以拯救之因

强制恢复导致 ORA-600 4000 错误案例

我曾在一本书上写道：

一知半解比无知更可怕。

一知半解的草率行事可能使数据库遭受意想不到的灾难，所以在接触到一个数据库时，如果没有相当的把握，请谨慎操作，不要让自己和数据库陷入到未知的危险境地。最基本的，**如果不可避免地要进行某些危险的维护性操作，应当在之前做好备份**。

在面对数据时，无知者不能无畏。

灾难描述

2011 年 12 月 30 日，一个运营商客户的核心数据库发生故障，无法启动。随后在工程师的草率介入后，数据库遭遇无法启动的灾难。

整个过程是这样的。

1. 客户误操作删除了一个数据文件，数据库报错。
2. 第三方服务商介入，试图清除该缺失文件。
3. 工程师选择重建控制文件，在当前数据库下执行不完全恢复。
4. 通过内部隐含参数强制重置日志打开数据库。
5. 数据库出现一系列 600 内部错误，无法启动。
6. 检查备份，发现近期未进行备份，无从备份恢复。
7. 灾难形成。

这则案例原本并不复杂，丢失了一个数据文件，如果没有备份，可以进行下列操作。

1. 在关闭数据库之前，类似前一节的情况，从文件描述符中进行恢复。
2. 如果数据库关闭不可恢复，并且可以接受损失，则离线抛弃该文件即可。
3. 如果该文件非常重要，可以通过存储级别的恢复，找回该文件。

具体选择何种措施，取决于数据的重要程度，但是贸然采取行动则会消灭一些可能性。比如，关闭数据库，措施 1 就无效了；如果在存储级别又执行了文件复制，覆盖了存储内容，则措施 3 也就无效了。

所以，针对不同的数据灾难，做出正确的判断，找到合适的技术支持尤为重要。而对于这个案例，以上三种可能性都被掩盖，数据库走向了一个错误的方向，并且在错误的方向上走得很远。

案例警示

分析这个案例的整个过程，我们总结出了如下教训。

1. **一知半解比无知更可怕。**

在这个案例中，技术人员显然对 Oracle 的技术认知不足，草率地采取了错误的手段和步骤，最终导致数据库状况和用户初衷相差了十万八千里。

前面提到的三种可能性在一系列的误操作面前变得无效，数据库必须接受不一致的灾难性考验。

通过这个案例，技术人员需要铭记，在着手处理故障之前，要遵循这样一个重要的守则：**保护现场，至少不要使情况变得更坏。**

在保护好现场之后，再进行破坏性或者把握性不大的恢复尝试；对于某些海量数据的情况，如果无法进行及时的备份，不得不在当前环境下进行恢复，那么就必须要求工程师具有极高的素质和精准的判断，明确知道每一个命令和步骤可能带来的后果和后续的处理方式。

每个工程师都要想一想：如果一个恢复尝试之后，数据库彻底无法启动，我们还能如何做？多思多想是对用户和自己负责，无知无畏不应当在生产环境中尝试。

2. **不要超越自己的能力范围。**

在技术领域，如果你采用的技术手段和方法超越了自己的能力范围，可能出现不可预知的后果，那么最好不要做这样的冒险，冒险意味着对用户的不负责任，同时也是对自己的折磨。如果决定冒险，那么**至少在自己的测试环境中进行同类测试，明确可能会出现的种种情况，这样的测试并不会耗费太多的时间。**

在这个案例中，Oracle 数据库的内部参数使用是不恰当的，导致了问题变得更加糟糕。实际上，内部参数的使用要求非常清晰地了解其含义，以及可能引致的后果，并且对随之而来的后果有应对方案。

对于用户，应当能够对第三方服务商进行适当的监督、确认，确保不可预期的后果不被贸然引入到数据库中。

3. **在不可逆操作之前执行备份。**

在需要执行不可逆操作之前，执行备份，要确保可以回退到之前的状态，以便尝试恢复失败之后，别人还有机会从头开始，这也是保护现场的概念。

除了对数据库的备份，实际上还应当在一些修改操作之前执行备份。比如对于特定数据块、文件头、ASM 磁盘组信息、日志文件等的备份，这些备份可以在关键时候帮助我们。

这里必须着重提一下日志文件，因为在强制 Resetlogs 时，日志文件会被清空刷新，如果不进行备份，其中的内容将永远丢失，而我们知道，有时候通过日志解析可以最大限度地找回数据，所以日志文件的备份也非常重要。

4. 管理者需要参与决策。

对于重要的数据环境，在执行重要的操作之前，管理者即便不了解详细的技术细节，也应当和技术人员进行沟通，听取技术方案、操作计划、实施步骤、现场保全、回退方案等。

管理者虽然可能不了解技术细节，但是其大局观和缜密思考应当为技术决策提供保障。管理者也应当在交流中感知技术人员是否了解详细情况，是否对方案有清晰把握、对执行自信无误。交流、提问和质疑也是对技术人员完善方案、认真思考的一种促进。

有了这样一个环节，就可以规避很多风险，而管理者的判断和决策也应当成为数据管理中重要的组成部分。

保护数据，保护现场，在处理故障时认真思考，谨慎决策，是用户和工程师们共同的职责。只有大家共同努力才能保障数据环境的持久安全。

技术回放

后来我们得知，这个数据库系统的数据量大约为 600GB，是个极其重要的业务系统，其近期备份也不能用于恢复。

```
SQL> select sum(bytes)/1024/1024/1024 from v$datafile;
SUM(BYTES)/1024/1024/1024
-------------------------
                617.027023
```

以下是从日志中获得的一些操作信息，从中可以看到第三方工程师的恢复操作。

首先尝试的是使用备份控制文件执行不完全恢复，恢复没有成功，未能打开数据库，日志信息如下，最后

一个文件号是 180，意味着数据库中有 180 个数据文件。

```
Fri Dec 30 14:10:12 2011
ALTER DATABASE RECOVER  database using backup controlfile until cancel
Fri Dec 30 14:10:12 2011
Media Recovery Start
WARNING! Recovering data file 1 from a fuzzy file. If not the current file
it might be an online backup taken without entering the begin backup command.
.................
WARNING! Recovering data file 180 from a fuzzy file. If not the current file
it might be an online backup taken without entering the begin backup command.
 parallel recovery started with 11 processes
ORA-279 signalled during:
ALTER DATABASE RECOVER  database using backup controlfile until cancel  ...
Fri Dec 30 14:10:20 2011
ALTER DATABASE RECOVER    CONTINUE DEFAULT
Fri Dec 30 14:10:20 2011
Media Recovery Log /archive/DMA/archivelog/2011_12_30/o1_mf_1_53868_%u_.arc
Errors with log /archive/DMA/archivelog/2011_12_30/o1_mf_1_53868_%u_.arc
ORA-308 signalled during: ALTER DATABASE RECOVER    CONTINUE DEFAULT  ...
```

这里的 ORA-308 错误是指无法找到需要的归档日志用于恢复。

```
Fri Dec 30 14:10:20 2011
ALTER DATABASE RECOVER CANCEL
ORA-1547 signalled during: ALTER DATABASE RECOVER CANCEL ...
Fri Dec 30 14:10:59 2011
ALTER DATABASE OPEN RESETLOGS
Fri Dec 30 14:10:59 2011
ORA-1194 signalled during: ALTER DATABASE OPEN RESETLOGS...
```

这里的 ORA-1194 是指文件不一致仍需恢复，所以无法使用 Resetlogs 方式打开数据库。通常这里会报出第一个 SYSTEM 文件需要进一步恢复。

在接下来的实例启动中，一个重要的隐含参数（ _allow_resetlogs_corruption ）被加入进来，用于强制打开数据库，这个参数的使用是有一系列后续影响的。

```
ksdpec: called for event 13740 prior to event group initialization
Starting up ORACLE RDBMS Version: 10.2.0.1.0.
System parameters with non-default values:
  processes                = 500
  sessions                 = 885
  __shared_pool_size       = 1811939328
  __large_pool_size        = 16777216
  __java_pool_size         = 16777216
  __streams_pool_size      = 33554432
  nls_language             = SIMPLIFIED CHINESE
  nls_territory            = CHINA
  sga_target               = 13639876608
  db_block_size            = 8192
  __db_cache_size          = 11744051200
  compatible               = 10.2.0.1.0
  db_file_multiblock_read_count= 16
  db_recovery_file_dest_size= 157286400000
  _allow_resetlogs_corruption= TRUE
  undo_management          = AUTO
  undo_tablespace          = UNDOTBS1
  remote_login_passwordfile= EXCLUSIVE
  db_domain                =
  job_queue_processes      = 10
  db_name                  = dma
  open_cursors             = 300
  pga_aggregate_target     = 2552233984
PMON started with pid=2, OS id=4487
```

设置后，数据库可以抛弃一致性，强制执行日志刷新，尝试打开数据库。当用户再次强制打开数据库时，遇到 ORA-600 4000 号错误，日志如下所示。

```
Fri Dec 30 17:54:22 2011
SMON: enabling cache recovery
Fri Dec 30 17:54:22 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_9074.trc:
```

```
ORA-00600: internal error code, arguments: [4000], [28], [], [], [], [], [], []
Fri Dec 30 17:54:27 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_9074.trc:
ORA-00704: 引导程序进程失败
ORA-00704: 引导程序进程失败
ORA-00600: internal error code, arguments: [4000], [28], [], [], [], [], [], []
```

然后工程师重建了控制文件。

```
Fri Dec 30 14:22:34 2011
CREATE CONTROLFILE REUSE DATABASE "DMA" RESETLOGS  NOARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 5
    MAXDATAFILES 1024
    MAXINSTANCES 8
    MAXLOGHISTORY 1168
……
```

当然，这样的步骤仍然解决不了问题，于是工程师又加入 ROLLBACK 参数，强制将 28 号回滚段离线标记为损坏。

```
   _allow_resetlogs_corruption= TRUE
  _offline_rollback_segments= (_SYSSMU28$)
  _corrupted_rollback_segments= (_SYSSMU28$)
```

这些尝试无助于故障的解决，最终的错误停留在 ORA-600 4000 上，其中第二个参数 28 的确代表的是回滚段号，也就是在 28 号回滚段上存在不一致事务。

```
Fri Dec 30 19:22:18 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_10809.trc:
ORA-00600: 内部错误代码, 参数: [4000], [28], [], [], [], [], [], []
Fri Dec 30 19:22:22 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_10809.trc:
ORA-00704: 引导程序进程失败
ORA-00704: 引导程序进程失败
ORA-00600: 内部错误代码, 参数: [4000], [28], [], [], [], [], [], []
```

该数据库在重建控制文件后显示共有 180 个数据库文件，以下是查询 v$datafile 视图后的末端输出。

```
NAME
------------------------------------------------------------------
/data2/userdata/d_wireless_dbs_2.dbf
/data2/userdata/d_wireless_dbs_18.dbf
/data2/userdata/d_wireless_dbs_17.dbf
/data2/userdata/d_wireless_dbs_15.dbf

已选择 180 行。
```

数据库最后完成的检查点如下。

```
SQL> col CHECKPOINT_CHANGE#  for 99999999999999999
SQL> select file#,checkpoint_change# from v$datafile where rownum <10;
     FILE# CHECKPOINT_CHANGE#
---------- ------------------
         1     12506717382696
         2     12506717382696
         3     12506717382696
         4     12506717382696
         5     12506717382696
         6     12506717382696
         7     12506717382696
         8     12506717382696
         9     12506717382696

已选择 9 行。
```

这个案例原本不应该走到这个地步，但由于工程师对于技术判断的一知半解，无知无畏，最终将数据库带入了一个灾难的境地。而且工程师处理之前未进行备份，无法从头开始，因此只能从这个 ORA-600 的错误状态上进行进一步的分析处理。

恢复过程

ORA-600 的 4000 号错误是指，在数据库启动检测中，发现某些数据块的 SCN（具体与 CSC 和 Commit SCN

有关）大于系统当前的 SCN，违反了数据库的一致性，出现内部错误异常。

Oracle 对这个错误的描述如下。

This has the potential to be a very serious error. It means that Oracle has tried to find an undo segment number in the dictionary cache and failed.

这个错误和潜在的严重错误有关，它意味着 Oracle 试图在字典缓存中查找一个回滚段号，但是失败了。实际上也就是在执行回退时读取回滚段遇到了错误。

ORA-600 4000 错误揭秘

我们可以根据出现错误时抛出的跟踪文件进行进一步的分析。

```
*** 2011-12-30 16:14:02.081
ksedmp: internal or fatal error
ORA-00600: 内部错误代码, 参数: [4000], [28], [], [], [], [], [], []
Current SQL statement for this session:
select ctime, mtime, stime from obj$ where obj# = :1
----- Call Stack Trace -----
calling              call     entry                argument values in hex
location             type     point                (? means dubious value)
-------------------- -------- -------------------- ----------------------------
ksedst()+64          call     _etext_f()+23058430  000000000 ? 000000001 ?
                              09017233136
ksedmp()+1680        call     _etext_f()+23058430  000000000 ?
                              09017233136          C000000000000D20 ?
                                                   40000000052C8DF0 ?
                                                   000000000 ? 000000000 ?
                                                   000000000 ?
ksfdmp()+48          call     _etext_f()+23058430  000000003 ?
                              09017233136
```

错误出现在对于 OBJ$的访问上，OBJ$**是数据库初始化启动时必须访问的一张数据表**。在跟踪文件中还包含存在不一致事务的数据块信息，以下是第一个抛出异常的数据块。

注意这里的两个内容。

1. XID：0x001c.01d.00069f98。

这里 XID 的第一部分为 UNDO 段号，1C 即 28，也就是 ORA-600 4000 错误抛出的第一个参数。

2. CSC：最后一次完成的块清除 SCN。

CSC 即 **SCN of the last block cleanout**，这个 SCN 是 0xb5f.f28cc1fd，即 12 506 719 109 629。注意这个 SCN 大于前面查询到的数据文件最后 SCN（12 506 717 382 696），两者相差 1 726 933。

这两者就是“ORA-00600: 内部错误代码，参数: [4000], [28]……”的成因。

```
Block header dump:  0x0040007a
 Object id on Block? Y
 seg/obj: 0x12  csc: 0xb5f.f28cc1fd  itc: 1  flg: -  typ: 1 - DATA
     fsl: 0  fnx: 0x0 ver: 0x01

 Itl           Xid                  Uba         Flag  Lck        Scn/Fsc
0x01   0x001c.01d.00069f98  0x008092df.75bc.07  --U-    1  fsc 0x0000.f28cc1fe

data_block_dump,data header at 0xc00000025d526044
===============
tsiz: 0x1fb8
hsiz: 0xea
pbl: 0xc00000025d526044
bdba: 0x0040007a
     76543210
flag=--------
ntab=1
nrow=108
frre=-1
fsbo=0xea
fseo=0x40c
avsp=0x368
tosp=0x368
0xe:pti[0]      nrow=108        offs=0
0x12:pri[0]     offs=0x1f7c
```

```
0x14:pri[1]     offs=0x1f3c
0x16:pri[2]     offs=0x1efb
0x18:pri[3]     offs=0x1ebc
0x1a:pri[4]     offs=0x1e7c
0x1c:pri[5]     offs=0x1e3c
0x1e:pri[6]     offs=0x1e00
0x20:pri[7]     offs=0x1dbe
```

这个 ITL 事务槽锁定了一行记录，可以从后面的转储中找到这条记录，第 30 行（块内行记录由 0 开始）。

```
tab 0, row 29, @0x40c
tl: 71 fb: --H-FL-- lb: 0x1  cc: 17
col  0: [ 2]  c1 02                    --这是 1
col  1: [ 5]  c4 02 4d 1e 2f
col  2: [ 1]  80                       --这是 0
col  3: [12]  5f 4e 45 58 54 5f 4f 42 4a 45 43 54
col  4: [ 2]  c1 02
col  5: *NULL*
col  6: [ 1]  80
col  7: [ 7]  78 6a 01 05 0b 0f 11     --这三个字段都是时间，长度为 7
col  8: [ 7]  78 6f 0c 1e 0c 17 2a
col  9: [ 7]  78 6a 01 05 0b 0f 11
col 10: [ 1]  80
col 11: *NULL*
col 12: *NULL*
col 13: [ 1]  80
col 14: *NULL*
col 15: [ 1]  80
col 16: [ 4]  c3 07 38 24
```

那么这行记录是什么数据呢?

通过下列函数可以对字符型数据做一个简单的转换，帮助进一步分析。

```
create or replace function hextostr(hexstr varchar2) return varchar2 is
   i       number;
   s       char;
```

```
    str1  varchar2(20);
    rst    varchar2(200);
begin
    i     := 1;
    rst  := '';
    Loop
        str1  :=  substr(replace(hexstr,  '  '),  i,  2);
        select  chr(to_number(str1,  'xxxxxxxxxx'))  into  s  from  dual;
        rst  :=  rst  ||  s;
        exit  when  i  >  lengthb(hexstr);
        i  :=  i  +  2;
    end  loop;
    return  rst;
end;
/
```

经过转换可以得出，这个记录行是_NEXT_OBJECT。

```
SQL> select hextostr('5f 4e 45 58 54 5f 4f 42 4a 45 43 54') colname from dual;

COLNAME
--------------------------------------------------------------------------------
_NEXT_OBJECT
```

以下通过杨廷琨提供的一个函数（参见本书附录）进行全面的数据类型转换。

```
SQL> select F_GET_FROM_DUMP(replace('5f 4e 45 58 54 5f 4f 42 4a 45 43 54',' ',','),'VARCHAR2')
OUTPUT from dual;
OUTPUT
--------------------------------------------------------------------------------
_NEXT_OBJECT
SQL> select F_GET_FROM_DUMP(replace('78 6a 01 05 0b 0f 11 ',' ',','),'DATE') OUTPUT from dual;
OUTPUT
--------------------------------------------------------------------------------
2006-1-5 10:14:16
SQL> select F_GET_FROM_DUMP(replace('c4 02 4d 1e 2f',' ',','),'NUMBER') OUTPUT from dual;
```

```
OUTPUT
--------------------------------------------------------------------------------
1762946
```

注意这里的_NEXT_OBJECT 是一个数据库中隐藏的特殊对象，实际上是为了生成下一个对象的 object_id 或 dba_object_id 而设定的，在数据库中的对象被 Truncate 或者生成一个新的对象时使用。所以这个对象上常常会持有锁定，如果数据库异常，其锁定通常难以正常释放。

对比以下信息，可以确认这条记录的正确性。

```
SQL> desc obj$
 Name                                                      Null?    Type
 ----------------------------------------------------------------- -------- ---------
 OBJ#                                                      NOT NULL NUMBER
 DATAOBJ#                                                           NUMBER
 OWNER#                                                    NOT NULL NUMBER
 NAME                                                      NOT NULL
VARCHAR2(30)
 NAMESPACE                                                 NOT NULL NUMBER
 SUBNAME
VARCHAR2(30)
 TYPE#                                                     NOT NULL NUMBER
 CTIME                                                     NOT NULL DATE
 MTIME                                                     NOT NULL DATE
 STIME                                                     NOT NULL DATE
 STATUS                                                    NOT NULL NUMBER
 REMOTEOWNER
VARCHAR2(30)
 LINKNAME
VARCHAR2(128)
 FLAGS                                                              NUMBER
 OID$                                                               RAW(16)
 SPARE1                                                             NUMBER
 SPARE2                                                             NUMBER
 SPARE3                                                             NUMBER
```

```
 SPARE4
VARCHAR2(1000)
 SPARE5
VARCHAR2(1000)
 SPARE6                                                                    DATE
SQL> select obj#,dataobj#,owner#,name from obj$ where name='_NEXT_OBJECT';
      OBJ#    DATAOBJ#      OWNER# NAME
---------- ---------- ---------- ------------------------------
         1     198226          0 _NEXT_OBJECT
```

检查整个跟踪文件，可以发现 0xb5f.f28cc1fd 是最大的一个 CSC 号。

```
eygle$ grep seg/obj dma_ora_6991.trc
 seg/obj: 0x12  csc: 0xb5f.f28cc1fd  itc: 1  flg: -  typ: 1 - DATA
 seg/obj: 0x38  csc: 0x00.141  itc: 1  flg: O  typ: 1 - DATA
 seg/obj: 0x24  csc: 0xb57.571f621  itc: 1  flg: -  typ: 2 - INDEX
 seg/obj: 0x24  csc: 0xb5a.c4e1f05b  itc: 1  flg: -  typ: 2 - INDEX
 seg/obj: 0x24  csc: 0x00.279a  itc: 2  flg: -  typ: 2 - INDEX
 seg/obj: 0x12  csc: 0xb5f.f28cc1fd  itc: 1  flg: -  typ: 1 - DATA
```

由于是通过这个 CSC 号进行回滚段的分析，因而这个非法的 SCN 号导致了最终故障。

找到这个原因之后，我们只需要将数据库的 SCN 整体向前推进消除掉这个 SCN 差异，数据库就会通过延迟提交清除这个事务信息，错误即可解除。

通过_minimum_giga_scn 消除 SCN 异常

在 Oracle 10g 中，可以通过隐含参数_minimum_giga_scn 来推进 SCN，每个 1 将 SCN 推进到 1G。

```
SQL> select ksppinm,ksppdesc from x$ksppi
  2  where ksppinm like '%giga%'
  3  /
KSPPINM                        KSPPDESC
------------------------------ --------------------------------------------
_minimum_giga_scn              Minimum SCN to start with in 2^30 units
```

以下将 SCN 0xb5f.f28cc1fd 进行一个转换运算，获得十进制值，再转换成以_minimum_giga_scn 为单位。

```
SQL> col scn for 99999999999999999
SQL> select to_number('b5ff28cc1fd','xxxxxxxxxxxx') scn from dual;
        SCN
-------------------
    12506719109629
SQL> select 12506719109629/1024/1024/1024 giga from dual;
      GIGA
----------
11647.7898
```

对于这个数据库，将该参数设置为 11649，稍微多推进一点，数据库的错误就可以被成功消除了。我们首先在初始化参数文件中增加如下两个参数。

```
*._allow_resetlogs_corruption=true
*._minimum_giga_scn=11649
```

然后尝试打开数据库。

```
SQL> startup mount pfile=initdma_eygle.ora
ORACLE 例程已经启动。

Total System Global Area 1.3640E+10 bytes
Fixed Size                  2115392 bytes
Variable Size            1889450176 bytes
Database Buffers         1.1744E+10 bytes
Redo Buffers                4259840 bytes
数据库装载完毕。
SQL> recover database using backup controlfile until cancel;
ORA-00279: 更改 12506717382696 (在 12/30/2011 19:22:18 生成) 对于线程 1 是必需的
ORA-00289: 建议:/archive/DMA/archivelog/2011_12_31/o1_mf_1_5_%u_.arc
ORA-00280: 更改 12506717382696 (用于线程 1) 在序列 #5 中


指定日志: {<RET>=suggested | filename | AUTO | CANCEL}
cancel
ORA-01547: 警告: RECOVER 成功但 OPEN RESETLOGS 将出现如下错误
```

```
ORA-01194: 文件 1 需要更多的恢复来保持一致性
ORA-01110: 数据文件 1: '/dma_sys/sysdata/dma/system01.dbf'

ORA-01112: 未启动介质恢复
SQL> alter database open resetlogs;
alter database open resetlogs
*
第 1 行出现错误:
ORA-00603: ORACLE 服务器会话因致命错误而终止
```

此时观察告警日志文件，可以发现 4000 错误已经不再出现，但是由于 UNDO 的 4194 错误导致数据库再次崩溃。4194 错误的解决就容易很多了。

在以上尝试启动时，告警日志的提示如下。

```
Sat Dec 31 16:06:30 2011
alter database open resetlogs
Sat Dec 31 16:06:30 2011
RESETLOGS is being done without consistancy checks. This may result in a corrupted
database. The database should be recreated.
RESETLOGS after incomplete recovery UNTIL CHANGE 12506717382696
Resetting resetlogs activation ID 984777556 (0x3ab28354)
Sat Dec 31 16:07:12 2011
Setting recovery target incarnation to 3
Sat Dec 31 16:07:12 2011
Advancing SCN to 12508018507776 according to _minimum_giga_scn
```

注意以上日志提示，因为_minimum_giga_scn 参数的设置，将 SCN 增进到了 12508018507776。

继续看打开数据库接下来的日志信息。

```
Sat Dec 31 16:07:12 2011
Assigning activation ID 984841183 (0x3ab37bdf)
Thread 1 opened at log sequence 1
  Current log# 4 seq# 1 mem# 0: /dma_sys/sysdata/dma/redo04a.log
  Current log# 4 seq# 1 mem# 1: /dma_sys/sysdata/dma/redo04b.log
Successful open of redo thread 1
```

```
Sat Dec 31 16:07:12 2011
MTTR advisory is disabled because FAST_START_MTTR_TARGET is not set
Sat Dec 31 16:07:12 2011
SMON: enabling cache recovery
Sat Dec 31 16:07:13 2011
Successfully onlined Undo Tablespace 1.
Dictionary check beginning
Tablespace 'TEMP' #3 found in data dictionary, but not in the controlfile. Adding to
controlfile.
File #181 found in data dictionary but not in controlfile.
Creating OFFLINE file 'MISSING00181' in the controlfile.
This file can no longer be recovered so it must be dropped.
Dictionary check complete
```

注意这里提示说 181 号文件在数据字典（file$）中存在，但是在控制文件中不存在，现在被添加回控制文件，命名为 MISSING00181。这也告诉我们，试图通过重建控制文件以去掉某个文件的做法是根本错误的。

这里的 181 号文件就是被误操作删除掉的文件。提示也告知用户，当数据库被 resetlogs 打开，这个文件将不能够再通过恢复加入到数据库中，只能被删除。

接下来的下列提示说应当添加临时表空间。

```
**********************************************************************
Sat Dec 31 16:07:13 2011
SMON: enabling tx recovery
WARNING: The following temporary tablespaces contain no files.
         This condition can occur when a backup controlfile has
         been restored.  It may be necessary to add files to these
         tablespaces.  That can be done using the SQL statement:

         ALTER TABLESPACE <tablespace_name> ADD TEMPFILE

         Alternatively, if these temporary tablespaces are no longer
         needed, then they can be dropped.
Sat Dec 31 16:07:13 2011
           Empty temporary tablespace: TEMP
```

```
**********************************************************************
Database Characterset is ZHS16GBK
```

ORA-600 4194 错误 UNDO 故障消除

接下来是因为一致性被破坏而出现的 ORA-600 4194 数据库内部错误，这与 UNDO 回滚有关。

```
Sat Dec 31 16:07:14 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_4379.trc:
ORA-00600: 内部错误代码, 参数: [4194], [42], [30], [], [], [], [], []
Doing block recovery for file 2 block 16237
Block recovery from logseq 1, block 111 to scn 12508018507983
Sat Dec 31 16:07:18 2011
Recovery of Online Redo Log: Thread 1 Group 4 Seq 1 Reading mem 0
  Mem# 0 errs 0: /dma_sys/sysdata/dma/redo04a.log
  Mem# 1 errs 0: /dma_sys/sysdata/dma/redo04b.log
Block recovery stopped at EOT rba 1.1008.16
Block recovery completed at rba 1.1008.16, scn 2912.1073742029
Doing block recovery for file 2 block 441
Block recovery from logseq 1, block 111 to scn 12508018507979
Sat Dec 31 16:07:18 2011
Recovery of Online Redo Log: Thread 1 Group 4 Seq 1 Reading mem 0
  Mem# 0 errs 0: /dma_sys/sysdata/dma/redo04a.log
  Mem# 1 errs 0: /dma_sys/sysdata/dma/redo04b.log
Block recovery completed at rba 1.112.16, scn 2912.1073742028
```

以上日志提示了恢复到达的 SCN 位置和 RBA 信息。

以下提示说明在恢复文件 2 block 45224 时出现了错误，这是对于 UNDO 的恢复尝试.

```
Sat Dec 31 16:08:25 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_4379.trc:
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
Sat Dec 31 16:08:27 2011
DEBUG: Replaying xcb 0xc00000046193d230, pmd 0xc0000000a79a88d0 for failed op 8
Doing block recovery for file 2 block 45224
```

```
No block recovery was needed
Sat Dec 31 16:09:35 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_4379.trc:
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
Sat Dec 31 16:09:37 2011
DEBUG: Replaying xcb 0xc00000046193d230, pmd 0xc0000000a79a88d0 for failed op 8
Sat Dec 31 16:09:37 2011
Errors in file /opt/app/oracle/admin/dma/udump/dma_ora_4379.trc:
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
Doing block recovery for file 2 block 45224
No block recovery was needed
Sat Dec 31 16:10:44 2011
Errors in file /opt/app/oracle/admin/dma/bdump/dma_smon_4365.trc:
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
Sat Dec 31 16:10:46 2011
DEBUG: Replaying xcb 0xc00000046193d230, pmd 0xc0000000a79a88d0 for failed op 8
Doing block recovery for file 2 block 45224
No block recovery was needed
Sat Dec 31 16:10:46 2011
Fatal internal error happened while SMON was doing active transaction recovery.
Sat Dec 31 16:10:46 2011
Errors in file /opt/app/oracle/admin/dma/bdump/dma_smon_4365.trc:
ORA-00600: 内部错误代码, 参数: [4194], [66], [60], [], [], [], [], []
SMON: terminating instance due to error 474
Instance terminated by SMON, pid = 4365
```

最后数据库因为 4914 错误实例崩溃。通过内部参数强制打开数据库，就破坏了数据库的一致性，不可避免地会遇到一系列的内部错误。

ORA-600 [4194]错误的官方解释是：“Undo Record Number Mismatch While Adding Undo Record”，当数据库通过 REDO 恢复来增加 UNDO 记录时，发现 UNDO 记录的号码不匹配，也就是出现了不一致。

这可以通过重建 UNDO 表空间来解决。设置如下初始化参数。

```
*.undo_management='MANUAL'
```

然后启动数据库重建 UNDO 表空间。

```
SQL> startup pfile=initdma_eygle.ora
ORACLE 例程已经启动。

Total System Global Area 1.3640E+10 bytes
Fixed Size                  2115392 bytes
Variable Size            1889450176 bytes
Database Buffers         1.1744E+10 bytes
Redo Buffers                4259840 bytes
数据库装载完毕。
数据库已经打开。
SQL> drop tablespace undotbs1;
表空间已删除。
SQL> create undo tablespace undotbs1
  2  datafile '/dma_sys/sysdata/dma/undotbs202.dbf' size 500M;
表空间已创建。
```

在这之后可以修改 undo_management 参数。

```
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS1'
```

最后重新启动数据库，此时数据库通常就可以正常无误地启动。

```
[oracle] % sqlplus "/ as sysdba"
SQL*Plus: Release 10.2.0.1.0 - Production on 星期六 12月 31 16:41:01 2011
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

已连接到空闲例程。
SQL> startup
ORACLE 例程已经启动。

Total System Global Area 1.3640E+10 bytes
Fixed Size                  2115392 bytes
```

```
Variable Size             8264792256 bytes
Database Buffers          5368709120 bytes
Redo Buffers                 4259840 bytes
数据库装载完毕。
数据库已经打开。
SQL> ! ls -l /dma_sys/sysdata/dma/ |grep temp
-rw-r-----    1 oracle     oinstall   16106135552 12 月 31 日 16:41 perftemp01.dbf
-rw-r-----    1 oracle     oinstall   16106135552 12 月 30 日 11:41 temp01.dbf
SQL>  ALTER TABLESPACE temp ADD TEMPFILE '/dma_sys/sysdata/dma/temp02.dbf' size 16G ;
表空间已更改。
SQL> create table test as select * from obj$;
表已创建。
SQL> drop table test purge;
表已删除。
SQL> alter database datafile '/dma_sys/sysdata/dma/undotbs202.dbf'
  2  autoextend on next 200m maxsize 16G;
数据库已更改。
```

随后查询得到的文件名列表大致如下（这里的 MISSING00181 即丢失的文件，数据库默认以 MISSING 命名）。

```
NAME
--------------------------------------------------------------------------------
/data2/userdata/d_wireless_dbs_2.dbf
/data2/userdata/d_wireless_dbs_18.dbf
/data2/userdata/d_wireless_dbs_17.dbf
/data2/userdata/d_wireless_dbs_15.dbf
/opt/app/oracle/product/10.2.0/dbs/MISSING00181
已选择 181 行。
```

这样就最终完成了恢复。在通过这些手段恢复之后，由于数据库的一致性和完整性被破坏，因此通常建议导出数据，重建数据库。在消除 4194 错误之后，数据库还可能遇到一些其他的 ORA-600 错误，不过这些错误通常都是可以解决和消除的。

以优化之名

存储优化导致表空间误删除案例

一人蛇先成，引酒且饮之，乃左手持卮，右手画蛇曰：“吾能为之足！”为蛇足者，终亡其酒。

——《战国策•齐二》

嘉招欲覆杯中渌，丽唱仍添锦上花。

——宋•王安石《即事》

宁锦上添花，勿画蛇添足。

在很多数据灾难中，我们看到很多原本是可有可无的操作，或者是锦上添花的工作，最后却由于处置不当、准备不足或认知不够，导致了数据灾难，当事人后悔不迭。

所谓数据无小事，在和数据打交道时，我们一定要谨而慎之，切勿一失足而至追悔莫及。

灾难描述

2011 年 12 月 31 日，我们接到一个紧急的数据救援请求。据粗略了解，事情大致是这样的。

1. 年底，开发商帮助用户进行数据库性能优化。
2. 优化内容之一是存储优化。
3. 存储优化的方法是进行表空间重建。
4. 开放商先删除表空间（成功），然后重建表空间（失败）。
5. 用户发现删除的表空间中的数据未备份。
6. 灾难形成。

对于这则案例，如果数据库不进行优化，那么稳定运行的问题还是不大的。然而，准备不足的这些维护操作，则使数据库彻底陷入了瘫痪的困境。

同样在 2011 年底，我们还遇到了另外一则完全类似的案例，情形大致如下。

1. 客户数据库的安全性要求极高。

2. 客户要定期接受上级单位的例行检查。
3. 为了满足安全评估要求，客户决定对系统进行全新重构。
4. 在导出备份之后，将主机格式化并重建数据库。
5. 在重建后恢复时发现，导出的备份文件有错误，无法导入数据库中。
6. 灾难形成。

现场分析发现，导出文件大小是正确的，但是通过十六进制模式查看发现文件末尾全是空白，没有数据，推测可能是通过移动硬盘备份时出现了问题，未正常退出或插拔移动硬盘导致数据损失。这个导出文件是不足以用来恢复了。后来通过存储级别的恢复，在格式化后的硬盘上找到了之前的历史备份，最终总算恢复了大部分数据。

有时一个良好的设想因为执行不当可能成为灾难的源头，所以在完成了基本的保障之后，无为而治在有些情况下却能保证数据库的安全稳定运行，折腾是数据灾难之源。

案例警示

这两个数据灾难带给了我们如下教训。

1. **在任何破坏性操作之前，必须严格验证备份的有效性。**

这里所说的破坏性操作包括删除表空间、数据文件、数据表等。

严格验证备份的有效性，如果可能，应该进一步演进到，**操作之前进行全面有效的备份**。有时候不要相信别人传达的诸如已经备份、存在备份、有人备份之类的信息，因为不同方式的备份可能不适合你的操作恢复，别人的备份也可能存在你所不知道的问题，一旦需要的数据未成功备份，灾难就会出现。

所以，**重要的环节需要亲自确认，避免传递错误带来的不确定性**。

2. **在可能情况下保留多份备份介质。**

很多事实显示，有时候单纯一份备份是不可靠的，磁带、移动硬盘等介质的备份更不可靠。所以，如果可能，对于重要数据的备份，最好保留多份介质，尤其是要对原有环境进行破坏、迁移、重构等情况时。

通常的备份策略，还应当结合物理备份和逻辑备份、表结构备份等来实施。对于极其重要的核心表，要保持经常性备份，多一份备份就多一份安全。

3. **避免使用无法把握的新技术。**

前述案例的客户使用了自动存储管理技术（Oracle ASM）。ASM 使用裸设备，这造成了用户无法直接看到

数据文件，也就无从直接进行文件级别的复制备份，因而增加了备份的复杂性和难度。

显然用户和开发商运维人员都不能够深入了解这一技术。从这个意义上说，**并非先进的技术就适合任何环境，好的技术要和好的运维结合起来才能为用户创造价值**。客户在选择 Oracle 数据库的同时，必须要同时接受因此而可能支付的运行维护成本。

当然，系统架构人员也应当从用户的实际出发，为用户建议适合的系统架构，**只有真正适合用户的技术才是最好的技术**。

4. **必须有人能够把握技术全局。**

在这个案例中，显然没有人能够从整体方案上把握全局。如果在核心的数据运维中没有人能把握全局，那么任何操作都应当极其慎重，或者干脆选择放弃进行破坏性数据维护操作。

典型地，如果没有回退方案，任何破坏性的数据操作都不应当进行。

5. **制订方案并且按照方案的步骤执行。**

重要的维护工作应当制订方案，并且列出明确的操作步骤和命令。这些步骤和命令应当是通过测试验证的。在执行过程中，遵循方案的步骤来进行相关操作，一旦遇到异常必须停下来进行分析或者回退。

严格对数据负责，不因主观的重要和非重要判断行事。

6. **数据库维护操作应当通过命令行完成，避免使用图形化工具。**

虽然图形化工具会为工作带来便利，但是其背后隐藏着不确定性和风险。通常用户很难确定图形后面默认和非默认的选项，任何一个错误的勾选都可能使情况变得复杂。

鉴于已经有很多用户在图形工具上遭遇挫折，我们建议用户通过命令行完成对数据库的维护操作，比如使用 SQL*Plus 工具。命令行会迫使你明确所发出的每一个指令，无形中可减少风险的出现。

明确工具的风险和用途，这也是对 DBA 的一个专业素质要求。

充足的数据备份和良好的维护计划是数据安全的守护者，在数据运维中要时刻注意。

技术回放

接下来我们看一看第一个案例的具体情形。

2011 年 12 月 31 日 12 时 34 分 53 秒，简简单单的一句 DROP TABLESPACE，删除了一个表空间中的三个

数据文件，用时不过 3 秒钟，可是后来的恢复工作却用时数天。

```
Sat Dec 31 12:34:53 2011
DROP TABLESPACE XF_PRODUCT INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS

Sat Dec 31 12:34:56 2011
Deleted file +DATA/orcl/datafile/xf_product01.dbf
Deleted file +DATA/orcl/datafile/xf_product02.dbf
Deleted file +DATA/orcl/datafile/xf_product03.dbf
Completed:
DROP TABLESPACE XF_PRODUCT INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS
```

在这里可以看到，Oracle 的 ASM 自动将数据文件删除了。在用户发出的命令中，还包含了删除数据文件和内容的指令，结果非常彻底地消灭了数据。

注意，这显然是通过工具发出的命令，将一条命令书写得如此完备对 DBA 的要求是非常高的。所以我们建议，重要的维护操作要在命令行完成，避免使用图形化的工具软件。工具虽然可带来方便，但是也会带来隐藏的风险和不确定性。

用户随后还删除了一系列的索引文件和临时文件。从为表空间规划了独立的索引文件和临时文件来看，这个数据库的早期规划是相当专业和细致的，只是后期的管理和运维显然没有跟上规划的步伐。

```
Sat Dec 31 12:34:56 2011
DROP TABLESPACE XF_PRODUCT_INDEX  INCLUDING CONTENTS AND DATAFILES CASCADE
CONSTRAINTS

Deleted file +DATA/orcl/datafile/xf_product_index01.dbf
Completed:
DROP TABLESPACE XF_PRODUCT_INDEX  INCLUDING CONTENTS AND DATAFILES CASCADE
CONSTRAINTS
Sat Dec 31 12:34:56 2011
DROP TABLESPACE XF_PRODUCT_TEMP   INCLUDING CONTENTS AND DATAFILES CASCADE
CONSTRAINTS

Deleted file +DATA/orcl/tempfile/xf_product_temp03.dbf
Deleted file +DATA/orcl/tempfile/xf_product_temp02.dbf
Deleted file +DATA/orcl/tempfile/xf_product_temp01.dbf
```