```
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
```

使用 BBED，通过 SYSTEM 表空间作为参照物，修改离线数据文件的检查点信息等：

```
[oracle@hpserver2 lib]$ bbed parfile=par.txt
Password:

BBED: Release 2.0.0.0.0 - Limited Production on Mon Jan 16 17:29:06 2012

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

************* !!! For Oracle Internal Use only !!! ***************
BBED> p kcvfhckp file 1
struct kcvfhckp, 36 bytes                    @484
   struct kcvcpscn, 8 bytes                  @484
      ub4 kscnbas                            @484      0x00416bb3 <  当前检查点信息
      ub2 kscnwrp                            @488      0x0000
   ub4 kcvcptim                              @492      0x2e0f0c15 <  当前检查点时间
   ub2 kcvcpthr                              @496      0x0001
   union u, 12 bytes                         @500
      struct kcvcprba, 12 bytes              @500
         ub4 kcrbaseq                        @500      0x000000d0
         ub4 kcrbabno                        @504      0x00000002
         ub2 kcrbabof                        @508      0x0010
   ub1 kcvcpetb[0]                           @512      0x02
   ub1 kcvcpetb[1]                           @513      0x00
   ub1 kcvcpetb[2]                           @514      0x00
   ub1 kcvcpetb[3]                           @515      0x00
   ub1 kcvcpetb[4]                           @516      0x00
   ub1 kcvcpetb[5]                           @517      0x00
   ub1 kcvcpetb[6]                           @518      0x00
   ub1 kcvcpetb[7]                           @519      0x00
BBED> p kcvfhckp file 7
```

```
struct kcvfhckp, 36 bytes                    @484
   struct kcvcpscn, 8 bytes                  @484
      ub4 kscnbas                            @484      0x00415a2f < 文件目前的检查点
      ub2 kscnwrp                            @488      0x0000
   ub4 kcvcptim                              @492      0x2e0f01b8
   ub2 kcvcpthr                              @496      0x0001
   union u, 12 bytes                         @500
      struct kcvcprba, 12 bytes              @500
         ub4 kcrbaseq                        @500      0x000000cc
         ub4 kcrbabno                        @504      0x00000374
         ub2 kcrbabof                        @508      0x0010
   ub1 kcvcpetb[0]                           @512      0x02
   ub1 kcvcpetb[1]                           @513      0x00
   ub1 kcvcpetb[2]                           @514      0x00
   ub1 kcvcpetb[3]                           @515      0x00
   ub1 kcvcpetb[4]                           @516      0x00
   ub1 kcvcpetb[5]                           @517      0x00
   ub1 kcvcpetb[6]                           @518      0x00
   ub1 kcvcpetb[7]                           @519      0x00
BBED> set count 8
    COUNT            8
BBED> dump
 File: /u01/app/oracle/oradata/ORCL/eygle01.dbf (7)
 Block: 1               Offsets:  484 to  491          Dba:0x01c00001
------------------------------------------------------------------------
 2f5a4100 00000000

 <32 bytes per line>
```

以下将文件的检查点修改为 SYSTEM 表空间的检查点：

```
BBED> modify /x b36b file 7 block 1 offset 484 < 修改为 SYSTEM 表空间的检查点
 File: /u01/app/oracle/oradata/ORCL/eygle01.dbf (7)
 Block: 1               Offsets:  484 to  491          Dba:0x01c00001
------------------------------------------------------------------------
```

```
 b36b4100 00000000

 <32 bytes per line>
```

以下命令修改检查点时间：

```
BBED> modify /x 150c0f2e file 7 block 1 offset 492 < 修改检查点时间
 File: /u01/app/oracle/oradata/ORCL/eygle01.dbf (7)
 Block: 1                Offsets:  492 to  499           Dba:0x01c00001
------------------------------------------------------------------------
 150c0f2e 01000000

 <32 bytes per line>
```

以下偏移量 500 为文件的 RBA 信息，修改该信息与 SYSTEM 表空间一致：

```
BBED> modify /x d0 file 7 block 1 offset 500
 File: /u01/app/oracle/oradata/ORCL/eygle01.dbf (7)
 Block: 1                Offsets:  500 to  507           Dba:0x01c00001
------------------------------------------------------------------------
 d0000000 74030000

 <32 bytes per line>
BBED> modify /x 7403 file 7 block 1 offset 504
 File: /u01/app/oracle/oradata/ORCL/eygle01.dbf (7)
 Block: 1                Offsets:  504 to  511           Dba:0x01c00001
------------------------------------------------------------------------
 74030000 1000fab8

 <32 bytes per line>
```

修改完成之后，可以通过 SUM 重新计算并应用新的校验值，否则数据块会被标记为损坏：

```
BBED> sum apply file 7
Check value for File 7, Block 1:
current = 0x6df9, required = 0x6df9
BBED> verify file 7
```

```
DBVERIFY - Verification starting
FILE = /u01/app/oracle/oradata/ORCL/eygle01.dbf


DBVERIFY - Verification complete

Total Blocks Examined         : 1280
Total Blocks Processed (Data) : 697
Total Blocks Failing   (Data) : 0
Total Blocks Processed (Index): 0
Total Blocks Failing   (Index): 0
Total Blocks Empty            : 575
Total Blocks Marked Corrupt   : 0
Total Blocks Influx           : 0
```

最后通过简单的 Recover 即可将文件 Online 加载：

```
[oracle@hpserver2 lib]$ sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.4.0 - Production on Mon Jan 16 17:33:28 2012
Copyright (c) 1982, 2007, Oracle.  All Rights Reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> recover datafile 7;
Media recovery complete.
SQL> alter database datafile 7 online;
Database altered.
```

这也是这类故障恢复的重要方法之一。

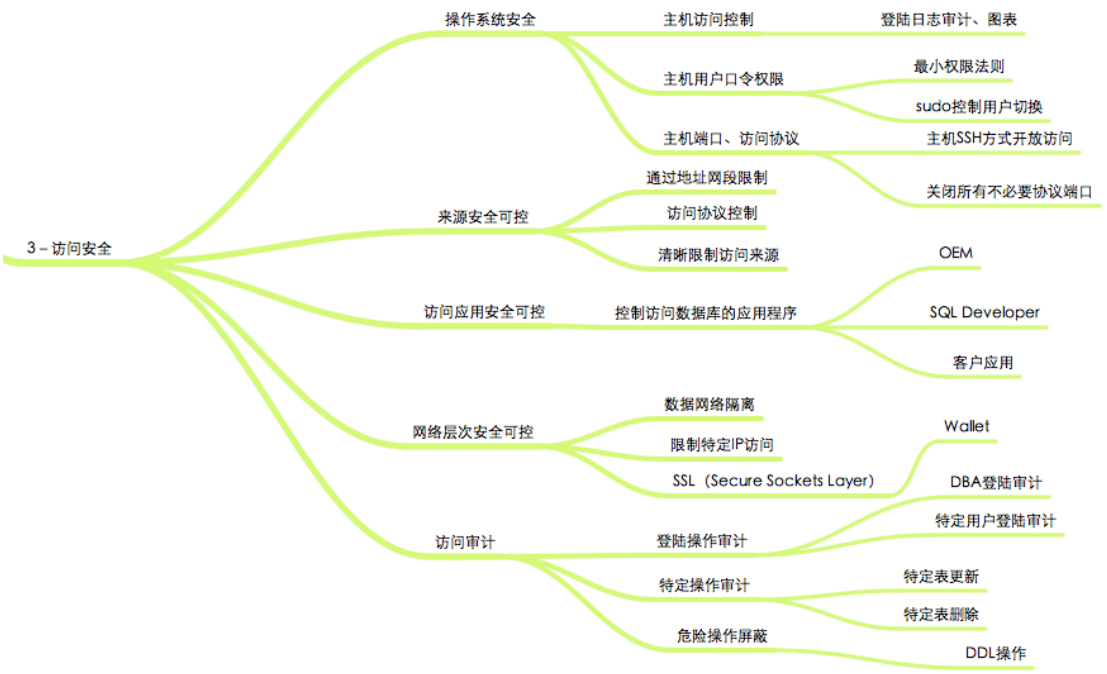
心存目想，三思后行

杨君缄书赍图请予为记。予按图握笔，心存目想，飜缕梗概，十不得其二三。

——白居易《白苹洲五亭记》

季文子三思而后行。子闻之曰：“再，斯可矣。”

——《论语·公冶长》



我们遇到的很多数据灾难，其产生过程往往就在一念之间，有时候眼睛看到的，未作认真思考，下意识的一个动作就使数据库陷入苦难。

所以当我们进行数据库的维护工作时，应当将目之所见，经过心之所想，再转化为手之所动，三思而后行，这样才能够减少灾难的发生。

本章涉及的案例与访问安全有关，随意的、无规范的管理常常导致数据灾难，很多用户在图形工具远程管理之下，严重破坏了访问安全守则，PL/SQL Developer 或 OEM 在很多情况下导致了严重的数据库故障，简便易用必然导致安全的牺牲。

篇首图对访问安全的诸多方面进行了分析，从操作系统安全，到应用访问来源等等，通常我们应当能够连接数据库的程序，很多工具是危险的：

控制访问安全，是实现数据安全的核心环节。

Truncate 导致的灾难

核心字典表误操作 TRUNCATE

灾难描述

以下这则案例就发生在一念之间，手之所动未经心之所想。

1. 客户 DBA 使用 PL/SQL Develper 之类工具管理数据库
2. 发现几个表占用的空间很大
3. 这几个表不是业务数据表
4. DBA 随手 Truncate 截断了这几个数据表
5. 数据库告警报错，运行不再正常
6. 检查发现这几张表是数据库的字典表
7. 灾难形成

回顾这个灾难的形成过程，我们可以想象，如果 DBA 经过多几分钟的思考，这个故障就不会发生，一念之间，天差地别。

案例警示

这个案例给我们的警示是：

1. **任何业务数据库的操作都不能草率**

这个数据库是非常重要的一个业务数据库，DBA 这样的操作是非常草率的，我们警示任何数据库的维护人员，在接触数据时都不能掉以轻心，草率行事。

在权限管理上，应当做到权限分离，作为业务维护人员不应当授予 DBA 权限，而 DBA 在进行非维护操作时，也无需以 SYSDBA 角色进行登陆。

DBA 常规的维护工作可以用 SYSTEM 账户登陆数据库，这个账户会屏蔽掉一些危险的操作，防止破坏性

灾难的发生。

2. 眼之所见必须要经过心之所想才能转换为手之所做

这次故障实际上是一时轻率酿成的，眼之所见未经过用心思考，就转换为手之动作，动作一发出可能 DBA 就已经后悔了，但是 DDL 语句无法回退，没有退路可选。

所以我们警示 DBA，在执行任何维护操作时都应当三思后行，慎之再慎。

3. 数据库维护操作尽量以命令行方式完成

对于数据库的维护操作，应当在数据库本地，以命令行方式完成，远程的图形终端操作是不可靠、不安全的，应当严格杜绝远程的维护操作。

图形界面中的一个功能描述，可能很多 DBA 无法直观的将其对应到后台命令上去，这就导致了麻痹疏忽，如果主动发出 Truncate 的命令，我相信绝大多数 DBA 都会多想一想。

通过触发器可以实现 DDL 本地化限制，这可以参考第二章中的内容。

4. 养成文档和操作记录的习惯

在进行数据库维护操作工作时，应当进行事前的文档记录和准备工作，然后再具体执行，即便是简单的工作，用铅笔在纸上做一些条目梳理也有助于帮助我们理清思路、延缓操作、减少失误。

5. 任何关键性的修改应两个人以上确认

任何人都有思维不清晰的时刻，而两个人同时犯一个低级错误的可能性就要小得多。因此建议执行关键性操作或执行有潜在风险的操作之前，一定要找同事一起确认脚本或命令，这样可以最大程度降低由于头脑一时不清所引发的灾难。如果身边没有熟悉数据库的人员，可以常识将自己的步骤和操作讲给其他人听，如果你的步骤中存在明显的错误，很可能在你讲解的过程中自己就发现了。

规范、计划、记录、思考，这些习惯在 DBA 的工作中应当时刻记在心头。

技术回放

这个案例可以概括为由于 DBA 的轻率误操作，导致部分数据字典信息丢失，影响了数据库的稳定性和一致性。

IDL_CHAR$等字典被截断

在现场经过检查确定，这些被 Truncate 的表主要有：

```
SQL> select object_name,object_type from dba_objects where object_name like 'IDL%';
OBJECT_NAME                    OBJECT_TYPE
------------------------- ------------------
IDL_CHAR$                      TABLE
IDL_SB4$                       TABLE
IDL_UB1$                       TABLE
IDL_UB2$                       TABLE
```

这些字典表用于存放 Oracle 数据库的存储过程、触发器、Package 编译后的执行信息，缺少这些信息，数据库的任何 DML 操作都无法进行。对于一个存在大量过程、触发器等的数据库来说，这些字典表的存储可能越来越大，在有些用户的数据库中，这些字典表可能占用几个 GB 到 10 几个 GB 的容量。

这几个字典表的定义可以从 sql.bsq 及相关文件中找到：

```
create table idl_ub1$                          /* idl table for ub1 pieces */
( obj#          number not null,                        /* object number */
  part          number not null,
        /* part: 0 = diana, 1 = portable pcode, 2 = machine-dependent pcode */
  version       number,                                  /* version number */
  piece#        number not null,                         /* piece number */
  length        number not null,                         /* piece length */
  piece         long raw not null)                         /* ub1 piece */
  storage (initial 10k next 100k maxextents unlimited pctincrease 0)
/
create table idl_char$                         /* idl table for char pieces */
( obj#          number not null,                        /* object number */
  part          number not null,
        /* part: 0 = diana, 1 = portable pcode, 2 = machine-dependent pcode */
  version       number,                                  /* version number */
  piece#        number not null,                         /* piece number */
  length        number not null,                         /* piece length */
  piece         long not null)                             /* char piece */
```

```
    storage (initial 10k next 100k maxextents unlimited pctincrease 0)
/
create table idl_ub2$                                   /* idl table for ub2 pieces */
( obj#              number not null,                                /* object number */
    part            number not null,
                /* part: 0 = diana, 1 = portable pcode, 2 = machine-dependent pcode */
    version         number,                                          /* version number */
    piece#          number not null,                                  /* piece number */
    length          number not null,                                 /* piece length */
    piece           long ub2 not null)                                  /* ub2 piece */
    storage (initial 10k next 100k maxextents unlimited pctincrease 0)
/
create table idl_sb4$                                   /* idl table for sb4 pieces */
( obj#              number not null,                                /* object number */
    part            number not null,
                /* part: 0 = diana, 1 = portable pcode, 2 = machine-dependent pcode */
    version         number,                                          /* version number */
    piece#          number not null,                                  /* piece number */
    length          number not null,                                 /* piece length */
    piece           long sb4 not null)                                  /* sb4 piece */
    storage (initial 10k next 100k maxextents unlimited pctincrease 0)
/
```

这几个表被 Truncate 之后又生成了少量记录，这些信息来自于用户的一些编译尝试：

```
SQL> select count(*) from idl_ub1$;
  COUNT(*)
----------
        33
SQL> select count(*) from idl_ub2$;
  COUNT(*)
----------
        66
SQL> select count(*) from idl_sb4$;
  COUNT(*)
```

```
----------
        66
SQL> select count(*) from idl_char$;
  COUNT(*)
----------
        33
```

ORA-600 17069 导致故障

此时数据库出现的典型错误是 ORA-600 错误：

```
Mon Jan 21 16:10:07 2008
Errors in file /ora_arch/admin/bdump/bgcw2_j000_20123.trc:
ORA-00600: internal error code, arguments: [17069], [0xC0000000B2E4A750], [], [], [],
[], [], []
Mon Jan 21 16:10:08 2008
Errors in file /ora_arch/admin/bdump/bgcw2_j000_20123.trc:
ORA-00600: internal error code, arguments: [17069], [0xC0000000B2E4A750], [], [], [],
[], [], []
Mon Jan 21 16:10:08 2008
Trace dumping is performing id=[cdmp_20080121161008]
Mon Jan 21 16:11:05 2008
Errors in file /ora_arch/admin/bdump/bgcw2_j001_20125.trc:
ORA-00600: internal error code, arguments: [17069], [0xC0000000B2E4A750], [], [], [],
[], [], []
```

由于这一故障的出现，数据库无法正常使用，所有的过程调用都会出现 ORA-600 错误。

ORA-600 17069 错误，其含义是进程无法 Pin 锁定某个 library cache object,这个问题通常与过程的失效或删除有关，通常通过重新编译过程等对象就可以解决。

只不过对于这个案例，是用户手工截断了相关的字典表，一些尝试编译工作也失败了。进程跟踪文件如下：

```
*** SESSION ID:(36.7) 2008-01-20 12:32:27.357
*** 2008-01-20 12:32:27.357
ksedmp: internal or fatal error
```

```
ORA-00600: internal error code, arguments: [17069], [0xC0000000DDDFA690], [], [], [],
[], [], []
No current SQL statement being executed.
----- Call Stack Trace -----
calling              call     entry                argument values in hex
location             type     point                (? means dubious value)
-------------------- -------- -------------------- ----------------------------
ksedmp()+528         call     _etext_f()+23058430  000000000 ?
                              09102745972          C000000000000996 ?
                                                   4000000002801BE0 ?
ksfdmp()+64          call     _etext_f()+23058430  000000003 ?
                              09102745972
kgeriv()+432         call     _etext_f()+23058430  600000000004A0F0 ?
                              09102745972          000000003 ?
                                                   C000000000000716 ?
                                                   40000000052BAFB0 ?
                                                   FF9C00000001802F ?
                                                   60000000004DCC48 ?
                                                   000000000 ? 000000000 ?
```

进程状态信息显示该进程正在等待 Library cache pin,请求对象的地址为 c0000000dcd182e0：

```
    ----------------------------------------
    SO: c0000000d62e1730, type: 4, owner: c0000000d6298f90, flag: INIT/-/-/0x00
    (session) trans: 0000000000000000, creator: c0000000d6298f90, flag: (8000041)
USR/- BSY/-/-/-/-/-
              DID: 0001-0010-00000030, short-term DID: 0000-0000-00000000
              txn branch: 0000000000000000
              oct: 0, prv: 0, sql: c0000000ddcb7398, psql: 0000000000000000, user:
39/FMIS_SHARE
    O/S info: user: , term: , ospid: , machine:
              program:
    last wait for 'library cache pin' blocking sess=0x0 seq=53 wait_time=44
                handle address=c0000000dddfa690, pin address=c0000000dcd182e0,
100*mode+namespace=12d
```

```
    temporary object counter: 0
      ---------------------------------------
```

在进程的授权队列上，我们可以找到详细的依赖信息，以下是进程的主要信息：

```
    ------------process---------------------
proc version      : 3
Local node        : 0
gid               : 0
pid               : 5675
lkp_node          : 0
proc state        : KJP_NORMAL
Total accesses    : 684
Imm.  accesses    : 683
Locks on ASTQ     : 0
Locks Pending AST : 0
Granted locks     : 1
AST_Q     :NULL
PENDING_Q :NULL
GRANTED_Q :0xc0000000d68e75b8
    AST_Q:
    PENDING_Q:
    GRANTED_Q:
    lp 0xc0000000d68e75b8 gl KJUSERPR rp 0xc0000000d680b3c0 master 0 pid 5675
      bast 0 rseq 0 history 0x14351435 open opt KJUSERDEADLOCK KJUSERPROCESS_OWNED
```

接下来的依赖关系显示 c0000000dcd182e0 请求 Pin 的对象是 SYS.STANDARD 标准包，这是数据库最核心的一个系统包，该 Package 失效，大量依赖它的对象全部失效，系统的症结首先体现在这里：

```
----------------------------------------
SO: c0000000d633e198, type: 3, owner: c0000000d6298f90, flag: INIT/-/-/0x00
(call) sess: cur c0000000d62e1730, rec c0000000d62e1730, usr c0000000d62e1730; depth: 0
  ----------------------------------------
  SO: c0000000d633c8a8, type: 3, owner: c0000000d633e198, flag: INIT/-/-/0x00
  (call) sess: cur c0000000d62e1730, rec 0, usr c0000000d62e1730; depth: 1
    ----------------------------------------
    SO: c0000000dcd182e0, type: 52, owner: c0000000d633c8a8, flag: INIT/-/-/0x00
    LIBRARY OBJECT PIN: pin=c0000000dcd182e0 handle=c0000000dddfa690 mode=S lock=c0000000dcd17550
    user=c0000000d62e1730 session=c0000000d62e1730 count=1 mask=001d savepoint=230 flags=[00]
    ----------------------------------------
    SO: c0000000dcd17550, type: 51, owner: c0000000d633c8a8, flag: INIT/-/-/0x00
    LIBRARY OBJECT LOCK: lock=c0000000dcd17550 handle=c0000000dddfa690 mode=S
    call pin=c0000000dcd182e0 session pin=0000000000000000
    htl=c0000000dcd175c0[c0000000dcd79a88,c0000000dcd79a88] htb=c0000000dcd79a88
    user=c0000000d62e1730 session=c0000000d62e1730 count=1 flags=PNC/[04] savepoint=230
    LIBRARY OBJECT HANDLE: handle=c0000000dddfa690
    name=SYS.STANDARD
    hash=bf7b5f21 timestamp=11-19-2001 00:00:00
    namespace=TABL/PRCD/TYPE flags=KGHP/TIM/SML/[02000000]
    kkkk-dddd-llll=0000-001d-001d lock=S pin=S latch#=1
    lwt=c0000000dddfa6c0[c0000000dddfa6c0,c0000000dddfa6c0] ltm=c0000000dddfa6d0[c0000000dddfa6d0,c0000000dddfa6d0]
    pwt=c0000000dddfa6f0[c0000000dddfa6f0,c0000000dddfa6f0] ptm=c0000000dddfa780[c0000000dddfa780,c0000000dddfa780]
    ref=c0000000dddfa6a0[c0000000dddfa6a0,c0000000dddfa6a0] lnd=c0000000dddfa798[c0000000ddde5360,c0000000dddf93e0]
      LOCK INSTANCE LOCK: id=LB238ccb5b1667ba4a
      PIN INSTANCE LOCK: id=NB238ccb5b1667ba4a mode=S release=F flags=[00]
      INVALIDATION INSTANCE LOCK: id=IV00000028713010101 mode=S
      LIBRARY OBJECT: object=c0000000ddca9ef0
      type=PCKG flags=EXS/LOC[0005] pflags=NST/IVR [201] status=VALD load=0
      DATA BLOCKS:
      data#     heap  pointer status pins change
      ----- -------- -------- ------ ---- ------
          0 c0000000ddcaa208 c0000000ddca9c00 I/P/A     0 NONE
          2 c0000000ddcaa010        0 -/P/-     1 NONE
          3 c0000000ddcaa0c0        0 -/P/-     1 NONE
          4 c0000000ddca9b78        0 -/P/-     1 NONE
    ----------------------------------------
```

恢复过程

解决这个问题，我们首先想到的思路是：

主要损坏的字典表存储的信息来自对象编译过程，那么通过重新编译所有对象可以重新生成这些数据，从而使数据库恢复正常运行，基于这一考虑，我们可以将主要对象的创建脚本进行再次运行，重新编译数据库相关的 Procedure/Trigger/Package/Package Body 等，从而解决这一故障。

这是一个 RAC 集群数据库，首先应当设置 cluster_database 参数为 false：

```
SQL> alter system set cluster_database=false scope=spfile sid='bgcw2';
System altered.
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

然后将数据库启动在 Migrate 模式执行编译：

```
SQL> startup migrate
ORACLE instance started.

Total System Global Area 2422165840 bytes
Fixed Size                  729424 bytes
Variable Size             738197504 bytes
Database Buffers         1677721600 bytes
Redo Buffers                5517312 bytes
Database mounted.
Database opened.
SQL>@?/rdbms/admin/catalog
SQL>@?/rdbms/admin/catproc
SQL>@?/rdbms/admin/utlrp
```

具体执行过程如下，首先执行 catalog.sql 脚本：

```
SQL> @?/rdbms/admin/catalog
DOC>#########################################################################
DOC>#########################################################################
DOC>    The following statement will cause an "ORA-01722: invalid number"
DOC>    error and terminate the SQLPLUS session if the user is not SYS.
DOC>    Disconnect and reconnect with AS SYSDBA.
DOC>#########################################################################
DOC>#########################################################################
DOC>#

no rows selected

Package created.

Package body created.

Grant succeeded.
```

```
Package created.

Synonym created.

Grant succeeded.

Table created.
.....
```

然后可以执行 catproc.sql 脚本：

```
SQL> @?/rdbms/admin/catproc
DOC>##############################################################################
DOC>##############################################################################
DOC>    The following PL/SQL block will cause an ORA-20000 error and
DOC>    terminate the current SQLPLUS session if the user is not SYS.
DOC>    Disconnect and reconnect with AS SYSDBA.
DOC>##############################################################################
DOC>##############################################################################
DOC>#

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

View created.

View created.

Comment created.

Comment created.

Comment created.
```

```
Comment created.

Comment created.

Comment created.

Comment created.

Synonym created.

Grant succeeded.

View created.

Comment created.
.....
```

完成这两个步骤之后，可以通过运行 utlrp.sql 脚本重新编译失效对象完成恢复。

恢复完成之后，所有相关数据被成功恢复过来：

```
SQL> select count(*) from idl_ub1$;
  COUNT(*)
----------
     50947
SQL> select count(*) from idl_ub2$;
  COUNT(*)
----------
     50719
SQL> select count(*) from idl_sb4$;
  COUNT(*)
----------
     61987
SQL> select count(*) from idl_char$;
```

```
  COUNT(*)
----------
     24804
```

检查数据库的运行日志报告，其中信息输出正常，RAC Cluster 能够正常构建，数据库恢复了正常运行：

```
Mon Jan 21 21:17:58 2008
Successful mount of redo thread 2, with mount id 2791344706
Mon Jan 21 21:17:58 2008
Database mounted in Shared Mode (CLUSTER_DATABASE=TRUE).
Completed: ALTER DATABASE   MOUNT
Mon Jan 21 21:17:58 2008
ALTER DATABASE OPEN
This instance was first to open
Picked Lamport scheme to generate SCNs
Mon Jan 21 21:17:58 2008
Thread 2 opened at log sequence 29826
  Current log# 6 seq# 29826 mem# 0: /dev/vgoracle/rora_redo23a
Successful open of redo thread 2
Mon Jan 21 21:17:58 2008
SMON: enabling cache recovery
Instance recovery: looking for dead threads
Instance recovery: lock domain invalid but no dead threads
Mon Jan 21 21:17:59 2008
Successfully onlined Undo Tablespace 6.
Mon Jan 21 21:17:59 2008
SMON: enabling tx recovery
Mon Jan 21 21:17:59 2008
Database Characterset is ZHS16GBK
Mon Jan 21 21:18:28 2008
replication_dependency_tracking turned off (no async multimaster replication found)
Completed: ALTER DATABASE OPEN
```

新的节点加入，RAC 环境重新配置：

```
Mon Jan 21 21:19:29 2008
```

```
Reconfiguration started (old inc 1, new inc 2)
List of nodes:
 0 1
 Global Resource Directory frozen
 Communication channels reestablished
 Master broadcasted resource hash value bitmaps
 Non-local Process blocks cleaned out
 Resources and enqueues cleaned out
 Resources remastered 1517
 191750 GCS shadows traversed, 0 cancelled, 658 closed
 191092 GCS resources traversed, 0 cancelled
 126678 GCS resources on freelist, 210430 on array, 210430 allocated
 set master node info
 Submitted all remote-enqueue requests
 Update rdomain variables
 Dwn-cvts replayed, VALBLKs dubious
 All grantable enqueues granted
 191750 GCS shadows traversed, 107340 replayed, 658 unopened
 Submitted all GCS remote-cache requests
 0 write requests issued in 83752 GCS resources
 0 PIs marked suspect, 0 flush PI msgs
Mon Jan 21 21:19:31 2008
Reconfiguration complete
Mon Jan 21 21:19:31 2008
Instance recovery: looking for dead threads
Instance recovery: lock domain invalid but no dead threads
Mon Jan 21 21:26:30 2008
Thread 2 advanced to log sequence 29827
  Current log# 4 seq# 29827 mem# 0: /dev/vgoracle/rora_redo21a
```

经业务系统正常运行测试通过，数据库恢复正常，故障处理完成。

脚本错误导致的灾难

数据库整体被删除故障

以下这则灾难初始是由于脚本问题，后续的发展则是由于工程师的判断失误，导致事件逐步恶化，最后不可收拾。

灾难描述

这次灾难的出现过程如下：

1. 客户的备份和数据库位于同一主机的不同卷
2. 备份策略是每日执行备份，压缩传递至远程后，删除本地备份
3. 某日脚本出错，带入的环境变量不是备份目录，而是主机数据库目录
4. 再次误执行脚本，本地备份也被删除
5. 数据库所有文件及本地备份被删除
6. DBA 尝试传输远程到本地恢复
7. 当解压后恢复时，发现备份的文件不足
8. 灾难形成

在这些处理过程中，DBA 为了遮掩自己的失误，一直试图弱化故障的严重程度，直至最后发现备份不足以用来恢复，才将事情如实向上汇报，但是这已经造成了不可挽回的损失。

特别是当本地目录已经被解压覆盖，使得基于文件系统的恢复也不再可能。

案例警示

这个案例与之前很多案例具有类似之处，部分警示前面已经描述过，此处不再赘述。有一些新的警示是：

1. **解决问题最好的办法是诚实，不要隐瞒问题**

当系统因为种种原因遭遇故障后，最好的解决途径是通过集体的力量，个人在极端情况下，思维容易走入

死胡同，不断做出错误的判断。

所以，当出现故障或误操作时，最重要的是不要隐瞒问题，以解决问题为核心，发动团队的智慧。另外，向上级诚实反应情况，也有助于更准确的判断问题的严重程度，以及对外的应对方式。隐瞒和草率的掩盖往往使事情变得更加糟糕。

在这个案例中,DBA 在原目录下解压缩备份文件,导致磁盘存储被覆盖,否则原有文件是可以恢复出来的。

2. 在出现问题时，不要急于做出第一个判断

最为初始的判断会决定后续恢复的走向，如果处置不当，则可能使问题恶化。如果是 DBA 个人的判断，很有可能走向草率的极端，错误叠加错误，所以当问题出现时，不要急于做第一个判断和举措，除非具有十足的把握。

依赖团队和专业人士，有时候可能一个电话就可以解决问题。对于这个案例，也许从文件描述符就可以恢复问题，至不济还可以从磁盘恢复，但是最后覆盖的结果是，数据彻底损失。

这个案例的数据库是一个公司几百员工，几年的工作积累，如果丢失，则损失极为惨痛。

技术回放

在数据库文件被删除之后，告警日志中首先出现错误：

```
Wed Apr  7 11:39:46 2010
Thread 1 advanced to log sequence 20265
  Current log# 7 seq# 20265 mem# 0: /opt/oracle/oradata/MEDIA/redo07.log
Wed Apr  7 11:39:46 2010
ARC0: Beginning to archive log# 6 seq# 20264
ARC0: Completed archiving log# 6 seq# 20264
Wed Apr  7 12:24:24 2010
Errors in file /opt/oracle/admin/MEDIA/udump/media_ora_12238.trc:
ORA-01115: IO error reading block from file 12 (block # 901174)
ORA-01110: data file 12: '/opt/oracle/oradata/MEDIA/media_04.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
```

```
ORA-01115: IO error reading block from file 12 (block # 856947)
ORA-01110: data file 12: '/opt/oracle/oradata/MEDIA/media_04.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
Wed Apr  7 12:25:19 2010
Errors in file /opt/oracle/admin/MEDIA/bdump/media_pmon_5364.trc:
ORA-00604: error occurred at recursive SQL level 1
ORA-01115: IO error reading block from file 1 (block # 89)
ORA-01110: data file 1: '/opt/oracle/oradata/MEDIA/system01.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
```

注意以上信息，SYSTEM 表空间文件也不可访问，这是一次非常彻底的批量误删除操作。

恢复过程

经过分析备份失败的原因，我们发现实质上是由于本地空间不足，在压缩时 SYSTEM 表空间等文件压缩失败，也就没有传送到远程备份主机，而另外还有 3 个数据文件没有备份，遗漏掉了。

另外一点幸运的是，备份卷只是删除了备份文件，并没有被重写覆盖，所以我们首先从备份卷上恢复除了 SYSTEM 表空间，还后利用不完全的备份，结合起来，就只缺少 4 个数据文件，而其中的三个数据文件，是一个表空间的 1，2，3 号文件，该表空间共有 19 个数据文件，这几个缺失的文件之前的已经基本写满，19 号文件也从删除中恢复了出来；**现在我们就利用这几个历史文件备份、恢复出来的新的文件、以及最近的成功备份，整合起来，通过文件修复、同步，将数据库成功打开**。

以下是部分数据文件，来自 2008 年的备份，以及从磁盘恢复出来最新的 SYSTEM 表空间和第 19 号媒体文件：

```
oracle@v880 ~/product/8.1.7.4/dbs$ls -al /opt/oracle/oradata/MEDIA/
total 76024148
drwxr-xr-x   2 oracle   dba          512 Apr 10 18:17 .
drwxr-xr-x   7 oracle   dba          512 Apr 10 19:33 ..
-rw-r-----   1 oracle   dba     16031744 Apr 10 18:21 control01.ctl
```

```
-rw-r-----   1 oracle   dba      16031744 Apr 10 18:21 control02.ctl
-rw-r-----   1 oracle   dba    8588976128 Sep 24  2008 media_01.dbf
-rw-r-----   1 oracle   dba    8587608064 Sep 24  2008 media_02.dbf
-rw-r-----   1 oracle   dba    8510480384 Sep 24  2008 media_03.dbf
-rw-r-----   1 oracle   dba    5028233216 Sep 24  2008 media_07.dbf
-rw-r-----   1 oracle   dba    7759470592 Apr  6 05:45 media_19.dbf
-rw-r-----   1 oracle   dba     398467072 Apr  6 06:33 system01.dbf
```

在以下目录中，找到了最近的其他备份文件：

```
oracle@v880 ~/oradata/data_2010_04_02$ls -l
total 190233104
-rw-r-----   1 oracle   dba    1048584192 Apr  2 06:14 PEPSI_01.dbf
-rw-r-----   1 oracle   dba    1048584192 Apr  2 06:15 PEPSI_02.dbf
-rw-r-----   1 oracle   dba      16031744 Apr 10 13:26 control01.ctl
-rw-r-----   1 oracle   dba    8482045952 Apr  2 03:02 media_04.dbf
-rw-r-----   1 oracle   dba    8587509760 Apr  2 03:15 media_05.dbf
-rw-r-----   1 oracle   dba    7961485312 Apr  2 03:28 media_06.dbf
-rw-r-----   1 oracle   dba    5956255744 Apr  2 03:41 media_08.dbf
-rw-r-----   1 oracle   dba    5515517952 Apr  2 03:51 media_09.dbf
-rw-r-----   1 oracle   dba    5614084096 Apr  2 03:59 media_10.dbf
-rw-r-----   1 oracle   dba    6912811008 Apr  2 04:08 media_11.dbf
-rw-r-----   1 oracle   dba    6736060416 Apr  2 04:19 media_12.dbf
-rw-r-----   1 oracle   dba    7914659840 Apr  2 04:30 media_13.dbf
-rw-r-----   1 oracle   dba    6316630016 Apr  2 04:44 media_14.dbf
-rw-r-----   1 oracle   dba    7050633216 Apr  2 04:55 media_15.dbf
-rw-r-----   1 oracle   dba    6970941440 Apr  2 05:07 media_16.dbf
-rw-r-----   1 oracle   dba    4823457792 Apr  2 05:19 media_17.dbf
-rw-r-----   1 oracle   dba    6396321792 Apr  2 05:27 media_18.dbf
```

所以现在我们拥有的是来自三个时期的文件。

经过检查，恢复出来的 SYSTEM 文件是完好的，这就为恢复提供了第一个可能性：

```
oracle@v880 ~/oradata/MEDIA$dbv file=system01.dbf blocksize=8192
DBVERIFY: Release 8.1.7.4.0 - Production on Sat Apr 10 20:04:03 2010
(c) Copyright 2000 Oracle Corporation.  All rights reserved.
```