规避掉误删除的风险。

4. **数据备份应进行必要的检查和确认。**

很多用户在部署了数据备份手段之后就认为可以高枕无忧了，而很少去检验备份的有效性和完整性，一旦出现问题需要恢复，才发现备份无效，就很难挽回了。我们曾经多次遇到客户在恢复时发现磁带不可用的数据安全故障。

所以提醒大家：在**进行了数据备份之后，要定期检查备份成功与否，以及备份的完好完整性**，如果使用了备份介质，还要检查备份介质的完好性。很多用户在进行了磁带备份之后，恢复时却发现磁带不可读取，因而造成无法挽救的灾难，这是非常令人感到惋惜的。

5. **避免在疲劳或不清醒状态独自做出重要判断。**

这次故障首次发生和处置是在凌晨。人在疲劳、熬夜或者睡梦中被叫起的半梦半醒状态下，所有的判断都可能考虑不周。所以，如同拒绝疲劳驾驶一样，应当拒绝疲劳运维，或者至少避免在不清醒的状态下独自做出重要判断，必要时，至少获得一个额外的指导或确认。

6. **在对故障做出清晰判断之前不要贸然采取措施。**

这个故障的恢复得益于用户的正确判断。在问题出现后，用户并未关闭数据库，而是将其保持在打开状态下，然后进行故障分析处理，这就为简便恢复提供了可能。

所以当遇到数据库故障时，在得出清晰的分析结论和处理方法之前，不要贸然采取措施，草率的决策可能导致问题复杂化，进而引发更加复杂的级联故障。

遵循一定的守则和规范，将显著提高数据安全性，保障数据环境的稳定运行。

技术回放

接下来让我们通过技术回放，来看一看数据库是如何陷入这场灾难的。

从告警日志看最初数据库出现的问题是归档日志无法写出，出现归档错误，归档停滞导致数据库的所有 DML 事务无法进行，在线业务受到影响。

以下是日志摘录信息，注意时间，故障出现在 2011 年 12 月 30 日凌晨 3 点 34 分。发生在夜间的故障通常影响的业务范围小，但是其处理响应也可能会相对缓慢，处理上也可能出现误操作，应当警惕。

```
Fri Dec 30 03:34:33 2011
```

```
ARC0: Encountered disk I/O error 19502
ARC0: Closing local archive destination LOG_ARCHIVE_DEST_1:
'/archlog/1_6578_765575017.dbf' (error 19502) (com1)
ARC0: I/O error 19502 archiving log 5 to '/archlog/1_6578_765575017.dbf'
ARCH: Archival stopped, error occurred. Will continue retrying
ORACLE Instance com1 - Archival Error
ORA-16038: log 5 sequence# 6578 cannot be archived
ORA-19502: write error on file "", block number  (block size=512)
ORA-00312: online log 5 thread 1: '/oradata2/redolog5_1.dbf'
ORA-00312: online log 5 thread 1: '/oradata3/redolog5_2.dbf'
ARCH: Archival stopped, error occurred. Will continue retrying
```

经过处理，在凌晨 5 点 22 分左右，归档得以继续，归档进程从失败中得到释放。此次的故障处理时间大约是 50 分钟。

```
Fri Dec 30 05:22:51 2011
Archived Log entry 15601 added for thread 1 sequence 6578 ID 0xffffffffcf4313f7 dest
1: krse_arc_driver_core: Successful archiving of previously failed ORL
Archiver process freed from errors. No longer stopped
```

那么，这里的空间是如何释放出来的呢?

显然是操作系统级别的行为，通过删除归档、删除文件释放空间，但是注意，这位在凌晨被吵醒的工程师很可能草率地做出了错误的判断。

后果在日志中表露无疑，十几分钟后，一系列的数据文件出现无法访问、无法打开的读/写错误。

```
Fri Dec 30 05:38:22 2011
Errors in file /opt/oracle/app/diag/rdbms/com/com1/trace/com1_m000_4496.trc:
ORA-01116: error in opening database file 229
ORA-01110: data file 229: '/oradata4/LTYY_DATA55.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
Errors in file /opt/oracle/app/diag/rdbms/com/com1/trace/com1_m000_4496.trc:
ORA-01116: error in opening database file 235
ORA-01110: data file 235: '/oradata4/LTYY_DATA56.dbf'
```

```
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
Errors in file /opt/oracle/app/diag/rdbms/com/com1/trace/com1_m000_4496.trc:
ORA-01116: error in opening database file 232
ORA-01110: data file 232: '/oradata4/SMS_DATA47.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
```

经过检查，最终用户确认，oradata4 整个目录在操作系统上被删除，其中的所有数据文件荡然无存。以下是丢失的文件列表和状态，从 v$datafile 中可以查询到这些信息。这些都是极为重要的业务数据文件，必须要恢复出来。

TS#	FILE#	NAME	BYTES	STATUS
5	229	/oradata4/YDDY_DATA55.dbf	0	ONLINE
19	230	/oradata4/EFB01.dbf	0	ONLINE
18	231	/oradata4/undotbs203.dbf	0	RECOVER
9	232	/oradata4/SMS_DATA47.dbf	0	ONLINE
9	233	/oradata4/SMS_DATA48.dbf	0	ONLINE
9	234	/oradata4/SMS_DATA49.dbf	0	ONLINE
5	235	/oradata4/YDDY_DATA56.dbf	0	ONLINE

而当用户尝试从备份开始恢复时，发现备份无法还原出来，此前的几次备份都失败了，不可用。现在问题就相当严重了。

恢复过程

通过文件描述符进行数据恢复

这种情况下的恢复有多种可能（后面的章节还会介绍），这个用户的恢复是最简单的一种情况。恢复过程主要利用了 UNIX 系统下的“文件描述符”，通过文件描述符的指向映射恢复数据文件。

1. 通过文件描述符进行恢复

在操作系统中，内核（kernel）利用文件描述符（file descriptor）来访问文件。文件描述符是非负整数。打开现存文件或新建文件时，内核会返回一个文件描述符。读/写文件也需要使用文件描述符来指定对应的文件。

实际上文件描述符是进程态的一种组织，进程通过文件描述符将文件纳入其地址空间，然后文件描述符指向具体的文件结构。图 1 是进程与文件描述符的示意图，众多的进程都通过文件描述符进行文件的空间和地址映射。

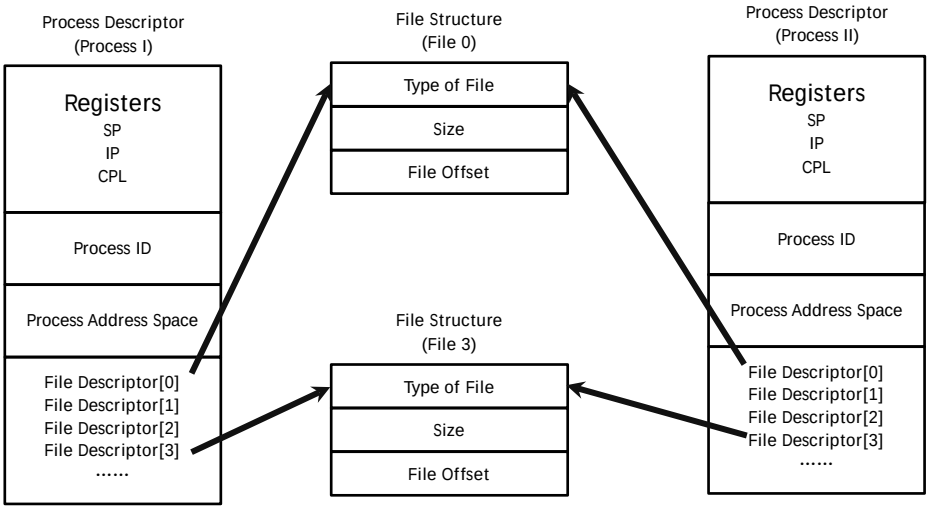

图 1 进程与文件描述符示意图

在 UNIX、Linux 系统中，误删除数据文件后，虽然该文件已从操作系统中删除，但是其文件句柄仍由数据库进程打开持有，所以在数据库层面仍然不会释放其链表信息，因而也就能够从进程的地址信息中，通过复制将其直接恢复。但是请注意，这要求数据库不能中途关闭，如果关闭了数据库，则所有文件句柄均被释放，文件就真的难以找回了。

再次重申这则案例给我们的一个重要教训：如果系统出现问题，在做出准确判断之前，绝不要贸然采取应对措施，更不能贸然关闭或者重启数据库或主机操作系统。因为在一些特殊情况下，有可能主机在关闭后无法启动，或者数据库在关闭后无法启动。

还有一个特殊的地方需要注意，在前面的文件列表输出中，18 号 UNDO 文件的状态已经由 ONLINE 变成了 RECOVER，这说明这个文件还遇到了其他问题。从 RAC 环境的节点 2 上，我们找到了相关的日志信息。

首先，用户尝试删除丢失的 UNDO 文件，遇到了错误而失败。

```
Fri Dec 30 10:12:39 2011
alter tablespace UNDOTBS2 drop datafile '/oradata4/undotbs203.dbf'
ORA-3262 signalled during:
alter tablespace UNDOTBS2 drop datafile '/oradata4/undotbs203.dbf'...
Fri Dec 30 10:13:40 2011
Errors in file /opt/oracle/app/diag/rdbms/com/com2/trace/com2_j000_7662.trc:
ORA-12012: error on auto execute of job 41
ORA-01116: error in opening database file 231
ORA-01110: data file 231: '/oradata4/undotbs203.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
```

接下来，用户尝试创建一个新的 UNDO 表空间，结果失败。这个尝试执行了多次，这**在实践中也应当尽量避免，反复尝试一个出错的命令是极其不慎重的**。

```
Fri Dec 30 10:15:07 2011
create tablespace undotbs3 datafile'/oradata4/undotbs301.dbf' size 16374m
ORA-604 signalled during:
create tablespace undotbs3 datafile'/oradata4/undotbs301.dbf' size 16374m...
Fri Dec 30 10:15:33 2011
create tablespace undotbs3 datafile '/oradata4/undotbs301.dbf' size 16374m
Fri Dec 30 10:15:51 2011
Errors in file /opt/oracle/app/diag/rdbms/com/com2/trace/com2_j000_8575.trc:
ORA-12012: error on auto execute of job 41
ORA-01116: error in opening database file 231
ORA-01110: data file 231: '/oradata4/undotbs203.dbf'
ORA-27041: unable to open file
SVR4 Error: 2: No such file or directory
Additional information: 3
Fri Dec 30 10:17:25 2011
ORA-604 signalled during:
create tablespace undotbs3 datafile '/oradata4/undotbs301.dbf' size 16374m...
```

再往后，用户发出 OFFLINE 命令，成功地将 UNDO 文件离线。这里的成功离线，也就意味着存储上的文件句柄被释放，文件被删除。此后需要通过这个 UNDO 进行回滚或一致性读的操作都会出现“无法读取文件”错误。

```
Fri Dec 30 10:23:31 2011
alter database datafile '/oradata4/undotbs203.dbf' offline
Completed: alter database datafile '/oradata4/undotbs203.dbf' offline
Fri Dec 30 10:23:39 2011
Errors in file /opt/oracle/app/diag/rdbms/com/com2/trace/com2_ora_7511.trc:
ORA-00376: 此时无法读取文件 231
ORA-01110: 数据文件 231: '/oradata4/undotbs203.dbf'
ORA-00376: 此时无法读取文件 231
ORA-01110: 数据文件 231: '/oradata4/undotbs203.dbf'
Fri Dec 30 10:23:52 2011
Errors in file /opt/oracle/app/diag/rdbms/com/com2/trace/com2_ora_7511.trc:
ORA-00376: 此时无法读取文件 231
ORA-01110: 数据文件 231: '/oradata4/undotbs203.dbf'
ORA-00376: 此时无法读取文件 231
ORA-01110: 数据文件 231: '/oradata4/undotbs203.dbf'
```

2. 关于文件状态的说明

关于数据文件的离线和状态，在此需要做一个简要的说明。前面我们注意到了文件的两种状态：ONLINE 和 RECOVER。对于数据文件，还有两种状态，分别是 SYSTEM 和 OFFLINE，系统表空间的状态为 SYSTEM。

下面是一个正常的数据库。

```
SQL> select name,status from v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/home/ora11g/oradata/orcl11g/system01.dbf          SYSTEM
/home/ora11g/oradata/orcl11g/sysaux01.dbf          ONLINE
/home/ora11g/oradata/orcl11g/undotbs01.dbf         ONLINE
/home/ora11g/oradata/orcl11g/users01.dbf           ONLINE
```

在归档模式下，对于文件的离线操作，会将文件状态标记为 RECOVER（正因为存在归档，才存在恢复的

可能性），文件的离线不会执行表空间级别的检查点。为了维持表空间级别的一致性，在对文件 ONLINE 时必须要执行恢复。而如果是基于表空间进行，则表空间检查点会被执行，在文件进一步 ONLINE 时，就不需要进行 RECOVER。

下面对文件执行文件级别的 OFFLINE 操作。

```
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     120
Next log sequence to archive   122
Current log sequence           122
SQL> alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' offline;
Database altered.
SQL> select name,status from v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/home/ora11g/oradata/orcl11g/system01.dbf          SYSTEM
/home/ora11g/oradata/orcl11g/sysaux01.dbf          ONLINE
/home/ora11g/oradata/orcl11g/undotbs01.dbf         ONLINE
/home/ora11g/oradata/orcl11g/users01.dbf           RECOVER
SQL> alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' online;
alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' online
*
ERROR at line 1:
ORA-01113: file 4 needs media recovery
ORA-01110: data file 4: '/home/ora11g/oradata/orcl11g/users01.dbf'
SQL> recover datafile 4;
Media recovery complete.
SQL> alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' online;
Database altered.
```

对于表空间级别的离线和在线加载，则可以顺利地进行。

```
SQL> alter tablespace users offline;
```

```
Tablespace altered.
SQL> select name,status from v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/home/ora11g/oradata/orcl11g/system01.dbf          SYSTEM
/home/ora11g/oradata/orcl11g/sysaux01.dbf          ONLINE
/home/ora11g/oradata/orcl11g/undotbs01.dbf         ONLINE
/home/ora11g/oradata/orcl11g/users01.dbf           OFFLINE
SQL> alter tablespace users online;
Tablespace altered.
```

而在非归档模式下，是不允许对文件级别执行离线操作的，可以改用 OFFLINE DROP 对文件进行离线操作。但如果后期的日志不可用，则这个离线的文件可能永远无法正常 ONLINE 加载。

```
SQL> alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' offline;
alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' offline
*
ERROR at line 1:
ORA-01145: offline immediate disallowed unless media recovery enabled
SQL> alter database datafile '/home/ora11g/oradata/orcl11g/users01.dbf' offline drop;
Database altered.
SQL> select name,status from v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/home/ora11g/oradata/orcl11g/system01.dbf          SYSTEM
/home/ora11g/oradata/orcl11g/sysaux01.dbf          ONLINE
/home/ora11g/oradata/orcl11g/undotbs01.dbf         ONLINE
/home/ora11g/oradata/orcl11g/users01.dbf           RECOVER
```

至于 v$datafile 中的状态定义，来自以下 SQL 查询，从中可以看到 Oracle 对于状态的标示与定义。通过视图的底层创建语句分析 Oracle 上层体现，是学习 Oracle 技术的重要方法之一。这些视图的定义可以通过 v$fixed_view_definition 来获得。

```
select fe.inst_id,
       fe.fenum,
       to_number(fe.fecrc_scn),
```

```
to_date(fe.fecrc_tim,
          'MM/DD/RR HH24:MI:SS',
          'NLS_CALENDAR=Gregorian'),
fe.fetsn,
fe.ferfn,
-- 此处即文件状态的定义信息，其内容来自 x$kccfe 结构
-- X$KCCFE - File [E]ntries ( from control file )
decode(fe.fetsn,
          0,
          decode(bitand(fe.festa, 2), 0, 'SYSOFF', 'SYSTEM'),
          decode(bitand(fe.festa, 18),
                    0,'OFFLINE', 2, 'ONLINE','RECOVER')),
decode(fe.fedor, 2, 'READ ONLY',
          decode(bitand(fe.festa, 12),0,'DISABLED',4,'READ ONLY',
                    12,'READ WRITE','UNKNOWN')),
to_number(fe.fecps),
to_date(fe.fecpt, 'MM/DD/RR HH24:MI:SS', 'NLS_CALENDAR=Gregorian'),
to_number(fe.feurs),
to_date(fe.feurt, 'MM/DD/RR HH24:MI:SS', 'NLS_CALENDAR=Gregorian'),
to_number(fe.fests),
decode(fe.fests,
          NULL, to_date(NULL),
          to_date(fe.festt,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian')),
to_number(fe.feofs),
to_number(fe.feonc_scn),
to_date(fe.feonc_tim, 'MM/DD/RR HH24:MI:SS', 'NLS_CALENDAR=Gregorian'),
fh.fhfsz * fe.febsz,
fh.fhfsz,
fe.fecsz * fe.febsz,
fe.febsz,
fn.fnnam,
fe.fefdb,
fn.fnbof,
```

```
        decode(fe.fepax, 0, 'UNKNOWN', 65535, 'NONE', fnaux.fnnam),
        to_number(fh.fhfirstunrecscn),
        to_date(fh.fhfirstunrectime, 'MM/DD/RR HH24:MI:SS', 'NLS_CALENDAR=Gregorian'),
        fe.fepdi,
        fe.fefcrs,
        fe.fefcrt,
        decode(fe.fefdb, 1, 'YES', 'NO'),
        fe.feplus,
        fe.feprls,
        fe.feprlt
  from x$kccfe fe, x$kccfn fn, x$kccfn fnaux, x$kcvfh fh
 where ((fe.fepax != 65535 and fe.fepax != 0 and fe.fepax = fnaux.fnnum) or
        ((fe.fepax = 65535 or fe.fepax = 0) and fe.fenum = fnaux.fnfno and
        fnaux.fntyp = 4 and fnaux.fnnam is not null and
        bitand(fnaux.fnflg, 4) != 4 and fe.fefnh = fnaux.fnnum))
    and fn.fnfno = fe.fenum     and fn.fnfno = fh.hxfil
    and fe.fefnh = fn.fnnum     and fe.fedup != 0
    and fn.fntyp = 4     and fn.fnnam is not null
    and bitand(fn.fnflg, 4) != 4
 order by fe.fenum
```

3. 具体案例的恢复过程

几乎同样的案例，2011 年 12 月 18 日我们为用户处理过另外一则，这里一并介绍一下。这一案例同样是归档日志由于空间不足无法写出，同样是在接近凌晨时出现问题，也同样是因为误操作删除了整个目录。

以下日志摘录显示故障时间是在 23：41，午夜时分，错误提示为空间满，无法写归档日志，错误由归档进程抛出。

```
Sun Dec 18 23:41:15 2011
Errors in file /u01/admin/sxnms/bdump/sxnms_arc0_784.trc:
ORA-19504: failed to create file "/ora_backup/arch/1_14289.dbf"
ORA-27044: unable to write the header block of file
SVR4 Error: 28: No space left on device
```

```
Additional information: 1
ARC0: Archiving not possible: error count exceeded
Sun Dec 18 23:41:16 2011
ARC1: I/O error 19502 archiving log 4 to '/ora_backup/arch/1_14288.dbf'
Sun Dec 18 23:41:16 2011
Errors in file /u01/admin/sxnms/bdump/sxnms_arc1_788.trc:
ORA-19502:
write error on file "/ora_backup/arch/1_14288.dbf", blockno 63489 (blocksize=512)
ORA-27063: skgfospo: number of bytes read/written is incorrect
Additional information: 7680
Additional information: 1048576
ORA-19502:
write error on file "/ora_backup/arch/1_14288.dbf", blockno 57345 (blocksize=512)
ARC1: Archiving not possible: error count exceeded
```

在以上错误提示中，有一个关于日志块大小的说明，Oracle 的日志块大小是 512 字节，与数据库文件块大小不同。注意，DBV 工具不能用于检查日志文件。

以上问题出现后，经过一系列处理（删文件），日志最终可以继续归档，这显然也是得益于空间释放。

```
Mon Dec 19 01:07:35 2011
ARC1: Evaluating archive   log 4 thread 1 sequence 14288
ARC1: Beginning to archive log 4 thread 1 sequence 14288
Creating archive destination LOG_ARCHIVE_DEST_1: '/ora_backup/arch/1_14288.dbf'
Mon Dec 19 01:07:36 2011
ARC0: Evaluating archive   log 4 thread 1 sequence 14288
ARC0: Unable to archive log 4 thread 1 sequence 14288
      Log actively being archived by another process
ARC0: Evaluating archive   log 5 thread 1 sequence 14289
ARC0: Beginning to archive log 5 thread 1 sequence 14289
Creating archive destination LOG_ARCHIVE_DEST_1: '/ora_backup/arch/1_14289.dbf'
Mon Dec 19 01:07:39 2011
ARC1: Completed archiving  log 4 thread 1 sequence 14288
Archiver process freed from errors. No longer stopped
```

但是紧急处理及后续处理出现了误操作，大量文件被误删除，具体的删除时间不确定，不过日志中出现的

提示是在中午 12 点左右。首先报告的错误是日志文件无法访问，文件状态不能获得。

```
Mon Dec 19 11:52:11 2011
ARCH: Evaluating archive   log 5 thread 1 sequence 14294
Mon Dec 19 11:52:11 2011
Errors in file /u01/admin/cinms/udump/cinms_ora_16062.trc:
ORA-00313: open failed for members of log group 5 of thread 1
ORA-00312: online log 5 thread 1: '/u02/oradata/cinms_redo05.log'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
ARCH: Error 313 opening/verifying online redo log 5
Mon Dec 19 11:52:11 2011
Errors in file /u01/admin/cinms/udump/cinms_ora_16062.trc:
ORA-00313: open failed for members of log group 5 of thread 1
ORA-00312: online log 5 thread 1: '/u02/oradata/cinms_redo05.log'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
Additional information: 3
Mon Dec 19 11:52:11 2011
ARC1: Evaluating archive   log 5 thread 1 sequence 14294
ARC1: Unable to archive log 5 thread 1 sequence 14294
      Log actively being archived by another process
```

随后用户确认 u02 目录下的文件全被删除。此次事件中共有 24 个数据文件受到影响，都是极其重要的用户数据。由于日志文件也存储在 u02 目录中，因而在进行切换写入时，首先由日志文件报告了写错误。

以下是丢失的数据文件列表。

FILE_ID	FILE_NAME	TABLESPACE_NAME	MB
8	/u02/oradata/sxnms_user01.dbf	SXNMS_USER	1000
9	/u02/oradata/sxnms_user02.dbf	SXNMS_USER	1000
10	/u02/oradata/sxnms_user03.dbf	SXNMS_USER	1000
11	/u02/oradata/sxnms_user04.dbf	SXNMS_USER	1000
27	/u02/oradata/sxnms_user07.dbf	SXNMS_USER	1000

```
28 /u02/oradata/sxnms_user08.dbf            SXNMS_USER                1000
29 /u02/oradata/sxnms_user09.dbf            SXNMS_USER                1024
30 /u02/oradata/sxnms_user10.dbf            SXNMS_USER                1024
31 /u02/oradata/sxnms_user11.dbf            SXNMS_USER                1024
36 /u02/oradata/sxnms_user15.dbf            SXNMS_USER                1000
37 /u02/oradata/sxnms_user16.dbf            SXNMS_USER                1000
50 /u02/oradata_res/data_res09.dbf          DATA_RES                  1000
51 /u02/oradata_res/data_res10.dbf          DATA_RES                  1000
52 /u02/oradata_res/data_res11.dbf          DATA_RES                  1000
53 /u02/oradata_res/data_res12.dbf          DATA_RES                  1000
55 /u02/oradata/sxnms_backup03              SXNMS_BACKUP              1314
56 /u02/oradata/sxnms_backup04              SXNMS_BACKUP            1313.5
61 /u02/oradata/sxnms_indexes05.dbf         SXNMS_INDEXES             1000
63 /u02/oradata/sxnms_user25.dbf            SXNMS_USER                1000
67 /u02/oradata/sxnms_indexes09.dbf         SXNMS_INDEXES             1000
68 /u02/oradata/sxnms_user27.dbf            SXNMS_USER                1000
86 /u02/oradata/sxnms_indexes10.dbf         SXNMS_INDEXES             1000
87 /u02/oradata/sxnms_indexes11.dbf         SXNMS_INDEXES             1024
88 /u02/oradata/sxnms_indexes12.dbf         SXNMS_INDEXES             1024

24 rows selected.
```

用户在这种情况下做出了正确的判断，未关闭数据库，并且和我们取得了联系。这种情况，是完全可以利用文件描述符的方式来进行恢复的。

首先找到一个后台进程（如 DBWR 进程），通过其进程地址找到文件句柄（/proc/<proc_id>/fd）。以下就是数据库文件的句柄显示信息，复制这些文件即可恢复那些被删除但尚未消失的数据文件。

```
bash-2.05$ ps -ef | grep dbw | grep -v grep
  oracle   762     1  0   Jun 10 ?       217:30 ora_dbw0_sxnms
bash-2.05$ ls  /proc/762/fd
0    12   256  260  264  268  272  276  280  284  288  292  296  3    303  307  311
315  319  323  327  331  335  339  343  347  61   13   257  261  265  269  273  277
281  285  289  293  297  300  304  308  312  316  320  324  328  332  336  340  344
348  710  14   258  262  266  270  274  278  282  286  290  294  298  301  305  309
```

```
313  317  321  325  329  333  337  341  345  4    811  2    259  263  267  271  275
279  283  287  291  295  299  302  306  310  314  318  322  326  330  334  338  342
346  5    9
db2% ls -l | grep oracle
-r--r--r--   1 oracle   dba       657920 Apr 26  2002 11
-rw-r-----   1 oracle   dba          923 Jun 10  2011 12
-rw-rw----   1 oracle   dba           24 Jun 10  2011 13
-rw-r-----   1 oracle   dba      1859584 Jan  5 16:42 256
-rw-r-----   1 oracle   dba      1859584 Jan  5 16:42 257
-rw-r-----   1 oracle   dba      1859584 Jan  5 16:42 258
-rw-r-----   1 oracle   dba    414195712 Jan  5 16:41 259
-rw-r-----   1 oracle   dba   6291464192 Jan  5 16:42 260
-rw-r-----   1 oracle   dba    161488896 Jan  5 16:18 261
-rw-r-----   1 oracle   dba     20979712 Jan  5 16:18 262
-rw-r-----   1 oracle   dba     26222592 Jan  5 16:18 263
-rw-r-----   1 oracle   dba     10493952 Jan  5 16:18 264
-rw-r-----   1 oracle   dba    473178112 Jan  5 16:18 265
-rw-r-----   1 oracle   dba   1048584192 Jan  5 16:40 266
-rw-r-----   1 oracle   dba   1048584192 Jan  5 16:18 267
-rw-r-----   1 oracle   dba   1048584192 Jan  5 16:40 268
-rw-r-----   1 oracle   dba   1048584192 Jan  5 16:40 269
-rw-r-----   1 oracle   dba   1048584192 Jan  5 16:41 270
-rw-r-----   1 oracle   dba   1048584192 Jan  5 16:42 271
-rw-r-----   1 oracle   dba   1384652800 Jan  5 16:18 272
```

本例中，我们在复制文件恢复之后，创建了一个新的目录（保留原来的目录结构不动），随后通过 offline、rename、recover、online 四个步骤恢复这些文件，将其加载到数据库中。以下是简要步骤记录。

首先复制文件到新分配的目录空间。

```
cp  /proc/762/fd/266  /new_u02/oradata/cinms_user01.dbf
cp  /proc/762/fd/267  /new_u02/oradata/cinms_user02.dbf
cp  /proc/762/fd/268  /new_u02/oradata/cinms_user03.dbf
cp  /proc/762/fd/269  /new_u02/oradata/cinms_user04.dbf
cp  /proc/762/fd/285  /new_u02/oradata/cinms_user07.dbf
```

```
cp  /proc/762/fd/286  /new_u02/oradata/cinms_user08.dbf
```

将相应的文件离线。

```
alter  database  datafile  8  offline;
alter  database  datafile  9  offline;
alter  database  datafile  10  offline;
alter  database  datafile  11  offline;
alter  database  datafile  27  offline;
alter  database  datafile  28  offline;
```

通过更名（RENAME）的方式对文件进行重定向。

```
alter  database  rename  file
'/u02/oradata/cinms_user01.dbf'  to  ' /new_u02/oradata/cinms_user01.dbf';
alter  database  rename  file
'/u02/oradata/cinms_user02.dbf'  to  '/new_u02/oradata/cinms_user02.dbf';
alter  database  rename  file
'/u02/oradata/cinms_user03.dbf'  to  '/new_u02/oradata/cinms_user03.dbf';
alter  database  rename  file
'/u02/oradata/cinms_user04.dbf'  to  '/new_u02/oradata/cinms_user04.dbf';
alter  database  rename  file
'/u02/oradata/cinms_user07.dbf'  to  '/new_u02/oradata/cinms_user07.dbf';
alter  database  rename  file
'/u02/oradata/cinms_user08.dbf'  to  '/new_u02/oradata/cinms_user08.dbf';
```

然后执行恢复。

```
recover  datafile  8;
recover  datafile  9;
recover  datafile  10;
recover  datafile  11;
recover  datafile  27;
recover  datafile  28;
```

最后将文件 Online 加载。

```
alter  database  datafile  8  online;
alter  database  datafile  9  online;
```

```
alter database datafile 10 online;
alter database datafile 11 online;
alter database datafile 27 online;
alter database datafile 28 online;
```

以下是日志中记录的操作信息。

```
Mon Dec 19 18:17:38 2011
alter database datafile 8 offline
Mon Dec 19 18:17:39 2011
Completed: alter database datafile 8 offline
Mon Dec 19 18:18:04 2011
alter database rename file '/u02/oradata/sxnms_user01.dbf'
                  to '/new_u02/oradata/sxnms_user01.dbf'
Mon Dec 19 18:18:20 2011
ALTER DATABASE RECOVER  datafile 8
Media Recovery Datafile: 8
Media Recovery Start
Starting datafile 8 recovery in thread 1 sequence 14295
Datafile 8: '/new_u02/oradata/sxnms_user01.dbf'
Media Recovery Log
Recovery of Online Redo Log: Thread 1 Group 1 Seq 14295 Reading mem 0
  Mem# 0 errs 0: /u01/oradata/sxnms/redo01.log
Media Recovery Complete
Completed: ALTER DATABASE RECOVER  datafile 8

Mon Dec 19 18:19:00 2011
alter database datafile 8 online
Completed: alter database datafile 8 online
```

这两则案例，处境相同，处理相同，结果也大致相同。由于数据库保持在启动状态，因而能够较为快速地恢复出被删除的数据文件，实在是一种幸运。

技术难点

在这两则案例中，还有一个关键的技术点，那就是如何从大量的 fd 中找到需要的文件。在 Oracle 数据库文件的第一个块（文件头块）上，存有数据文件号信息，只要找到这个文件号，就能和数据库建立起对应关系。

通过 BBED 获取文件号信息

通过 Oracle 的 BBED（Block Browse/EDit）工具，可以很容易获得这个信息。在 UNIX、Linux 系统上，需要通过 make 进行简单编译，生成对应的可执行文件。

```
[oracle@hpserver2 ~]$ cd $ORACLE_HOME/rdbms/lib
[oracle@hpserver2 lib]$ make -f ins_rdbms.mk $ORACLE_HOME/rdbms/lib/bbed
Linking BBED utility (bbed)
rm -f /u01/app/oracle/product/10.2.0/db_1/rdbms/lib/bbed
gcc -o
/u01/app/oracle/product/10.2.0/db_1/rdbms/lib/bbed
-L/u01/app/oracle/product/10.2.0/db_1/rdbms/lib/
-L/u01/app/oracle/product/10.2.0/db_1/lib/
-L/u01/app/oracle/product/10.2.0/db_1/lib/stubs/
......
```

然后就可以通过该工具（密码是 blockedit）来获得文件号。以下是一个示范。

```
[oracle@hpserver2 fd]$ ls
0  10  12  14  16  18  2   21  23  3  5  7  9
1  11  13  15  17  19  20  22  24  4  6  8
[oracle@hpserver2 ]$ ./bbed
Password: blockedit
BBED: Release 2.0.0.0.0 - Limited Production on Thu Jan 5 16:54:36 2012
Copyright (c) 1982, 2007, Oracle.  All rights reserved.
************* !!! For Oracle Internal Use only !!! ***************
BBED> set filename '/proc/23330/fd/18'
     FILENAME          /proc/23330/fd/18
BBED> set blocksize 8192
     BLOCKSIZE         8192
```

```
BBED> p kcvfh.kcvfhrfn
ub4 kcvfhrfn                                    @368       0x00000001
BBED> set filename '/proc/23330/fd/19'
     FILENAME          /proc/23330/fd/19
BBED> p kcvfh.kcvfhrfn
ub4 kcvfhrfn                                    @368       0x00000002
```

这里的几个数据结构需要说明一下，kcvfh 表示 Kernel Cache recoVery component File Header。前三个字母 kcv 表示是 Oracle 的内核层次代码，是恢复相关的组件，具体内容就是文件头信息。Kcvfhrfn 表示相对文件号。

关于文件头的信息，在数据库层面的展示来自视图 v$datafile_header，以下是该视图的字段信息。

```
SQL> desc v$datafile_header
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FILE#                                              NUMBER
 STATUS                                             VARCHAR2(7)
 ERROR                                              VARCHAR2(18)
 FORMAT                                             NUMBER
 RECOVER                                            VARCHAR2(3)
 FUZZY                                              VARCHAR2(3)
 CREATION_CHANGE#                                   NUMBER
 CREATION_TIME                                      DATE
 TABLESPACE_NAME                                    VARCHAR2(30)
 TS#                                                NUMBER
 RFILE#                                             NUMBER
 RESETLOGS_CHANGE#                                  NUMBER
 RESETLOGS_TIME                                     DATE
 CHECKPOINT_CHANGE#                                 NUMBER
 CHECKPOINT_TIME                                    DATE
 CHECKPOINT_COUNT                                   NUMBER
 BYTES                                              NUMBER
 BLOCKS                                             NUMBER
 NAME                                               VARCHAR2(513)
 SPACE_HEADER                                       VARCHAR2(40)
```

```
LAST_DEALLOC_CHANGE#                              VARCHAR2(16)
UNDO_OPT_CURRENT_CHANGE#                          VARCHAR2(40)
```

该视图（v$fixed_view_definition 视图是可以获得其他视图结构定义的字典）经由 X$KCVFH 创建，创建语句如下，X$KCVFH 正是来自文件头的结构体内容。

```
SQL> select view_definition
  2  from v$fixed_view_definition where view_name='GV$DATAFILE_HEADER';
VIEW_DEFINITION
--------------------------------------------------------------------------------
SELECT inst_id,
  hxfil,
  DECODE(hxons, 0, 'OFFLINE', 'ONLINE'),
  DECODE(hxerr, 0, NULL, 1,'FILE MISSING',2,'OFFLINE NORMAL', 3,'NOT VERIFIED',
4,'FILE NOT FOUND',5,'CANNOT OPEN FILE', 6,'CANNOT READ HEADER', 7,'CORRUPT
HEADER',8,'WRONG FILE TYPE', 9,'WRONG DATABASE', 10,'WRONG FILE NUMBER',11,'WRONG
FILE CREATE', 12,'WRONG FILE CREATE', 16,'DELAYED OPEN',14, 'WRONG RESETLOGS',
15,'OLD CONTROLFILE', 'UNKNOWN ERROR'),
  hxver,
  DECODE(hxnrcv, 0,'NO', 1,'YES', NULL),
  DECODE(hxifz, 0,'NO', 1,'YES', NULL),
  to_number(fhcrs),
  to_date(fhcrt,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
  fhtnm,
  fhtsn,
  fhrfn,
  to_number(fhrls),
  to_date(fhrlc,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
  to_number(fhscn),
  to_date(fhtim,'MM/DD/RR HH24:MI:SS','NLS_CALENDAR=Gregorian'),
  fhcpc,
  fhfsz*fhbsz,
  fhfsz,
  hxfnm,
  DECODE(hxlmdba, 0, NULL, hxlmdba),
```

```
  DECODE(hxlmld_scn, to_number('0'), NULL, hxlmld_scn),
  DECODE(hxuopc_scn, 0, NULL, hxuopc_scn)
FROM x$kcvfh
```

通过 v$datafile_header 获得的信息来自数据文件头，其中至关重要的有两个：数据文件的创建 SCN 和创建时间。

```
SQL> select file#,creation_change#,creation_time from v$datafile_header;
     FILE# CREATION_CHANGE# CREATION_TIME
---------- ---------------- -------------------
         1                7 2012-01-06 23:31:51
         2             1957 2012-01-06 23:32:14
         3             3103 2012-01-06 23:32:24
         4             3500 2012-01-06 23:32:32
```

文件的创建 SCN 和时间还记录在 file$视图中，file$中的信息在启动时用于和数据文件头进行比对校验。

```
SQL> select file#,crscnbas,status$,blocks,spare1 from file$;
     FILE#    CRSCNBAS    STATUS$     BLOCKS     SPARE1
---------- ---------- ---------- ---------- ----------
         1          7          2      51200    4194306
         2       1957          2     102400    8388610
         3       3103          2      25600   12582914
         4       3500          2      12800   16777218
         5     241976          1       1280   20971522
```

在后面的章节我们还会接触到这些信息。

在 BBED 工具中，首先应用到了 PRINT 命令，该命令可以缩写为 p，用于打印显示数据块中的指定数据结构、子结构及偏移量信息等，也可用于打印具体的存储数据行等。前面使用的 **p kcvfh.kcvfhrfn** 就用于显示相对文件号信息。

关于 BBED 的进一步说明，参见本书附录。

通过 od 命令获得文件号信息

另外一个简单的方式是通过操作系统的 od 命令读出文件的指定位置，获得文件的文件号。

首先使用 BBED 取得一个参照输出。

```
dtdb2_oracle%bbed
Password:
BBED: Release 2.0.0.0.0 - Limited Production on Thu Jan 5 18:05:30 2012
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
************* !!! For Oracle Internal Use only !!! ***************
BBED> set filename '/proc/762/fd/340'
BBED-00307: incorrect blocksize (2048) or truncated file
BBED> set blocksize 8192
        BLOCKSIZE       8192
BBED> set filename '/proc/762/fd/340'
        FILENAME
BBED> p kcvfh.kcvfhrfn
ub4 kcvfhrfn                                    @280       0x00000052
BBED> exit
```

注意 Oracle 9i 文件号的偏移量是 280，再加上数据文件头上一个操作系统块，od 跳过 8472 即可获得文件号。以下是 16 进制输出。

```
dtdb2_oracle%od -j 8472 -t x1 340 | head -1
0000000 00 00 00 52 00 00 00 00 00 00 00 00 00 00 00 00
```

以下是 10 进制输出，文件号为 82。

```
dtdb2_oracle%od -j 8472 -t d2 340 | head -1
0000000   00000   00082   00000   00000   00000   00000   00000   00000
```

在 Oracle 10g/11g 中，文件号的偏移量为 368。以下是在 Linux 平台上 od 命令的 10 进制输出，考察的 3 个文件文件号分别为 1、2 和 3。

```
[oracle@hpserver2 fd]$ od -j 8560 -t x1 18| head -1
0020560 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[oracle@hpserver2 fd]$ od -j 8560 -t d2 18| head -1
0020560       1       0       0       0       0       0       0       0
[oracle@hpserver2 fd]$ od -j 8560 -t d2 19| head -1
0020560       2       0       0       0       0       0       0       0
[oracle@hpserver2 fd]$ od -j 8560 -t d2 20| head -1
```

```
0020560      3      0      0      0      0      0      0      0
```

如果不熟悉 BBED 的用法，像上面这样使用操作系统的 od 命令，也可以非常简单、快速地获得文件号。

除 BBED 外，在 Linux 和 UNIX 上，还可以通过 lsof 工具查看进程的 FD 等信息。举个例子，可以首先选择一个后台进程（如 dbwr 进程），获取其进程号。

```
root@db2 # ps -ef | grep dbw
    root 14348  9859  0 10:15:50 pts/3    0:00 grep dbw
  oracle   762     1  0   Jun 10 ?      202:01 ora_dbw0_nms
```

然后通过 lsof 工具列举进程打开的文件，输出的内容中就包含了 FD 一列。

```
root@db2 # lsof -p 762
COMMAND PID   USER   FD   TYPE        DEVICE   SIZE/OFF   NODE NAME
oracle  762 oracle  259uW VREG       85,8272  414195712 110986 /u01/nms/system01.dbf
oracle  762 oracle  260uW VREG       85,8272 6291464192 110993 /u01/nms/undotbs01.dbf
oracle  762 oracle  277uW VREG       85,8242 1048584192      9
/u04/res/data_res02.dbf
oracle  762 oracle  287uW VREG         118,0 1048584192 192538 /u02/cnnc_user01.dbf
oracle  762 oracle  288uW VREG         118,0 1048584192 192539 /u02/cnnc_user02.dbf
oracle  762 oracle  289uW VREG         118,0 1048584192 192540 /u02/cnnc_user03.dbf
oracle  762 oracle  290uW VREG       85,8252 1048584192     13 /u03/cnnc_user12.dbf
oracle  762 oracle  291uW VREG       85,8252 1048584192     14 /u03/cnnc_user13.dbf
oracle  762 oracle  325uW VREG         118,0 1048584192 192557 / (/dev/dsk/c1t0d0s0)
oracle  762 oracle  326uW VREG         118,0 1048584192 192558 /u02/cnnc_user27.dbf
oracle  762 oracle  327uW VREG       85,8252 1048584192     27 /u03/cnnc_user28.dbf
oracle  762 oracle  328uW VREG       85,8242 1048584192     24 /u04/cnnc_user29.dbf
oracle  762 oracle  329uW VREG       85,8242 1048584192     25 /u04/cnnc30.dbf
oracle  762 oracle  340uW VREG       85,8242 1048584192     26 /u04/oradata/res13.dbf
oracle  762 oracle  341uW VREG       85,8242 1048584192     27 /u04 (/dev/md/dsk/d50)
oracle  762 oracle  342uW VREG       85,8252 1048584192     28
/u03/oradata/idx_res02.dbf
oracle  762 oracle  343uW VREG       85,8272 1409294336 111241 /u01 (/dev/md/dsk/d80)
oracle  762 oracle  345uW VREG         118,0 1073750016 192560
/u02/cnnc_indexes11.dbf
oracle  762 oracle  346uW VREG         118,0 1073750016 192561
```

```
/u02/cnnc_indexes12.dbf
```

通过 proc 下的文件目录，可以找到文件句柄信息，这就是 Oracle 的数据文件。

```
root@db2 # ls -l /proc/762/fd/329
-rw-r-----   1 oracle   dba      1048584192 Dec 20 10:16 /proc/762/fd/329
```

可以通过复制来恢复文件。以下是 Linux 上的一个测试范例，误删除的文件显示为 **deleted**。

```
$ cd /proc/2879/fd
$ ls -l
lr-x------ 1 oracle dba 64 Dec 19 21:50 12 ->
/oracle/10.2.0/db_1/rdbms/mesg/oraus.msb
lrwx------ 1 oracle dba 64 Dec 19 21:50 13 -> /oracle/10.2.0/db_1/dbs/hc_orcl.dat
lrwx------ 1 oracle dba 64 Dec 19 21:50 14 -> /oracle/10.2.0/db_1/dbs/lkORCL
lrwx------ 1 oracle dba 64 Dec 19 21:50 15 ->
/oradata/controlfile/o1_mf_555wq3ng_.ctl
lrwx------ 1 oracle dba 64 Dec 19 21:50 16 ->
/oradata/datafile/o1_mf_system_555wqbnk_.dbf
lrwx------ 1 oracle dba 64 Dec 19 21:50 17 -> /oradata/datafile/o1_mf_undotbs1.dbf
lrwx------ 1 oracle dba 64 Dec 19 21:50 18 -> /oradata/datafile/sysaux_555wr.dbf
lrwx------ 1 oracle dba 64 Dec 19 21:50 19 -> /oradata/datafile/users.dbf (deleted)
lr-x------ 1 oracle dba 64 Dec 19 21:50 2 -> /dev/null
lrwx------ 1 oracle dba 64 Dec 19 21:50 20 ->
/oradata/datafile/o1_mf_temp_555wrbnz_.tmp
lr-x------ 1 oracle dba 64 Dec 19 21:50 21 ->
/oracle/10.2.0/db_1/rdbms/mesg/oraus.msb
l-wx------ 1 oracle dba 64 Dec 19 21:50 5 -> /admin/udump/orcl_ora_2871.trc
l-wx------ 1 oracle dba 64 Dec 19 21:50 6 -> /admin/bdump/alert_orcl.log
lrwx------ 1 oracle dba 64 Dec 19 21:50 7 -> /oracle/10.2.0/db_1/dbs/lkinstorcl
(deleted)
l-wx------ 1 oracle dba 64 Dec 19 21:50 8 -> /admin/bdump/alert_orcl.log
lrwx------ 1 oracle dba 64 Dec 19 21:50 9 -> /oracle/10.2.0/db_1/dbs/hc_orcl.dat
```

lsof 是一个非常直观和方便的工具，在很多情况下可以给我们提供直接的帮助。