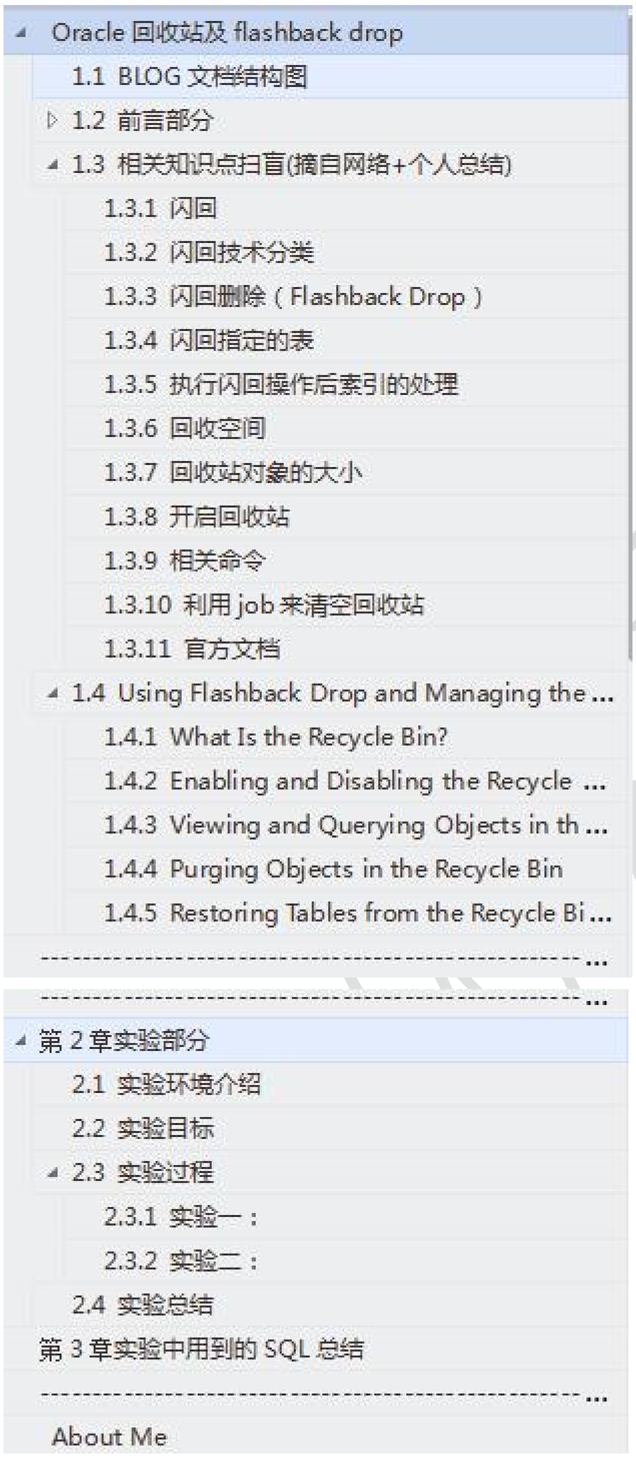
Oracle 回收站及 flashback drop

1.1 BLOG 文档结构图



1.2 前言部分

1.2.1 导读和注意事项

各位技术爱好者，看完本文后，你可以掌握如下的技能，也可以学到一些其它你所不知道的知识，~O(∩_∩)O~：

① Oracle 回收站的介绍(开启、关闭、清空)

② 闪回体系中 flashback drop 的介绍

③ job 批量删除回收站中的对象

④ dba_free_space 查询速度慢的问题（MOS:Queries on DBA_FREE_SPACE are Slow (文档 ID 271169.1)）

Tips：

① 本文在 ITpub（http://blog.itpub.net/26736162）和博客园(http://www.cnblogs.com/lhrbest)有同步更新

② 文章中用到的所有代码，相关软件，相关资料请前往小麦苗的云盘下载（http://blog.itpub.net/26736162/viewspace-1624453/）

③ 若文章代码格式有错乱，推荐使用搜狗、360 或 QQ 浏览器，也可以下载 pdf 格式的文档来查看，pdf 文档下载地址：http://blog.itpub.net/26736162/viewspace-1624453/

④ 本篇 BLOG 中命令的输出部分需要特别关注的地方我都用灰色背景和粉红色字体来表示，比如下边的例子中，thread 1 的最大归档日志号为 33，thread 2 的最大归档日志号为 43 是需要特别关注的地方；而命令一般使用黄色背景和红色字体标注；对代码或代码输出部分的注释一般采用蓝色字体表示。

```
  List of Archived Logs in backup set 11
  Thrd Seq     Low SCN    Low Time            Next SCN   Next Time
  ---- ------- ---------- ------------------- ---------- ---------
  1    32      1621589    2015-05-29 11:09:52 1625242    2015-05-29 11:15:48
  1    33      1625242    2015-05-29 11:15:48 1625293    2015-05-29 11:15:58
  2    42      1613951    2015-05-29 10:41:18 1625245    2015-05-29 11:15:49
  2    43      1625245    2015-05-29 11:15:49 1625253    2015-05-29 11:15:53



[ZHLHRDB1:root]:/>lsvg -o
T_XDESK_APP1_vg
rootvg
[ZHLHRDB1:root]:/>
00:27:22 SQL> alter tablespace idxtbs read write;
```

====》2097152*512/1024/1024/1024=1G

本文如有错误或不完善的地方请大家多多指正，ITPUB 留言或 QQ 皆可，您的批评指正是我写作的最大动力。

1.2.2 相关参考文章链接

【TSPITR】RMAN 表空间基于时间点的自动恢复

http://blog.itpub.net/26736162/viewspace-1671741/

更多闪回知识参考：http://blog.csdn.net/tianlesoftware/article/details/4677378

1.2.3 本文简介

今天执行健康检查脚本的时候脚本一直卡在了表空间查询这块，瞅了一眼 SQL，根据经验小麦苗预估是由于 DBA_FREE_SPACE 视图的缘故，这个视图若回收站的对象很多的话查询就会非常的慢，接下来单独执行 select count(1) from dba_free_space;果然非常的慢，没办法只能先将回收站的数据清理了再来查询表空间了，而回收站大约有 200W 的数据量，执行 purge DBA_RECYCLEBIN 非常慢，那就只能采用 job 的并行技术了，这个在本文最后给出了脚本，这个脚本比较通用，小麦苗以前做开发的时候经常性的用这个脚本，希望各位朋友能掌握。

说到回收站就涉及到闪回，而闪回分很多类，我们今天着重看看 flashback drop 和回收站。

1.3 相关知识点扫盲(摘自网络+个人总结)

1.3.1 闪回

当授权用户犯错，您需要使用工具来更正这些错误。Oracle 数据库 10g 提供了一系列人为错误更正技术，称为**闪回**。闪回从根本上改变了数据恢复。过去，数据库在几分钟内就可能损坏，但需要几小时才能恢复。利用闪回技术，更正错误的时间与错误发生时间几乎相同。而且它非常易用，使用一条短命令便可恢复整个数据库，而不必执行复杂的程序。闪回技术提供了一个 SQL 界面，能够快速分析和修复人为错误。闪回技术为本地数据损坏提供了细粒度外部分析和修复，如当错误删除客户订单时。闪回技术还支持修复更多广泛的损坏，同时快速避免长时间停机，如当本月的所有客户订单都被删除时。闪回技术是 Oracle 数据库独有的特性，支持各级恢复，包括行、事务、表、表空间和数据库范围。闪回技术是数据库恢复技术历史上一次重大的进步，从根本上改变了数据恢复。

Oracle 9i 实现了基于回滚段的闪回查询（Flashback Query）技术，即从回滚段中读取一定时间内对表进行操作的数据，恢复错误的 DML 操作。

在 Oracle 10g 中，除提高了闪回查询功能，实现了闪回版本查询、闪回事务查询外，还实现了闪回表、闪回删除和闪回数据库的功能。

采用闪回技术，可以针对行级和事务级发生过变化的数据进行恢复，减少了数据恢复的时间，而且操作简单，通过 SQL 语句就可以实现数据的恢复，大大提高了数据库恢复的效率。

1.3.2 闪回技术分类

1. 闪回查询（Flashback Query）：查询过去某个时间点或某个 SCN 值时表中的数据信息；
2. 闪回版本查询（Flashback Version Query）：查询过去某个时间段或某个 SCN 段内表中数据的变化情况；
3. 闪回事务查询（Flashback Transaction Query）：查看某个事务或所有事务在过去一段时间对数据进行的修改；
4. 闪回表（Flashback Table）：将表恢复到过去的某个时间点或某个 SCN 值时的状态；
5. 闪回删除（Flashback Drop）：将已经删除的表及其关联对象恢复到删除前的状态；

6． 闪回数据库（Flashback Database）：将数据库恢复到过去某个时间点或某个 SCN 值时的状态。

注意

① 闪回查询、闪回版本查询、闪回事务查询以及闪回表主要是基于撤销表空间中的回滚信息实现的；

② 闪回删除是基于 Oracle 10g 中的回收站（Recycle Bin）特性实现的；

③ 闪回数据库是基于闪回恢复区（Flash Recovery Area）中的闪回日志来实现的；

④ 为了使用数据库的闪回技术，必须启用撤销表空间自动管理回滚信息。

⑤ 如果要使用闪回删除技术和闪回数据库技术，还需要启用回收站、闪回恢复区。

1.3.3 闪回删除（Flashback Drop）

Oracle10g 之前，一旦删除了一个表，那么该表就会从数据字典里面删除。要恢复该表，需要进行不完全恢复。Oracle10g 以后，当我们删除表时，默认 Oracle 只是在数据库字典里面对被删的表的进行了重命名，并没有真正的把表删除。 Flashback Drop 是从 Oracle 10g 开始出现的，用于恢复用户误删除的对象(包括表，索引等)，这个技术依赖于 Tablespace Recycle Bin(表空间回收站)，这个功能和 windows 的回收站非常类似。

回收站：用来维护表被删除前的名字与删除后系统生成的名字之间的对应关系的数据字典，表上的相关对象（索引、触发器等）也会一并进入回收站

被 drop 掉的表能否闪回来与两个因素相关：

1、该表所在表空间的大小有关，即如果表空间够大，用 drop 语句删除的表，并不是真正的从数据库中删除，而是把表改成 BIN$开头的表，但是如果表空间不够大，在有新数据要存入该表空间的时候，就会覆盖这些 BIN$表的物理空间，此时也就没有办法利用闪回恢复该表了

2、删除该表的时候是否用的 purge，如果在 drop 的时候使用了 purge，则该表就被从表空间中彻底的被删除了，如果要恢复，必须用以前的备份恢复，可以用 TSPITR 或 12c 可以直接从备份集中恢复单张表。

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 是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。

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。

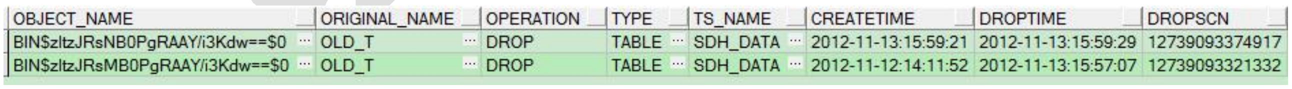
4). 当删除表时，信赖于该表的物化视图也会同时删除，但是由于物化视图并不会被放入 recycle bin，因此当你执行 flashback table to before drop 时，也不能恢复依赖其的物化视图，需要 dba 手工介入重新创建。

5). 对于 Recycle Bin 中的对象，只支持查询.

1.3.4 闪回指定的表

有时可能一个表被反复的建立和 drop，这样在 recycle 一个 origianl name 的有多个记录相对，默认将是恢复最后一个，如果要恢复指定的一个可以用他们的 OBJECT_name 通过指定 name 的方式。

```
select * from user_recyclebin t where t.original_name='OLD_T';
```



OBJECT_NAME	ORIGINAL_NAME	OPERATION	TYPE	TS_NAME	CREATETIME	DROPTIME	DROPSCN
BIN$zltzJRsNB0PgRAAY/i3Kdw==$0	OLD_T	DROP	TABLE	SDH_DATA	2012-11-13:15:59:21	2012-11-13:15:59:29	12739093374917
BIN$zltzJRsMB0PgRAAY/i3Kdw==$0	OLD_T	DROP	TABLE	SDH_DATA	2012-11-12:14:11:52	2012-11-13:15:57:07	12739093321332

```
select * from "BIN$zltzJRsMB0PgRAAY/i3Kdw==$0";
```

闪回回收站中指定的表：

```
flashback table "BIN$zltzJRsMB0PgRAAY/i3Kdw==$0" to before drop;
```

默认采用的是先进后出的方式，总是恢复最后被删除的表。

1.3.5　　执行闪回操作后索引的处理

表被恢复以后，表上的索引，需要重建，虽然索引可以随着表的闪回而闪回，但是闪回后的索引仍然使用 recyclebin 的名字，因此我们需要重建索引。

1.3.6　　回收空间

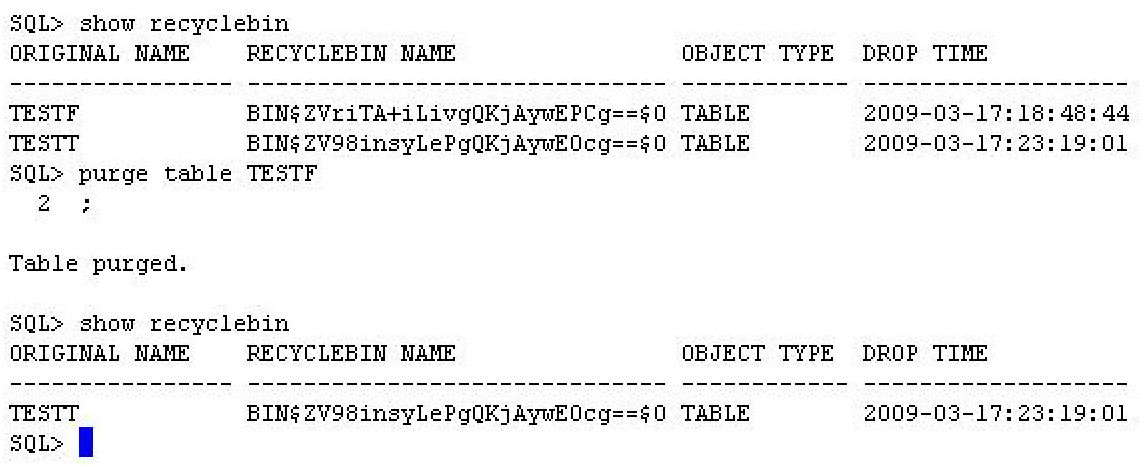
既然被删除的对象没有被物理的释放，那么该物理空间是如何进行回收的呢？Oracle 通过两种方式进行回收。

1、自动回收

当表空间出现压力时，Oracle 会首先使用表空间里不属于回收站的对象所占用的可用空间，如果这部分空间用完，仍然存在空间压力，则释放回收站里面最老的那些对象所占用的空间。直至释放完毕所有的空间，然后扩展数据文件（前提是数据文件支持自动扩展）

2、手工回收

使用 purge 命令来释放回收站里的对象所占用的空间



```
SQL> show recyclebin
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
TESTF            BIN$ZVriTA+iLivgQKjAywEPCg==$0 TABLE        2009-03-17:18:48:44
TESTT            BIN$ZV98insyLePgQKjAywE0cg==$0 TABLE        2009-03-17:23:19:01
SQL> purge table TESTF
  2  ;

Table purged.

SQL> show recyclebin
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
TESTT            BIN$ZV98insyLePgQKjAywE0cg==$0 TABLE        2009-03-17:23:19:01
SQL>
```

1.3.7　　回收站对象的大小

表空间的占用大小中 dba_free_space 中不包括回收站中的对象的大小。

```
 SELECT nvl(a.owner, '合计') owner,
        round(SUM(a.space *
                  (SELECT value FROM v$parameter WHERE name = 'db_block_size'))
```

```
/ 1024 / 1024,
              2) recyb_size_M,
         count(1) recyb_cnt
     FROM dba_recyclebin a
    GROUP BY ROLLUP(a.owner);
```

1.3.8 开启回收站

Property	Description
Parameter type	String
Syntax	RECYCLEBIN = { on | off }
Default value	on
Modifiable	ALTER SESSION, ALTER SYSTEM ... DEFERRED
Basic	No

RECYCLEBIN is used to control whether the Flashback Drop capability is turned on or off. If the parameter is set to off, then dropped tables do not go into the recycle bin. If this parameter is set to on, then dropped tables go into the recycle bin and can be recovered

```
alter system set recyclebin = on scope=spfile;
alter session set recyclebin= on;
```

1.3.9 回收站相关 SQL 命令

```
select * from "BIN$zltzJRsMB0PgRAAY/i3Kdw==$0";
select * from user_recyclebin;
select * from dba_recyclebin;

flashback table "BIN$zltzJRsMB0PgRAAY/i3Kdw==$0" to before drop;
flashback table t  to before drop rename to old_t;
show recyclebin
purge table test;//我们 purge 回收站中表的时候，相对应的索引也会被删除。
```

purge index “索引名字”；

purge tablespace users;清除回收站里面属于 users 表空间的对象所占用的空间

purge user_recyclebin：清除回收站里面属于当前用户的所有对象所占用的空间

purge dba_recyclebin：清除回收站里所有对象所占用的空间

drop table xxxx purge;直接删除表，不进入回收站。

1.3.10 利用 job 来清空回收站

若回收站内容较多，则用 dba_recyclebin 清空回收站比较慢，这个时候可以考虑采用 job 分割的方法来晴空回收站，脚本如下：

```
    SELECT D.owner,COUNT(1) FROM dba_recyclebin D GROUP BY D.owner;

    CREATE TABLE XB_recyclebin_LHR NOLOGGING AS
    SELECT ROWNUM RN, 'PURGE ' || A.type || ' ' || A.owner || '."' || A.object_name
|| '"' EXEC_SQL
      FROM dba_recyclebin A
     where a.type = 'TABLE';

    CREATE INDEX IDX_recyclebin_rn  on XB_recyclebin_LHR(rn) NOLOGGING
PARALLEL;

    create table XB_SPLIT_JOB_LHR
    (
       startrownum NUMBER(18),
       endrownum   NUMBER(18),
       flag        NUMBER(1)
    );
    SELECT * FROM xb_split_job_lhr;


    CREATE OR REPLACE PROCEDURE pro_split_job_lhr AUTHID CURRENT_USER IS
----------------------------------------------------------------------
        -- copy on 2012/4/2 23:28:21 by lhr
        --function：该存过用来分隔数据来建立 job
        --需要进行处理的数据量 ，需要处理的表加 rn 列，值取 rownum，rn 列加索引

        --alter table tmp_dp_idp_lhr add rn number;
        /* CREATE INDEX IDX_tmp_dp_idp_lhr_rn  on tmp_dp_idp_lhr(rn)
        TABLESPACE SDH_INDEX ONLINE  NOLOGGING COMPUTE STATISTICS PARALLEL;*/

        /*  create table XB_SPLIT_JOB_LHR
      (
        startrownum NUMBER(18),
        endrownum   NUMBER(18),
        flag        NUMBER(1)
      )*/

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

    n              NUMBER; --创建的 job 数

    j              NUMBER := 0;
    n_startrownum   NUMBER;
    n_endrownum     NUMBER;

    n_patchnum      NUMBER := 20000; -- 每批处理的记录数       ----modify

    v_jobname       VARCHAR2(200);

    v_count        NUMBER; --需要处理的表的数据量

  BEGIN

    SELECT COUNT(1) INTO v_count FROM XB_recyclebin_LHR; ----modify

    --需要创建的 job 个数

    n := trunc(v_count / n_patchnum) + 1;

     EXECUTE IMMEDIATE 'truncate table xb_split_job_lhr';
    WHILE j < n LOOP

      --得到 rownum

      n_startrownum := j * n_patchnum + 1;

      IF j = n - 1 THEN

        n_endrownum := v_count;
      ELSE
        n_endrownum := (j + 1) * n_patchnum;
      END IF;

      INSERT INTO xb_split_job_lhr
        (startrownum, endrownum)
      VALUES
        (n_startrownum, n_endrownum);
      COMMIT;
```

```
          j := j + 1;
      END LOOP;


      --循环创建 job
      j             := 0;

      FOR cur IN (SELECT * FROM xb_split_job_lhr) LOOP

          v_jobname := 'JOB_SUBJOB_SPLIT_LHR' || (j + 1);
          dbms_scheduler.create_job(job_name          => v_jobname,
                                    job_type          => 'STORED_PROCEDURE',
                                    job_action        => 'PRO_SUB_SPLIT_LHR',
--modify
                                    number_of_arguments => 2,
                                    start_date        => SYSDATE + 1 / 5760, --
15 秒后启动作业
                                    repeat_interval     => NULL,
                                    end_date          => NULL,
                                    job_class         => 'DEFAULT_JOB_CLASS',
                                    enabled           => FALSE,
                                    auto_drop         => TRUE,
                                    comments          => 'to split
job_subjob_Split_lhr');
          COMMIT;

          dbms_scheduler.set_job_argument_value(job_name          =>
v_jobname,
                                                argument_position => 1,
                                                argument_value    =>
cur.startrownum);
          COMMIT;
          dbms_scheduler.set_job_argument_value(job_name          =>
v_jobname,
                                                argument_position => 2,
                                                argument_value    =>
cur.endrownum);
          COMMIT;
          dbms_scheduler.enable(v_jobname);
          j := j + 1;
      END LOOP;
      COMMIT;


      -----等待所有的子 job 执行完
```

```
    LOOP

        SELECT COUNT(1)
        INTO   v_count
        FROM   xb_split_job_lhr t
        WHERE  t.flag IS NULL;

        IF v_count = 0 THEN
           EXIT;
        ELSE

           dbms_lock.sleep(10); ---存过休息 10 秒

        END IF;

    END LOOP;
   EXECUTE IMMEDIATE 'purge dba_recyclebin';
EXCEPTION
    WHEN OTHERS THEN
      NULL;

END pro_split_job_lhr;


create or replace procedure pro_sub_split_lhr(p_startrownum number,
                                              p_endrownum   number) is

begin

  for cur in (SELECT A.EXEC_SQL
              FROM XB_recyclebin_LHR A ---modify
             where A.rn <= p_endrownum
              and A.rn >= p_startrownum) loop
    begin
      EXECUTE IMMEDIATE CUR.EXEC_SQL;
    exception
      when others then
        null;
    end;
  end loop;

  commit;


  --更新标志

  update xb_split_job_lhr t
     set t.flag = 1
   where t.startrownum = p_startrownum
```

```
          and t.endrownum = p_endrownum;
        commit;

    exception

        when others then

            null;

    end pro_sub_split_lhr;
```

1.3.11 MOS

文档 15996460.8. Bug 15996460 - Performance issue on DBA_FREE_SPACE (文档 ID 15996460.8).mhtml

文档 271169.1 Queries on DBA_FREE_SPACE are Slow (文档 ID 271169.1).mhtml

文档 1904677.1 Query Against DBA_FREE_SPACE is Slow After Applying 11.2.0.4 (文档 ID 1904677.1).mhtml

1.3.12 官方文档

Managing Tables.mhtml

When you drop a table, normally the database does not immediately release the space associated with the table. Rather, the database renames the table and places it in a recycle bin, where it can later be recovered with the FLASHBACK TABLE statement if you find that you dropped the table in error. If you should want to immediately release the space associated with the table at the time you issue the DROP TABLE statement, include the PURGE clause as shown in the following statement:

```
DROP TABLE hr.admin_emp PURGE;
```

1.4 Using Flashback Drop and Managing the Recycle Bin

When you drop a table, the database does not immediately remove the space associated with the table. The database renames the table and places it and any associated objects in a recycle bin, where, in case the table was dropped in error, it can be recovered at a later time. This feature is called Flashback Drop, and the FLASHBACK TABLE statement is used to restore the table. Before discussing the use of the FLASHBACK TABLE statement for this purpose, it is important to understand how the recycle bin works, and how you manage its contents.

This section contains the following topics:

1.4.1 What Is the Recycle Bin?

The recycle bin is actually a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space. They continue to count against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints.

Each user can be thought of as having his own recycle bin, because, unless a user has the SYSDBA privilege, the only objects that the user has access to in the recycle bin are those that the user owns. A user can view his objects in the recycle bin using the following statement:

```
SELECT * FROM RECYCLEBIN;
```

When you drop a tablespace including its contents, the objects in the tablespace are not placed in the recycle bin and the database purges any entries in the recycle bin for objects located in the tablespace. The database also purges any recycle bin entries for objects in a tablespace when you drop the tablespace, not including contents, and the tablespace is otherwise empty. Likewise:

- When you drop a user, any objects belonging to the user are not placed in the recycle bin and any objects in the recycle bin are purged.

- When you drop a cluster, its member tables are not placed in the recycle bin and any former member tables in the recycle bin are purged.

When you drop a type, any dependent objects such as subtypes are not placed in the recycle bin and any former dependent objects in the recycle bin are purged.

•

Object Naming in the Recycle Bin

When a dropped table is moved to the recycle bin, the table and its associated objects are given system-generated names. This is necessary to avoid name conflicts that may arise if multiple tables have the same name. This could occur under the following circumstances:

•

A user drops a table, re-creates it with the same name, then drops it again.

•

•

Two users have tables with the same name, and both users drop their tables.

•

The renaming convention is as follows:

```
BIN$unique_id$version
```

where:

•

unique_id is a 26-character globally unique identifier for this object, which makes the recycle bin name unique across all databases

•

•

version is a version number assigned by the database

•

1.4.2 Enabling and Disabling the Recycle Bin

When the recycle bin is enabled, dropped tables and their dependent objects are placed in the recycle bin. When the recycle bin is disabled, dropped tables and their dependent objects are not placed in the recycle bin; they are just dropped, and you must use other means to recover them (such as recovering from backup).

Disabling the recycle bin does not purge or otherwise affect objects already in the recycle bin. The recycle bin is enabled by default.

You enable and disable the recycle bin by changing the `recyclebin` initialization parameter. This parameter is not dynamic, so a database restart is required when you change it with an `ALTER SYSTEM` statement.

To disable the recycle bin:

1.

Issue one of the following statements:

2.

```
ALTER SESSION SET recyclebin = OFF;
ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

3.

4.

If you used `ALTER SYSTEM`, restart the database.

5.

To enable the recycle bin:

1.

Issue one of the following statements:

2.

```
ALTER SESSION SET recyclebin = ON;

ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

3.

4.

If you used `ALTER SYSTEM`, restart the database.

5.

See Also:

- "About Initialization Parameters and Initialization Parameter Files" for more information on initialization parameters
-
- "Changing Initialization Parameter Values" for a description of dynamic and static initialization parameters
-

1.4.3 Viewing and Querying Objects in the Recycle Bin

Oracle Database provides two views for obtaining information about objects in the recycle bin:

View	Description
USER_RECYCLEBIN	This view can be used by users to see their own dropped objects in the recycle bin. It has a synonym RECYCLEBIN, for ease of use.
DBA_RECYCLEBIN	This view gives administrators visibility to all dropped objects in the recycle bin

One use for these views is to identify the name that the database has assigned to a dropped object, as shown in the following example:

```
SELECT object_name, original_name FROM dba_recyclebin
   WHERE owner = 'HR';

OBJECT_NAME                    ORIGINAL_NAME
------------------------------ --------------------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 EMPLOYEES
```

You can also view the contents of the recycle bin using the SQL*Plus command **SHOW RECYCLEBIN.**

```
SQL> show recyclebin

ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
EMPLOYEES        BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TABLE        2003-10-27:14:00:19
```

You can query objects that are in the recycle bin, just as you can query other objects. However, you must specify the name of the object as it is identified in the recycle bin. For example:

```
SELECT * FROM "BIN$yrMKlZaVMhfgNAgAIMenRA==$0";
```

1.4.4 Purging Objects in the Recycle Bin

If you decide that you are never going to restore an item from the recycle bin, you can use the PURGE statement to remove the items and their associated objects from the recycle bin and release their storage space. You need the same privileges as if you were dropping the item.

When you use the PURGE statement to purge a table, you can use the name that the table is known by in the recycle bin or the original name of the table. The recycle bin name can be obtained from either the DBA_ or USER_RECYCLEBIN view as shown in "Viewing and Querying Objects in the Recycle Bin". The following hypothetical example purges the table hr.int_admin_emp, which was renamed to BIN$jsleilx392mk2=293$0 when it was placed in the recycle bin:

```
PURGE TABLE "BIN$jsleilx392mk2=293$0";
```

You can achieve the same result with the following statement:

```
PURGE TABLE int_admin_emp;
```

You can use the PURGE statement to purge all the objects in the recycle bin that are from a specified tablespace or only the tablespace objects belonging to a specified user, as shown in the following examples:

```
PURGE TABLESPACE example;
PURGE TABLESPACE example USER oe;
```

Users can purge the recycle bin of their own objects, and release space for objects, by using the following statement:

```
PURGE RECYCLEBIN;
```

If you have the SYSDBA privilege, then you can purge the entire recycle bin by specifying DBA_RECYCLEBIN, instead of RECYCLEBIN in the previous statement.

You can also use the PURGE statement to purge an index from the recycle bin or to purge from the recycle bin all objects in a specified tablespace.

See Also:

Oracle Database SQL Language Reference for more information on the PURGE statement

1.4.5 Restoring Tables from the Recycle Bin

Use the FLASHBACK TABLE ... TO BEFORE DROP statement to recover objects from the recycle bin. You can specify either the name of the table in the recycle bin or the original table name. An optional RENAME TO clause lets you rename the table as you recover it. The recycle bin name can be obtained from either the DBA_ or USER_RECYCLEBIN view as shown in "Viewing and Querying Objects in the Recycle Bin". To use the FLASHBACK TABLE ... TO BEFORE DROP statement, you need the same privileges required to drop the table.

The following example restores int_admin_emp table and assigns to it a new name:

```
FLASHBACK TABLE int_admin_emp TO BEFORE DROP
   RENAME TO int2_admin_emp;
```

The system-generated recycle bin name is very useful if you have dropped a table multiple times. For example, suppose you have three versions of the int2_admin_emp table in the recycle bin and you want to recover the second version. You can do this by issuing two FLASHBACK TABLE statements, or you can query the recycle bin and then flashback to the appropriate system-generated name, as shown in the following example. Including the create time in the query can help you verify that you are restoring the correct table.

```
SELECT object_name, original_name, createtime FROM recyclebin;
```

```
OBJECT_NAME                    ORIGINAL_NAME   CREATETIME
------------------------------ --------------- -------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 INT2_ADMIN_EMP  2006-02-05:21:05:52
BIN$yrMKlZaVMhfgNAgAIMenRA==$0 INT2_ADMIN_EMP  2006-02-05:21:25:13
BIN$yrMKlZaQMhfgNAgAIMenRA==$0 INT2_ADMIN_EMP  2006-02-05:22:05:53

FLASHBACK TABLE "BIN$yrMKlZaVMhfgNAgAIMenRA==$0" TO BEFORE DROP;
```

Restoring Dependent Objects

When you restore a table from the recycle bin, dependent objects such as indexes do not get their original names back; they retain their system-generated recycle bin names. You must manually rename dependent objects to restore their original names. If you plan to manually restore original names for dependent objects, ensure that you make note of each dependent object's system-generated recycle bin name before you restore the table.

The following is an example of restoring the original names of some of the indexes of the dropped table JOB_HISTORY, from the HR sample schema. The example assumes that you are logged in as the HR user.

1.

After dropping JOB_HISTORY and before restoring it from the recycle bin, run the following query:

2.

```
SELECT OBJECT_NAME, ORIGINAL_NAME, TYPE FROM RECYCLEBIN;

OBJECT_NAME                    ORIGINAL_NAME             TYPE
------------------------------ ------------------------- --------
BIN$DBo9UChtZSbgQFeMiAdCcQ==$0 JHIST_JOB_IX              INDEX
BIN$DBo9UChuZSbgQFeMiAdCcQ==$0 JHIST_EMPLOYEE_IX         INDEX
BIN$DBo9UChvZSbgQFeMiAdCcQ==$0 JHIST_DEPARTMENT_IX       INDEX
BIN$DBo9UChwZSbgQFeMiAdCcQ==$0 JHIST_EMP_ID_ST_DATE_PK   INDEX
BIN$DBo9UChxZSbgQFeMiAdCcQ==$0 JOB_HISTORY               TABLE
```

3.

4.

Restore the table with the following command:

5.

```
FLASHBACK TABLE JOB_HISTORY TO BEFORE DROP;
```

6.

7.

Run the following query to verify that all JOB_HISTORY indexes retained their system-generated recycle bin names:

8.

```
SELECT INDEX_NAME FROM USER_INDEXES WHERE TABLE_NAME = 'JOB_HISTORY';

INDEX_NAME
------------------------------
BIN$DBo9UChwZSbgQFeMiAdCcQ==$0
BIN$DBo9UChtZSbgQFeMiAdCcQ==$0
```

```
BIN$DBo9UChuZSbgQFeMiAdCcQ==$0
BIN$DBo9UChvZSbgQFeMiAdCcQ==$0
```

9.

10.

Restore the original names of the first two indexes as follows:

11.

```
ALTER INDEX "BIN$DBo9UChtZSbgQFeMiAdCcQ==$0" RENAME TO JHIST_JOB_IX;
ALTER INDEX "BIN$DBo9UChuZSbgQFeMiAdCcQ==$0" RENAME TO JHIST_EMPLOYEE_IX;
```

12.

Note that double quotes are required around the system-generated names.

13.

第 2 章　实验部分

2.1 实验环境介绍

项目	primary db
db 类型	单实例
db version	11.2.0.2.0
db 存储	ASM

2.2 实验目标

本次我们模拟 2 个实验：

1、系统表空间的对象不能闪回

2、在版本为 11.2.0.3 及以下的情况下，当回收站对象过多时查询表空间大小时涉及到 dba_free_space 很慢，用 purge dba_recyclebin 又太慢，所以采用 job 来批量删除

2.3 实验过程

2.3.1 实验一：

首先建立测试库并打开回收站功能：

```
[ZT1MXP11:oracle]:/oracle>dbca -silent -createDatabase -templateName General_Purpose.dbc -responseFile
NO_VALUE \
> -gdbname oralhr  -sid oralhr \
> -sysPassword oracle -systemPassword lhr \
> -datafileDestination 'DATA2/' -recoveryAreaDestination 'DATA2/' \
> -redoLogFileSize 50 \
> -storageType ASM -asmsnmpPassword lhr  -diskGroupName 'DATA2' \
> -characterset AL32UTF8 -nationalCharacterSet AL16UTF16 \
> -sampleSchema false \
> -automaticMemoryManagement true -totalMemory 2048 \
> -databaseType OLTP  \
> -emConfiguration NONE
Copying database files
1% complete
3% complete
10% complete
17% complete
24% complete
31% complete
35% complete
Creating and starting Oracle instance
37% complete
42% complete
47% complete
52% complete
53% complete
56% complete
58% complete
Registering database with Oracle Restart
64% complete
Completing Database Creation
68% complete
71% complete
75% complete
85% complete
96% complete
100% complete
Look at the log file "/oracle/app/oracle/cfgtoollogs/dbca/oralhr/oralhr.log" for further details.
[ZT1MXP11:oracle]:/oracle>more /oracle/app/oracle/cfgtoollogs/dbca/oralhr/oralhr.log
Copying database files
DBCA_PROGRESS : 1%
DBCA_PROGRESS : 3%
DBCA_PROGRESS : 10%
DBCA_PROGRESS : 17%
DBCA_PROGRESS : 24%
DBCA_PROGRESS : 31%
DBCA_PROGRESS : 35%
Creating and starting Oracle instance
DBCA_PROGRESS : 37%
DBCA_PROGRESS : 42%
DBCA_PROGRESS : 47%
DBCA_PROGRESS : 52%
DBCA_PROGRESS : 53%
DBCA_PROGRESS : 56%
DBCA_PROGRESS : 58%
Registering database with Oracle Restart
```

```
DBCA_PROGRESS : 64%
Completing Database Creation
DBCA_PROGRESS : 68%
DBCA_PROGRESS : 71%
DBCA_PROGRESS : 75%
DBCA_PROGRESS : 85%
DBCA_PROGRESS : 96%
DBCA_PROGRESS : 100%
Database creation complete. For details check the logfiles at:
 /oracle/app/oracle/cfgtoollogs/dbca/oralhr.
Database Information:
Global Database Name:oralhr
System Identifier(SID):oralhr
[ZT1MXP11:oracle]:/oracle>

[ZT1MXP11:oracle]:/oracle>ORACLE_SID=oralhr
[ZT1MXP11:oracle]:/oracle>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.2.0 Production on Mon Jun 27 10:12:18 2016

Copyright (c) 1982, 2010, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options

SYS@oralhr> show parameter recy

NAME                                 TYPE                   VALUE
------------------------------------ ---------------------- ------------------------------
buffer_pool_recycle                  string
db_recycle_cache_size                big integer            0
recyclebin                           string                 OFF
SYS@oralhr> alter system set recyclebin=on scope=spfile;

System altered.

SYS@oralhr> startup force;  ====》慎用，不推荐
ORACLE instance started.

Total System Global Area 3089920000 bytes
Fixed Size                  2250360 bytes
Variable Size             721422728 bytes
Database Buffers         2348810240 bytes
Redo Buffers               17436672 bytes
Database mounted.
Database opened.
SYS@oralhr>  show parameter recy

NAME                                 TYPE                   VALUE
------------------------------------ ---------------------- ------------------------------
buffer_pool_recycle                  string
db_recycle_cache_size                big integer            0
recyclebin                           string                 ON

SYS@oralhr> create table tb_20160627_lhr as select * from dual;

Table created.

SYS@oralhr> drop table tb_20160627_lhr;

Table dropped.

SYS@oralhr> select * from dba_recyclebin;

no rows selected

SYS@oralhr> create table  tb_20160627_lhr tablespace users as select * from dual;
```

```
Table created.

SYS@oralhr> drop table tb_20160627_lhr;

Table dropped.

SYS@oralhr> select * from dba_recyclebin;

OWNER                          OBJECT_NAME                    ORIGINAL_NAME                    OPERATION TYPE
TS_NAME                        CREATETIME
------------------------------ ------------------------------ -------------------------------- ---------
------------------------- ------------------------------ -------------------
DROPTIME              DROPSCN PARTITION_NAME                   CAN CAN    RELATED BASE_OBJECT PURGE_OBJECT      SPACE
------------------- ---------- -------------------------------- --- --- ---------- ----------- ------------ ----------
SYS                            BIN$Njoq6PZtAGzgUxa8wKsAbA==$0 TB_20160627_LHR                  DROP      TABLE
USERS                          2016-06-27:11:16:01
2016-06-27:11:16:05    7268816                                  YES YES     450051      450051       450051          8


SYS@oralhr>
```

说明 SYSTEM 表空间的表 drop 后不会进入回收站空间。

2.3.2　　实验二：

我们遵循如下的实验步骤：

1、创建 10W 张表，并创建索引

2、开启回收站

3、删除创建的表

4、查询 dba_free_space 视图

5、清空回收站后再查询 dba_free_space 视图

实验开始：我们首先利用建表的脚本创建出 10W 张表，可以多开几个窗口，并行建表加快速度，另外，10W 张表大约占用 users 表空间 6G 多，这个需要注意一下：

等待 10W 张表建好的时候取消建表语句：

```
[ZT1MXP11:oracle]:/oracle>ORACLE_SID=oralhr
[ZT1MXP11:oracle]:/oracle>sqlplus / as sysdba

SQL*Plus: Release 11.2.0.2.0 Production on Mon Jun 27 09:12:18 2016
```

```
Copyright (c) 1982, 2010, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options

SYS@oralhr> begin
  2
  3    for cur in 1 .. 100000 loop
  4
  5      execute immediate 'create table tb_recyclebin_' || cur ||
  6                        ' nologging tablespace users as select * from dual';
  7      execute immediate 'create index idx_recyclebin_' || cur ||
  8                        ' on tb_recyclebin_' || cur ||' (dummy) nologging tablespace users';
  9
 10    end loop;
 11  end;
 12  /

PL/SQL procedure successfully completed.

SYS@oralhr>

SYS@oralhr> SELECT count(1) FROM dba_tables d WHERE d.table_name like 'TB_RECYCLEBIN%';

  COUNT(1)
----------
    187796

SYS@oralhr> SELECT sum(d.bytes)/1024/1024 FROM dba_segments d WHERE d.segment_name like
'%TB_RECYCLEBIN%';

SUM(D.BYTES)/1024/1024
----------------------
              11737.25
```

开启回收站：

```
SYS@oralhr> show parameter recy

NAME                                 TYPE                   VALUE
------------------------------------ ---------------------- ------------------------------
buffer_pool_recycle                  string
db_recycle_cache_size                big integer            0
recyclebin                           string                 OFF
SYS@oralhr> alter system set recyclebin=on scope=spfile;

System altered.

SYS@oralhr> startup force;  ====》慎用，不推荐
ORACLE instance started.

Total System Global Area 3089920000 bytes
Fixed Size                  2250360 bytes
Variable Size             721422728 bytes
Database Buffers         2348810240 bytes
Redo Buffers               17436672 bytes
Database mounted.
Database opened.
SYS@oralhr>  show parameter recy

NAME                                 TYPE                   VALUE
------------------------------------ ---------------------- ------------------------------
buffer_pool_recycle                  string
db_recycle_cache_size                big integer            0
```

```
recycleblin                          string                 ON
```

接下来我们 drop 掉刚刚创建的表：

```
SYS@oralhr> WITH wt1 AS
  2   (SELECT ts.TABLESPACE_NAME,
  3           df.all_bytes,
  4           decode(df.TYPE,
  5                  ’D’,
  6                  nvl(fs.FREESIZ, 0),
  7                  ’T’,
  8                  df.all_bytes - nvl(fs.FREESIZ, 0)) FREESIZ,
  9           df.MAXSIZ,
 10           ts.BLOCK_SIZE,
 11           ts.LOGGING,
 12           ts.FORCE_LOGGING,
 13           ts.CONTENTS,
 14           ts.EXTENT_MANAGEMENT,
 15           ts.SEGMENT_SPACE_MANAGEMENT,
 16           ts.RETENTION,
 17           ts.DEF_TAB_COMPRESSION,
 18           df.ts_df_count
 19    FROM   dba_tablespaces ts,
 20           (SELECT ’D’ TYPE,
 21                   TABLESPACE_NAME,
 22                   COUNT(*) ts_df_count,
 23                   SUM(BYTES) all_bytes,
 24                   SUM(decode(MAXBYTES, 0, BYTES, MAXBYTES)) MAXSIZ
 25            FROM   dba_data_files d
 26            GROUP  BY TABLESPACE_NAME
 27            UNION ALL
 28            SELECT ’T’,
 29                   TABLESPACE_NAME,
 30                   COUNT(*) ts_df_count,
 31                   SUM(BYTES) all_bytes,
 32                   SUM(decode(MAXBYTES, 0, BYTES, MAXBYTES))
 33            FROM   dba_temp_files d
 34            GROUP  BY TABLESPACE_NAME) df,
 35           (SELECT TABLESPACE_NAME,
 36                   SUM(BYTES) FREESIZ
 37            FROM   dba_free_space
 38            GROUP  BY TABLESPACE_NAME
 39            UNION ALL
 40            SELECT tablespace_name,
 41                   SUM(d.BLOCK_SIZE * a.BLOCKS) bytes
 42            FROM   gv$sort_usage   a,
 43                   dba_tablespaces d
 44            WHERE  a.tablespace = d.tablespace_name
 45            GROUP  BY tablespace_name) fs
 46    WHERE  ts.TABLESPACE_NAME = df.TABLESPACE_NAME
 47    AND    ts.TABLESPACE_NAME = fs.TABLESPACE_NAME(+))
 48  SELECT (SELECT A.TS#
 49          FROM   V$TABLESPACE A
 50          WHERE  A.NAME = UPPER(t.TABLESPACE_NAME)) TS#,
 51         t.TABLESPACE_NAME TS_Name,
 52         round(t.all_bytes / 1024 / 1024) ts_size_M,
 53         round(t.freesiz / 1024 / 1024) Free_Size_M,
 54         round((t.all_bytes - t.FREESIZ) / 1024 / 1024) Used_Size_M,
 55         round((t.all_bytes - t.FREESIZ) * 100 / t.all_bytes, 3) Used_per,
 56         round(MAXSIZ / 1024 / 1024/1024, 3) MAX_Size_g,
 57         round(decode(MAXSIZ, 0, to_number(NULL), (t.all_bytes - FREESIZ)) * 100 /
 58               MAXSIZ,
 59               3) USED_per_MAX,
 60         round(t.BLOCK_SIZE) BLOCK_SIZE,
 61         t.LOGGING,
 62         t.ts_df_count
 63  FROM   wt1 t
 64  UNION ALL
 65  SELECT to_number('') TS#,
 66         'ALL TS:' TS_Name,
 67         round(SUM(t.all_bytes) / 1024 / 1024, 3) ts_size_M,
 68         round(SUM(t.freesiz) / 1024 / 1024) Free_Size_m,
 69         round(SUM(t.all_bytes - t.FREESIZ) / 1024 / 1024) Used_Size_M,
 70         round(SUM(t.all_bytes - t.FREESIZ) * 100 / SUM(t.all_bytes), 3) Used_per,
 71         round(SUM(MAXSIZ) / 1024 / 1024/1024) MAX_Size,
 72         to_number('') "USED,% of MAX Size",
 73         to_number('') BLOCK_SIZE,
 74         '' LOGGING,
 75         to_number('') ts_df_count
 76  FROM   wt1 t
 77  order by TS#
 78  ;

       TS# TS_NAME                        TS_SIZE_M FREE_SIZE_M USED_SIZE_M   USED_PER MAX_SIZE_G USED_PER_MAX BLOCK_SIZE
LOGGING   TS_DF_COUNT
---------- ------------------------------ ---------- ----------- ----------- ---------- ---------- ------------ ----------
--------- -----------
         0 SYSTEM                              1110          10        1100     99.116         32        3.358
8192 LOGGING           1
         1 SYSAUX                               510          27         483     94.743         32        1.475
8192 LOGGING           1
         2 UNDOTBS1                             760         222         538     70.765         32        1.641
8192 LOGGING           1
         3 TEMP                                  29          25           4     13.793         32         .012
8192 NOLOGGING         1
         4 USERS                              24688        2032       22655     91.768         32       69.138
8192 LOGGING           1
           ALL TS:                          27096.5        2316       24780     91.453        160

6 rows selected.

SYS@oralhr>
SYS@oralhr> SELECT count(1) FROM dba_free_space;

  COUNT(1)
----------
       254
SYS@oralhr> SELECT count(1) FROM dba_recyclebin;

  COUNT(1)
----------
         0

SYS@oralhr>
SYS@oralhr>
SYS@oralhr> begin
  2    for cur in (SELECT d.table_name
  3                  FROM dba_tables d
  4                 WHERE d.table_name like 'TB_RECYCLEBIN%') loop
  5
  6      execute immediate 'drop table ' || cur.table_name;
  7
  8    end loop;
  9  end;
 10  /

PL/SQL procedure successfully completed.

SYS@oralhr> SELECT count(1) FROM dba_recyclebin;

  COUNT(1)
----------
    239829


SYS@oralhr> select count(1) from dba_free_space;
```

```
select count(1) from dba_free_space
                     *
ERROR at line 1:
ORA-01013: user requested cancel of current operation



SYS@oralhr>
```

回收站里有 239829 条数据，我们查询 dba_free_space 视图很久都不能出结果，接下来只能清空回收站了。

利用 purge dba_recyclebin 命令清理回收站：

```
10:39:50 SYS@oralhr> purge dba_recyclebin;
```

单独开窗口计算：

```
SYS@oralhr> set time on
10:43:44 SYS@oralhr> SELECT count(1) FROM dba_recyclebin;

  COUNT(1)
----------
    234164

10:44:11 SYS@oralhr> SELECT count(1) FROM dba_recyclebin;

  COUNT(1)
----------
    227432



10:48:02 SYS@oralhr>  select (234164-227432)/150 from dual;

(234164-227432)/150
-------------------
              44.88   ====》说明每秒大约删掉 45 条记录

10:48:05 SYS@oralhr> select 227432/45/60 from dual;

227432/45/60
------------
  84.2340741 ====》说明删除 22W 数据大约需要 1 个半小时，太慢了


10:48:28 SYS@oralhr>
```

下边我们采用 job 的形式来删除回收站对象：

```
10:51:28 SYS@oralhr> SELECT D.owner,COUNT(1) FROM dba_recyclebin D GROUP BY D.owner;

OWNER                                                               COUNT(1)
------------------------------------------------------------------- ----------
SYS                                                                    215333

10:53:25 SYS@oralhr> CREATE TABLE XB_recyclebin_LHR NOLOGGING AS
10:53:26   2  SELECT ROWNUM RN, 'PURGE ' || A.type || ' ' || A.owner || '."' || A.object_name || '"'
EXEC_SQL
10:53:26   3    FROM dba_recyclebin A
10:53:26   4   where a.type = 'TABLE';

Table created.
```

```
10:53:41 SYS@oralhr> CREATE INDEX IDX_recyclebin_rn  on XB_recyclebin_LHR(rn) NOLOGGING ;

Index created.

10:53:55 SYS@oralhr> create table XB_SPLIT_JOB_LHR
10:54:05   2  (
10:54:05   3    startrownum NUMBER(18),
10:54:05   4    endrownum   NUMBER(18),
10:54:05   5    flag        NUMBER(1)
10:54:05   6  );

Table created.

10:54:06 SYS@oralhr> SELECT * FROM xb_split_job_lhr;

no rows selected

10:54:12 SYS@oralhr> CREATE OR REPLACE PROCEDURE pro_split_job_lhr AUTHID CURRENT_USER IS
10:54:51   2      ----------------------------------------------------------------------
10:54:51   3      -- copy on 2012/4/2 23:28:21 by lhr
10:54:51   4      --function：该存过用来分隔数据来建立 job
10:54:51   5      --需要进行处理的数据量 ,需要处理的表加 rn 列，值取 rownum，rn 列加索引
10:54:51   6
10:54:51   7      --alter table tmp_dp_idp_lhr add rn number;
10:54:51   8      /* CREATE INDEX IDX_tmp_dp_idp_lhr_rn  on tmp_dp_idp_lhr(rn)
10:54:51   9      TABLESPACE SDH_INDEX ONLINE  NOLOGGING COMPUTE STATISTICS PARALLEL;*/
10:54:51  10
10:54:51  11      /*  create table XB_SPLIT_JOB_LHR
10:54:51  12      (
10:54:51  13        startrownum NUMBER(18),
10:54:51  14        endrownum   NUMBER(18),
10:54:51  15        flag        NUMBER(1)
10:54:51  16      )*/
10:54:51  17      ---------------------------------------------------------------------
10:54:51  18
10:54:51  19      n                NUMBER; --创建的 job 数
10:54:51  20      j                NUMBER := 0;
10:54:51  21      n_startrownum    NUMBER;
10:54:51  22      n_endrownum      NUMBER;
10:54:51  23      n_patchnum       NUMBER := 40000; -- 每批处理的记录数      ----modify
10:54:51  24      v_jobname        VARCHAR2(200);
10:54:51  25      v_count          NUMBER; --需要处理的表的数据量
10:54:51  26
10:54:51  27  BEGIN
10:54:51  28
10:54:51  29      SELECT COUNT(1) INTO v_count FROM XB_recyclebin_LHR; ----modify
10:54:51  30
10:54:51  31      --需要创建的 job 个数
10:54:51  32      n := trunc(v_count / n_patchnum) + 1;
10:54:51  33
10:54:51  34       EXECUTE IMMEDIATE 'truncate table xb_split_job_lhr';
10:54:51  35      WHILE j < n LOOP
10:54:51  36
10:54:51  37          --得到 rownum
10:54:51  38          n_startrownum := j * n_patchnum + 1;
10:54:51  39
10:54:51  40          IF j = n - 1 THEN
10:54:51  41
10:54:51  42              n_endrownum := v_count;
10:54:51  43          ELSE
10:54:51  44              n_endrownum := (j + 1) * n_patchnum;
10:54:51  45          END IF;
10:54:51  46
10:54:51  47          INSERT INTO xb_split_job_lhr
10:54:51  48              (startrownum, endrownum)
10:54:51  49          VALUES
10:54:51  50              (n_startrownum, n_endrownum);
10:54:51  51          COMMIT;
10:54:51  52
```

```
10:54:51  53          j := j + 1;
10:54:51  54      END LOOP;
10:54:51  55
10:54:51  56      --循环创建 job
10:54:51  57      j                := 0;
10:54:51  58
10:54:51  59      FOR cur IN (SELECT * FROM xb_split_job_lhr) LOOP
10:54:51  60
10:54:52  61          v_jobname := 'JOB_SUBJOB_SPLIT_LHR' || (j + 1);
10:54:52  62          dbms_scheduler.create_job(job_name            => v_jobname,
10:54:52  63                                    job_type            => 'STORED_PROCEDURE',
10:54:52  64                                    job_action          => 'PRO_SUB_SPLIT_LHR', --modify
10:54:52  65                                    number_of_arguments => 2,
10:54:52  66                                    start_date          => SYSDATE + 1 / 5760, -- 15 秒后启动作业
10:54:52  67                                    repeat_interval     => NULL,
10:54:52  68                                    end_date            => NULL,
10:54:52  69                                    job_class           => 'DEFAULT_JOB_CLASS',
10:54:52  70                                    enabled             => FALSE,
10:54:52  71                                    auto_drop           => TRUE,
10:54:52  72                                    comments            => 'to split job_subjob_Split_lhr');
10:54:52  73          COMMIT;
10:54:52  74
10:54:52  75          dbms_scheduler.set_job_argument_value(job_name          => v_jobname,
10:54:52  76                                                argument_position => 1,
10:54:52  77                                                argument_value    => cur.startrownum);
10:54:52  78          COMMIT;
10:54:52  79          dbms_scheduler.set_job_argument_value(job_name          => v_jobname,
10:54:52  80                                                argument_position => 2,
10:54:52  81                                                argument_value    => cur.endrownum);
10:54:52  82          COMMIT;
10:54:52  83          dbms_scheduler.enable(v_jobname);
10:54:52  84          j := j + 1;
10:54:52  85      END LOOP;
10:54:52  86      COMMIT;
10:54:52  87
10:54:52  88      -----等待所有的子 job 执行完
10:54:52  89
10:54:52  90      LOOP
10:54:52  91
10:54:52  92          SELECT COUNT(1)
10:54:52  93          INTO   v_count
10:54:52  94          FROM   xb_split_job_lhr t
10:54:52  95          WHERE  t.flag IS NULL;
10:54:52  96
10:54:52  97          IF v_count = 0 THEN
10:54:52  98              EXIT;
10:54:52  99          ELSE
10:54:52 100              dbms_lock.sleep(10); ---存过休息 10 秒
10:54:52 101          END IF;
10:54:52 102
10:54:52 103      END LOOP;
10:54:52 104     EXECUTE IMMEDIATE 'purge dba_recyclebin';
10:54:52 105  EXCEPTION
10:54:52 106      WHEN OTHERS THEN
10:54:52 107         NULL;
10:54:52 108
10:54:52 109  END pro_split_job_lhr;
10:54:54 110  /

Procedure created.

10:55:17 SYS@oralhr> show error
No errors.
10:55:21 SYS@oralhr> create or replace procedure pro_sub_split_lhr(p_startrownum number,
10:55:24   2                                                p_endrownum   number) is
10:55:24   3
10:55:24   4  begin
10:55:24   5
10:55:24   6    for cur in (SELECT A.EXEC_SQL
10:55:24   7                  FROM XB_recyclebin_LHR A ---modify
10:55:24   8                  where A.rn <= p_endrownum
10:55:24   9                    and A.rn >= p_startrownum) loop
10:55:24  10       begin
10:55:24  11         EXECUTE IMMEDIATE CUR.EXEC_SQL;
10:55:24  12       exception
10:55:24  13         when others then
10:55:24  14           null;
10:55:24  15       end;
10:55:24  16     end loop;
10:55:24  17
10:55:24  18     commit;
10:55:24  19
10:55:24  20     --更新标志
10:55:24  21     update xb_split_job_lhr t
10:55:24  22        set t.flag = 1
10:55:24  23      where t.startrownum = p_startrownum
10:55:24  24        and t.endrownum = p_endrownum;
10:55:24  25     commit;
10:55:24  26
10:55:24  27   exception
10:55:24  28
10:55:24  29     when others then
10:55:24  30
10:55:24  31       null;
10:55:24  32
10:55:24  33   end pro_sub_split_lhr;
10:55:25  34   /

Procedure created.

10:55:26 SYS@oralhr> show error
No errors.
10:55:29 SYS@oralhr> exec pro_split_job_lhr;
```

单独开窗口重新计算清空回收站的速度：

```
SYS@oralhr> set time on
11:04:38 SYS@oralhr> SELECT count(1) FROM dba_recyclebin;

  COUNT(1)
----------
    211055

11:06:00 SYS@oralhr> SELECT count(1) FROM dba_recyclebin;

  COUNT(1)
----------
    189105

11:08:00 SYS@oralhr>  select (211055-189105)/80 from dual;

(234164-227432)/150
-------------------
              274.375   ====》说明每秒大约删掉 275 条记录

11:08:10 SYS@oralhr> select 189105/275/60 from dual;

189105/275/60
-------------
   11.4609091 ====》说明删除 18W 数据大约需要 11 分钟

11:09:10 SYS@oralhr>
```

等待十几分钟后查看数据：

```
SYS@oralhr> select * from xb_split_job_lhr;
```

```
STARTROWNUM  ENDROWNUM       FLAG
----------- ---------- ----------
          1      40000
      40001      80000
      80001     120000
     120001     159915

SYS@oralhr>
SYS@oralhr> col owner for a5
SYS@oralhr> col CPU_USED for a18
SYS@oralhr> col ELAPSED_TIME for a18
SYS@oralhr> select OWNER,JOB_NAME,CPU_USED,ELAPSED_TIME,RUNNING_INSTANCE from
dba_scheduler_running_jobs;

OWNER JOB_NAME                       CPU_USED           ELAPSED_TIME       RUNNING_INSTANCE
----- ------------------------------ ------------------ ------------------ ----------------
SYS   JOB_SUBJOB_SPLIT_LHR1          +000 00:10:18.36   +000 00:19:15.29                  1
SYS   JOB_SUBJOB_SPLIT_LHR2          +000 00:10:14.71   +000 00:19:15.07                  1
SYS   JOB_SUBJOB_SPLIT_LHR3          +000 00:10:12.77   +000 00:19:14.95                  1
SYS   JOB_SUBJOB_SPLIT_LHR4          +000 00:10:14.70   +000 00:19:14.78                  1

SYS@oralhr>
```

若系统 CPU 强劲的话，该 SQL 会很快完成的，查询 dba_scheduler_running_jobs 视图将无数据表示 job 已完成。

2.4 实验总结

1、11.2.0.4 中若回收站对象过多的情况下，dba_free_space 查询过慢的问题已经解决了

2、实验二的脚本具有通用性，很多操作可以同时执行的时候我们可以修改该程序

第 3 章　实验中用到的 SQL 总结

实验一：

```
dbca -silent -createDatabase -templateName General_Purpose.dbc -responseFile NO_VALUE \
-gdbname oralhr  -sid oralhr \
-sysPassword oracle -systemPassword lhr \
-datafileDestination 'DATA2/' -recoveryAreaDestination 'DATA2/' \
-redoLogFileSize 50 \
-storageType ASM -asmsnmpPassword lhr  -diskGroupName 'DATA2' \
-characterset AL32UTF8 -nationalCharacterSet AL16UTF16 \
-sampleSchema false \
-automaticMemoryManagement true -totalMemory 2048 \
-databaseType OLTP  \
-emConfiguration NONE

show parameter recy
create table tb_20160627_lhr as select * from dual;
drop table tb_20160627_lhr;
select * from dba_recyclebin;
drop table tb_20160627_lhr;
drop table tb_20160627_lhr;
select * from dba_recyclebin;
```

实验二：

```
begin

  for cur in 1 .. 100000 loop

    execute immediate 'create table tb_recyclebin_' || cur ||
                      ' nologging tablespace users as select * from dual';
    execute immediate 'create index idx_recyclebin_' || cur ||
                      ' on tb_recyclebin_' || cur ||' (dummy) nologging tablespace users';

  end loop;
end;
/

begin

  for cur in 1 .. 100000 loop

    execute immediate 'create table tb_recyclebin_lhr_' || cur ||
                      ' nologging tablespace users as select * from dual';
    execute immediate 'create index idx_recyclebin_lhr_' || cur ||
                      ' on tb_recyclebin_lhr_' || cur ||' (dummy) nologging tablespace users';

  end loop;

end;
/

begin

  for cur in 1 .. 100000 loop

    execute immediate 'create table tb_recyclebin_lhr1_' || cur ||
                      ' nologging tablespace users as select * from dual';
    execute immediate 'create index idx_recyclebin_lhr1_' || cur ||
                      ' on tb_recyclebin_lhr1_' || cur ||' (dummy) nologging tablespace users';

  end loop;

end;

begin
  for cur in (SELECT d.table_name
              FROM dba_tables d
             WHERE d.table_name like 'TB_RECYCLEBIN%') loop
    execute immediate 'drop table ' || cur.table_name;
  end loop;
end;
/


CREATE TABLE XB_recyclebin_LHR NOLOGGING AS
SELECT ROWNUM RN, 'PURGE ' || A.type || ' ' || A.owner || '."' || A.object_name || '"' EXEC_SQL
  FROM dba_recyclebin A
 where a.type = 'TABLE';
```

```
CREATE INDEX IDX_recyclebin_rn  on XB_recyclebin_LHR(rn) NOLOGGING ;

create table XB_SPLIT_JOB_LHR
(
  startrownum NUMBER(18),
  endrownum   NUMBER(18),
  flag        NUMBER(1)
);

col CPU_USED for a18
col ELAPSED_TIME for a18
select OWNER,JOB_NAME,CPU_USED,ELAPSED_TIME,RUNNING_INSTANCE from dba_scheduler_running_jobs;
```

About Me

....

本文作者：小麦苗，只专注于数据库的技术，更注重技术的运用

本文在 ITpub（http://blog.itpub.net/26736162）和博客园(http://www.cnblogs.com/lhrbest)有同步更新

本文地址：http://blog.itpub.net/26736162/viewspace-2121136/和

http://blog.itpub.net/26736162/viewspace-2121137/

本文 pdf 版：http://yunpan.cn/cdEQedhCs2kFz （提取码：ed9b）

小麦苗分享的其它资料：http://blog.itpub.net/26736162/viewspace-1624453/

联系我请加 QQ 好友(642808185)，注明添加缘由

于 2016-06-24 10:00~ 2016-06-27 19:00 在中行完成

【版权所有，文章允许转载，但须以链接方式注明源地址，否则追究法律责任】

....