪回数据归档,没指定表示使用数据库默认的
alter table t1 flashback archive;
为表启用闪回数据归档,指定在特定的闪回数据归档中存储表的变化
alter table t1 flashback archive flash1;
3、数据库将把 T1 表的数据,归档到默认的闪回数据归档中
wl@orcl> select * from dba_flashback_archive_tables;
4、在使用闪回数据归档前，必须设置默认闪回数据归档
wl@orcl> select flashback_archive_name,status from dba_flashback_archive;
5、禁用闪回数据归档
wl@orcl> alter table t1 no flashback archive;

四、闪回数据归档的限制

使用闪回归档的过程中有某些限制。对于已经启用闪回的表，不能使用 DDL 命令 drop column（11r2 可以 drop column）,可以 add column 命令。
wl@orcl> select * from v$version;
11G R2 版可以删除列
wl@orcl> alter table t1 drop column name;
可以增加列
wl@orcl> alter table t1 add(name varchar2(100));
可以重命名表

```
wl@orcl> alter table t1 rename to t10;
```

可以截断表

```
wl@orcl> truncate table t10;
```

不可以删除表

```
wl@orcl> drop table t10;
 drop table t10
           *
 ERROR at line 1:
 ORA-55610: Invalid DDL statement on history-tracked table
wl@orcl> alter table t10 no flashback archive;
```

禁用后可以删除表

```
wl@orcl> drop table t10;
```

五、监控闪回数据归档

1、查哪些表已经启用了闪回数据归档

```
wl@orcl> select * from dba_flashback_archive_tables;
```

2、查数据库中所有的闪回数据归档

```
wl@orcl> select flashback_archive_name,retention_in_days from dba_flashback_archive;
```

3、查有关闪回数据归档所使用的表空间的信息

```
wl@orcl>select flashback_archive_name,tablespace_name,quota_in_mb from dba_flashback_archive_ts;
```

六、使用闪回数据归档(实例)

```
sys@orcl> create tablespace flash_tbs2 datafile '/u01/app/oracle/oradata/orcl/flash_tbs2.dbf' size 200M;
sys@orcl> CREATE USER w2 IDENTIFIED BY w2 DEFAULT TABLESPACE flash_tbs2 TEMPORARY TABLESPACE temp;

sys@orcl> grant  CONNECT,RESOURCE ,flashback archive administer to w2;

sys@orcl> create flashback archive flash2 tablespace flash_tbs2  quota 100M retention 5 year;
sys@orcl>conn  w2/w2
w2@orcl> create table test_wl (id int,name varchar2(10));
w2@orcl> alter table test_wl flashback archive flash2;
w2@orcl> begin for i in 1 .. 100 loop insert into test_wl values(i,'wl'||i); commit; end loop; end;
/
w2@orcl> select count(*) from test_wl;
w2@orcl> col FLASHBACK_ARCHIVE_NAME for a10
w2@orcl> col TABLE_NAME for a10
w2@orcl> col ARCHIVE_TABLE_NAME for a20
w2@orcl> col OWNER_NAME for a5
w2@orcl> select * from dba_flashback_archive_tables;
TABLE_NAME  OWNER   FLASHBACK_   ARCHIVE_TABLE_NAME      STATUS
----------  -----   ----------   ----------------------  -------
   TEST_WL      W2          FLASH1       SYS_FBA_HIST_88644      ENABLED
W2@orcl> select count(*) from SYS_FBA_HIST_88644;
W2@orcl> conn / as sysdba
```

```
sys@orcl> select current_scn from v$database;
            CURRENT_SCN
            ----------------------
          1050526
sys@orcl>conn w2/w2
w2@orcl> delete from test_wl;
w2@orcl> commit;
w2@orcl> conn / as sysdba
sys@orcl> select current_scn from v$database;
            CURRENT_SCN
            ----------------------
          1050633
sys@orcl>conn w2/w2
w2@orcl> select count(*) from test_wl as of scn  1050526;
w2@orcl> select count(*) from test_wl as of scn  1050633;
```

七、删除闪回归档数据

```
SQL> conn / as sysdba
drop flashback archive flash1;
```