如何dump Undo段的信息

UNDO_RETENTION

- 用来指定UNDO段中数据保存的最短时间，以秒为单位，是一个动态参数
- 可以在实例运行时随时修改，11g默认是900秒
- 在undo空间未满的情况下, 在undo_retention设定的时间内, 事务不会被覆盖, 超过该时间表示可以被覆盖
- 修改

```
SQL> alter system set undo_retention=3600
```

UNDO空间估算

- 至少要满足**undo_retention**需要的表空间大小。

估算所需回退的总大小 1）先计算平均每秒产生的回退 v$undostat UPS : number of undo data blocks generated per second 2）show parameter undo 查看 UR : undo_retention 900 DBS : db_block_size 3）回退所需总空间为：上述两个值相乘

估算UNDO大小语句

```
SQL> select ( (UR * (UPS * DBS)) + (DBS * 24) ) / 1024 AS "KB"
     from
         ( select value as UR
           from v$parameter
           where name = 'undo_retention'
         ),
         ( select (sum(undoblks) / sum( ( (end_time - begin_time) * 24* 3600) ) ) as UPS
           from v$undostat
         ),
         ( select value as DBS
           from v$parameter
           where name = 'db_block_size'
         );
```

每个时间段产生的UNDO块数

```
select to_char(begin_time,'yyyymmdd hh24:mi:ss'),to_char(end_time,'yyyymmdd hh24:mi:ss'),
(end_time-begin_time)*24*3600 ,
undoblks
from v$undostat order by 1;
```

获取UNDO信息

与Undo相关的视图

1、DBA_ROLLBACK_SEGS

```
# 所有undo段
SELECT segment_name,owner,tablespace_name,status
FROM dba_rollback_segs;

•SYS: Refers to a private undo segment
•PUBLIC: Refers to a public undo segment


# 查询历史undo是否还有事务(包含回滚事务)
SELECT a.tablespace_name,a.segment_name,b.ktuxesta,b.ktuxecfl,
b.ktuxeusn||'.'||b.ktuxeslt||'.'||b.ktuxesqn trans
FROM dba_rollback_segs a, x$ktuxe b
WHERE a.segment_id = b.ktuxeusn
AND a.tablespace_name = UPPER('&tsname')
AND b.ktuxesta <> 'INACTIVE';

Enter value for tsname: 输入undo表空间名
```

2、VROLLNAME 和 VROLLSTAT

```
# undo段的状态
SELECT n.name, n.usn,s.extents, s.rssize,s.hwmsize,s.xacts, s.status
FROM v$rollname n, v$rollstat s
WHERE n.usn = s.usn;

# 当前在线的回退段
select * from v$rollname;
```

3、V$UNDOSTAT

- 记录了undo历史统计信息，说明了系统的工作负载。
- undo的建议值。
- 10分钟采集一次。
- 共有576行，4天一个循环周期。

```
# undo_retention 最大建议值
select tuned_undoretention ,to_char(begin_time,'yyyymmdd HH24:mi:ss') ,
to_char(end_time,'yyyymmdd HH24:mi:ss')
from v$undostat
where tuned_undoretention=(select max(tuned_undoretention) from v$undostat);

# 某一时间段内undo_retention建议值
select tuned_undoretention ,undoblks,to_char(begin_time,'yyyymmdd HH24:mi:ss')
from v$undostat
where to_char(begin_time,'yyyymmdd HH24:mi:ss') > '20181025 21:40:00'
and to_char(begin_time,'yyyymmdd HH24:mi:ss') < '20181029 01:40:00';

# 查看undo的使用情况
select to_char(BEGIN_TIME,'yyyymmdd HH24:MI:SS') begin_time,
to_char(END_TIME,'yyyymmdd HH24:MI:SS') end_time,
ACTIVEBLKS,UNEXPIREDBLKS,EXPIREDBLKS
from v$undostat
order by 1;
```

4、V$SESSION 和 V$TRANSACTION

```
# 查看哪个事务正在使用undo
SQL> SELECT s.sid,s.serial#,s.username, t.xidusn,
t.ubafil, t.ubablk, t.used_ublk,t.used_urec
FROM v$session s, v$transaction t
WHERE s.saddr = t.ses_addr;

# 查看当前事务的OS进程PID
select p.SPID from v$session s,v$process p
where s.PADDR=p.ADDR
and s.SID in (select s.sid From v$transaction t,v$session s where t.addr=s.taddr);


# 一个事务，所需要的undo有多少块
SQL> begin
2 for i in 1..10000
3 loop
4 insert into t values(i,'ttt');
5 end loop;
6 end;
7 /

PL/SQL procedure successfully completed.

SQL> select addr,used_ublk from v$transaction;

ADDR USED_UBLK
---------------- ----------
000000008B365830 87
```

如何dump Undo段的信息

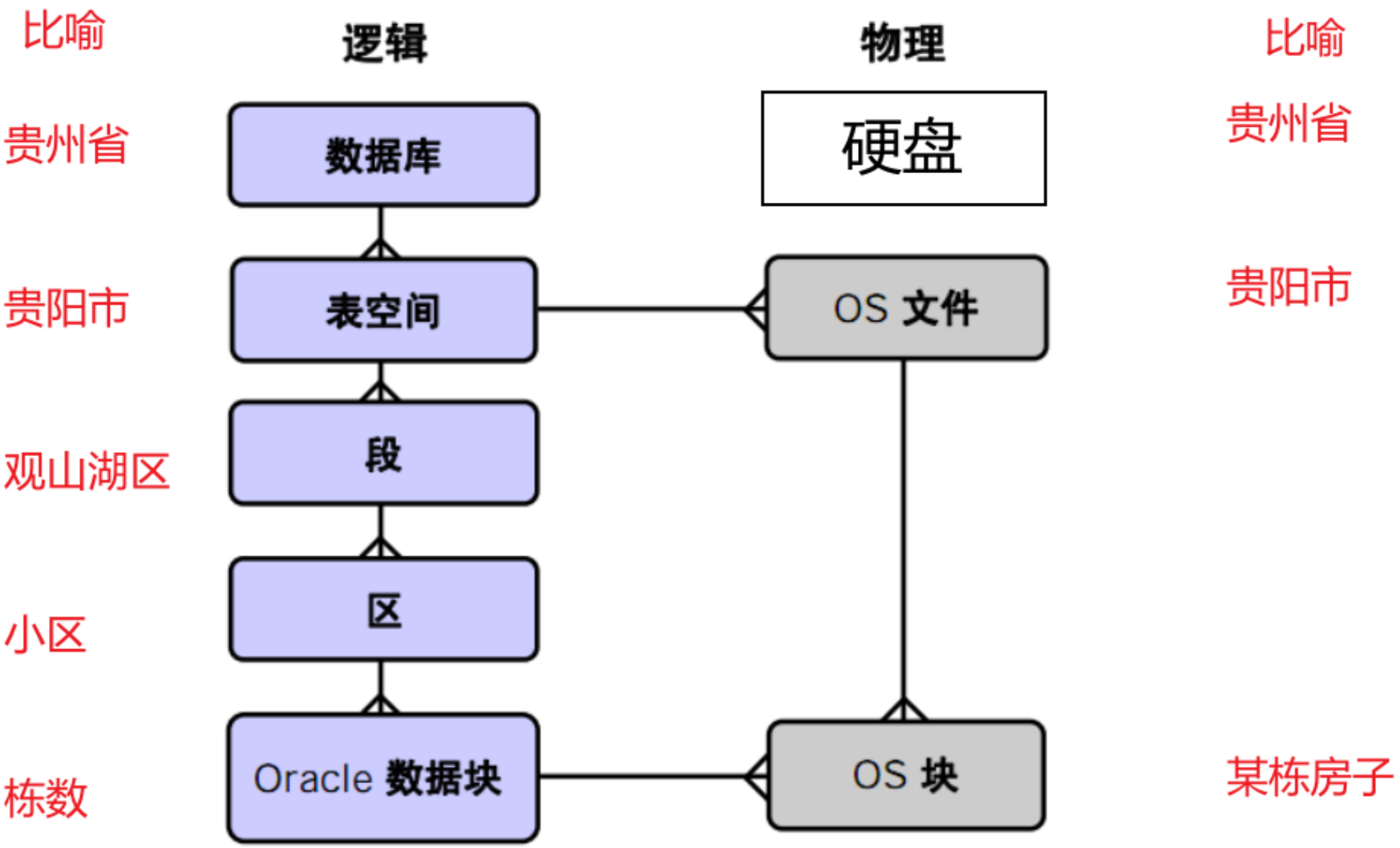
1）dump 回退段的头信息

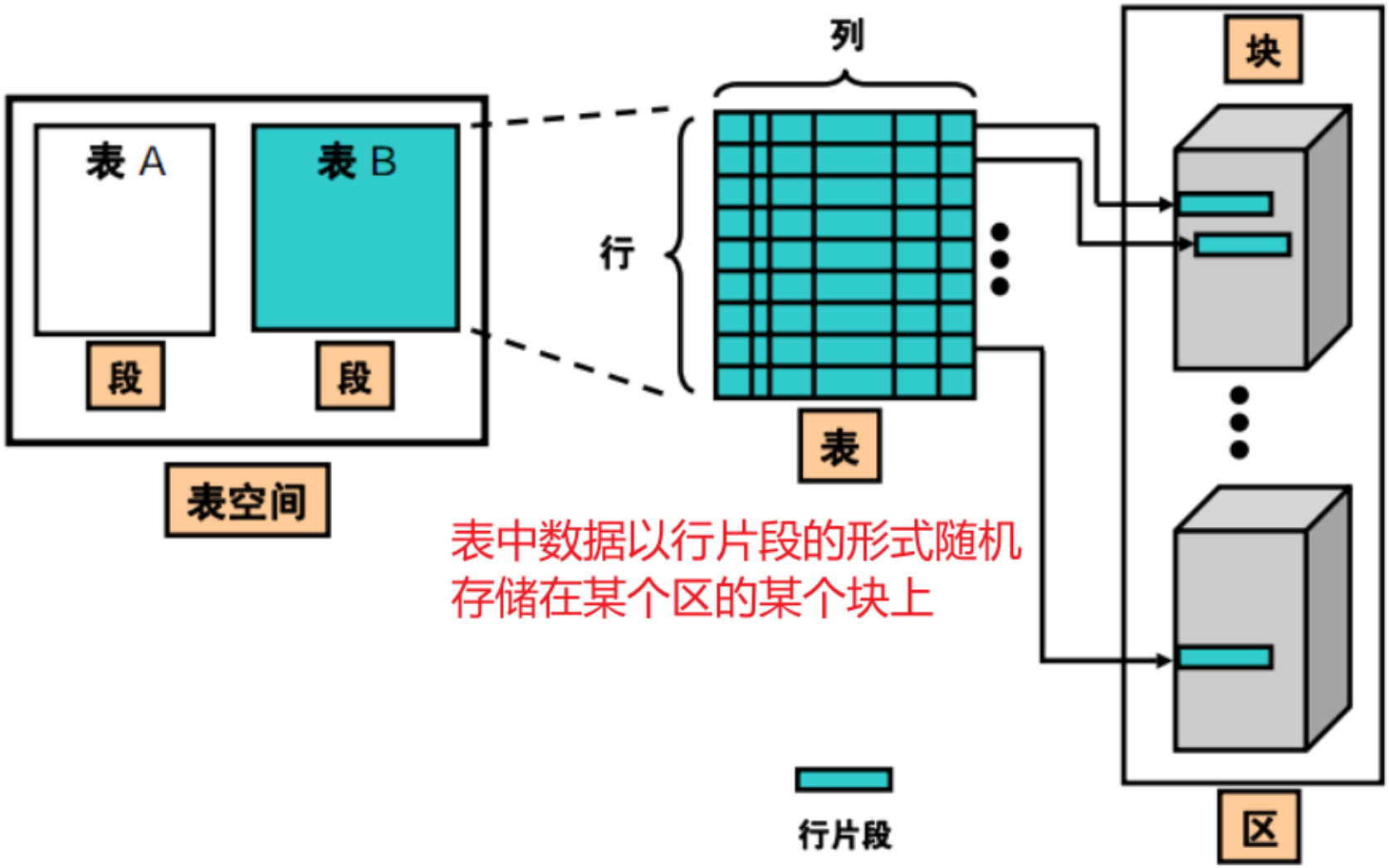
```
SQL> ALTER SYSTEM DUMP UNDO HEADER '_SYSSMU15_1377674437$';
```

2）dump 回退段的交易信息

```
SQL> ALTER SYSTEM DUMP UNDO BLOCK 'segment_name' XID xidusn xidslot xidsqn;
```

存储结构


比喻
逻辑
物理
比喻
贵州省
数据库
硬盘
贵州省
贵阳市
表空间
OS 文件
贵阳市
观山湖区
段
小区
区
栋数
Oracle 数据块
OS 块
某栋房子



逻辑结构

Oracle 数据库在**逻辑上**将数据存储在**表空间**中，在**物理上**将数据存储在**数据文件**中。

表空间

- 只能属于一个数据库
- 包括一个或多个数据文件
- 可进一步划分为逻辑存储单元
- 是存储方案对象数据的资料档案库
- 表空间中可以包含多个段，段中可以包含多个区，一个区可以包含多个块。
- 按照对系统的作用，可以将表空间分为：
    - SYSTEM表空间
        - 随数据库创建，包含数据字典，包含SYSTEM还原段。
        -
    - 非SYSTEM表空间
        - 用于分开存储段，易于空间管理，控制分配给用户的空间量。
        - system auxiliary 辅助
        - 10g新引入的新的表空间, 分担system表空间的压力
        - 不能改名称
        - 可以offline,但部分数据库功能受影响
- 表空间主要有：
    - 系统表空间 system

        - 数据库内最重要的表空间
        - 在建立数据库时,就诞生了
        - 在数据库open的时候必须online
        - 该表空间含有数据字典的基表
        - 含有包,函数,视图,存储过程的定义
        - 原则上不存放用户的数据
    - Sysaux表空间
        - system auxiliary 辅助
        - 10g新引入的新的表空间
        - 分担system表空间的压力
        - 不能改名称
        - 可以offline,但部分数据库功能受影响
    - 回滚段表空间 undo
        - UNDO表空间用于存储回滚段，保存回滚信息，不能包含任何其它对象，其中的区要在本地管理， 创建语句只能使用DATAFILE和EXTENT MANAGEMENT 子句。
    - 临时表空间 : 可以建多个，并且可以同时使用。

        - 临时表空间用于排序操作，保存排序、HASH连接等操作还有临时表中的临时信息， 不能包含任何永久对象，建议在本地管理区。
        - 缺省临时表空间：指定数据库范围内的缺省临时表空间，不要使用SYSTEM表空间存储临时数据。
        - 建多个的好处：避免竞争。
    - 用户表空间
- 任何用户数据都不应该存放在以下两个表空间内：
    - SYSTEM表空间，存放数据字典，即描述数据库的元数据。
    - SYSAUX表空间，存放由内部管理和维护程序包生成的动态的、可能很大量的数据。
- 除了用户表空间，其他表空间不得随意更改和破坏。
- 一个DB想存在，至少有**SYSTEM及UNDO**表空间。
- 一个表空间下可以有多个数据文件。
- 一个数据文件只能属于一个表空间，我们可以通过为表空间增加或减少数据文件或改变数据文件的大小，来改变表空间的大小。

```
# 视图
# # DBA_FREE_SPACE describes the free extents in all tablespaces in the database.
# # DBA_DATA_FILES describes database files.
# # DBA_SEGMENTS describes the storage allocated for all segments in the database.
# # DBA_EXTENTS describes the extents comprising the segments in all tablespaces in the database.

# 增
# # 创建表空间testbs 对应81K的数据文件'/oradata/zzdb1/testbs_01.dbf' 如果文件已存在则覆盖, 关闭自动扩展分配
SQL> create tablespace testbs datafile '/oradata/zzdb1/testbs_01.dbf' size 81K reuse autoextend off;
# # 给表空间testbs增加数据文件
SQL> alter tablespace testbs add datafile '/oradata/zzdb1/testbs_02.dbf' size 82K autoextend off;
# # 开启表空间自动扩展
SQL> alter database datafile FILE_ID autoextend on;

# 删
# # 删除表空间中的对象与数据文件(慎用)
SQL> drop tablespace testbs including contents and datafiles;
# # 关闭表空间自动扩展
SQL> alter database datafile 8 autoextend off;

# 改
# # 手动分配
SQL> alter table t1 allocate extent(datafile '/oradata/zzdb1/testbs_01.dbf');
# # 手工收回未使用的范围
SQL> alter table t1 deallocate unused;

# 查
# # 查询表所在表空间
SQL> select table_name,tablespace_name from user_tables where table_name='T1';
# # 查看新表是否已分配segment
SQL> select segment_name,segment_type from user_segments;
SQL> select segment_name,extent_id,file_id,block_id,blocks
  2  from dba_extents where owner='SCOTT' and segment_name='T1';
```

```
# # 查看表空间每个块的大小
SQL> select tablespace_name ,file_id,block_id,bytes from dba_free_space where tablespace_name='TESTBS';
# # 查看是否开启自动扩展分配
SQL> select file_name,AUTOEXTENSIBLE from dba_data_files where tablespace_name='TESTBS';
# # 查看表T1空间分配
SQL> select segment_name,extent_id,file_id,block_id,blocks
  2  from dba_extents where owner='SCOTT' and segment_name='T1';
```

数据块

- 块的大小在参数文件中的**db_block_size** 参数指定，数据库块大小应该是**操作系统块的倍数**。
- Block大小设置： 1）对于数据仓库OLAP(Online Analysis Process)的数据库应用，Block要尽量大。 因为会返回大量的数据，而且一般用户不多，最多的查询方式应该是全表扫描。 2）OLTP(Online Transaction Process)应用，Block尽量不要太大。

  因为查询会返回少量的数据，但是用户很多，并发很大，最多的方式应该是索引读。 如果块太大，容易导致大量并发查询及更新操作都指向同一个数据块，产生热点块竞争。
- Block的组成： 1）数据块头：概要信息，例如块地址及此块所属的段的类型(表还是索引) 2）表目录区：只要有一行数据插入到数据库块中，该行数据所在的表的信息将被存储在此区域。 3）行目录区：存放插入的行的地址。 4）可用空间区：块中的空余空间。由Oracle的PCTFREE参数设置。PCTFREE=10说明大约有10%的空间空余。 5）行数据区：存储具体的行的信息或者索引的信息。占用了块的绝大部分空间。

Extent 区

- 区是由若干个连续的块组成。
- 为了避免过度扩展。因为块的尺寸太小了，如果以块的尺寸为单位进行扩展，Oracle会很繁忙。
- 逻辑上是连续的块，但实际上，在磁盘做RAID和文件系统层，区是可以跨越物理磁盘的。

- 区无法跨越数据文件，数据文件不是逻辑结构的一部分，他为表空间提供实际的存储空间

Segment 段

- 段，通常包含一个或多个区。一个段，代表一些数据文件中的空间。
- 同属一个段的空间，有可能不连续。 因为段是由区组合的，如果段包含两个区，这两个区在磁盘上并不连续，段的空间也就不连续了。 我们常说的表，是从另一个方面来描述段，表指的是存储在段中的数据。而段指的是来存储这些数据的磁盘空间。
- 简单点说，**表是指数据，而段是指存储数据的空间。**
- 段无法跨越表空间，也就是说一个段，只能隶属与某一个表空间，但是可以跨越同一表空间下的多个数据文件。 比如说一个表段，他包含 3 个区，前两个区可以在甲，后一个区在乙。

数据文件

- 只能属于一个表空间和一个数据库
- 是构成表空间的基础文件
- 一个数据库最多可以包含65536个数据文件。

存储数据库文件的方式

- 文件系统 文件系统是数据库使用最广泛的存储方式，优势就是管理方便，简单。
- ASM（Automatic Storage Management）自动存储管理 ---- 集群中共享磁盘 有点类似LVM管理方式，却又不一样。ASM需要专门的ASM实例管理。 数据库实例可以通过 **direct I/O** (读写效率高的原因是因为应用程序可以直接从设备读取， 不需要经过操作系统的缓冲。)读写 ASM 上的磁盘，数据不需要经过ASM实例缓冲。 ASM实例**只能启动到nomount**，它跟普通的数据库实例不一样，没有控制文件，也没有数据文件，仅仅是用来管理ASM存储。

- Raw device 裸设备

  以前只有裸设备可以Direct I/O

附加

执行脚本创建Scott用户

```
cd $ORACLE_HOME/rdbms/admin
sqlplus / as sysdba
SQL>@utlsampl
从 Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options 断开
sqlplus / as sysdba
SQL>conn scott/tiger
Connected.
```

对回收站中的表进行回退

```
SQL> drop table t1;

表已删除。

SQL> show recyclebin;
ORIGINAL NAME     RECYCLEBIN NAME        OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
T1         BIN$1EKxcAUNL8HgUAAKDwIOKA==$0 TABLE         2021-12-29:12:51:38
SQL> flashback table t1 to before drop;

闪回完成。

SQL> select * from t1;

    ID
----------
       234


SQL> drop table t1 purge;

表已删除。

SQL> show recyclebin;
```