```
SQL*Plus: Release 10.2.0.4.0 - Production on Mon Jun 13 18:45:54 2011
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, Real Application Clusters, Oracle Label Security, OLAP,
Data Mining and Real Application Testing options
SQL> set pages 100 lines 120
SQL> select group_number, name, state, total_mb, free_mb from v$asm_diskgroup;
GROUP_NUMBER NAME          STATE          TOTAL_MB    FREE_MB
------------ ------------ ------------- ---------- ----------
1            DATAG1        DISMOUNT      0           0
2            DATAG2        MOUNTED       1024000     347457
SQL> select group_number, disk_number, mount_status, header_status, name, path
  2  from v$asm_disk;
GROUP_NUMBER DISK_NUMBER MOUNT_STATUS HEADER_STATUS  NAME        PATH
------------ ----------- ------------ -------------- ----------- --------------
0            0           CLOSED       FOREIGN                    /dev/raw/raw1
0            1           CLOSED       FOREIGN                    /dev/raw/raw2
0            2           CLOSED       FOREIGN                    /dev/raw/raw8
0            3           CLOSED       FOREIGN                    /dev/raw/raw10
0            4           CLOSED       FOREIGN                    /dev/raw/raw9
0            8           CLOSED       MEMBER                     /dev/raw/raw4
0            5           CLOSED       MEMBER                     /dev/raw/raw5
2            1           CACHED       MEMBER         DATAG2_0001 /dev/raw/raw7
2            0           CACHED       MEMBER         DATAG2_0000 /dev/raw/raw66
SQL> alter diskgroup datag1 mount;
Diskgroup altered.
SQL> shutdown immediate
ASM diskgroups dismounted
ASM instance shutdown
SQL> startup
ASM instance started
Total System Global Area 130023424 bytes
```

```
Fixed Size 2082208 bytes
Variable Size 102775392 bytes
ASM Cache 25165824 bytes
ASM diskgroups mounted
SQL> exit
```

利用 merge 修改磁盘头后，磁盘状态正常，磁盘组顺利挂载。此时检查节点 1 上的磁盘头信息，状态也恢复了正常。

在运行 merge 命令的时候，整个磁盘组并没有卸载，数据库也在正常运行。也就是说，在整个操作过程中，并没有一秒钟的停机时间，所有的修改都是 ONLINE 进行的。

ASM 数据抽取恢复

通过 AMDU 恢复数据案例一则

2012 年 2 月 21 日，一个香港用户的 ASM 破坏，请求数据恢复。

灾难描述

这则案例是由于存储误操作引起的。

1. 用户进行存储维护和磁盘添加操作。
2. 维护后发现 CRS 无法启动。
3. 检查发现 OCR 盘损坏，ASM 磁盘组受损。
4. 经用户反复确认，故障原因是误操作磁盘导致的 ASM 磁盘受损。
5. 为减少意外，客户请求在不更改配置等的情况下安全抽取数据。
6. 数据库为 3 节点 RAC 系统。

灾难再一次由于疏忽而降临。

案例警示

这类案例我们已经遭遇了很多，在这里再次郑重提示以下内容。

1. 存储的使用分配应当自始至终建立和维护详细的档案。

鉴于众多惨痛的数据灾难，结合标准化流程要求，我们建议用户对于数据库的存储规划、分配，建立详细的档案记录，在进行后续的维护操作时，严格通过文档进行对比确认，杜绝低级的误操作行为。

标准化和文档维护不仅仅是流程和管理的需要，也是对技术人员屏蔽错误、保障数据安全的基本要求。我们不能够把文档当做过场或可有可无的摆设，必须将其上升到数据安全的保障层面。

2. 涉及存储的调整，必须多部门协同反复确认。

由于底层存储对于数据库的核心作用，因而必须在进行维护时反复确认维护计划，多部门统一协调、共同确认，避免流程割裂导致的误操作行为。

常规的存储维护在企业中被定义为系统或存储部门的责任，但是运行于其上的数据库是数据存储的核心使用者，在存储维护的过程中，一定要数据部门介入和确认。对于 Oracle 数据库来说，由于前期的存储划分可能非常零散，包括 OCR、VOTING、REDO、DATA 等都可能存在独立的存储分区，所以如果不进行严格管理，在后期维护中就可能对其中的部分存储卷产生误操作，导致破坏。

当然，关于备份的重要性，如何强调都不为过，始终保有有效的备份才能够在出现问题时有备无患。

技术回放

对于这个案例，我们有多种手段可以恢复，只要 ASM 磁盘组完好，就可以很容易地从中提取数据。本案例我们使用了 AMDU 工具进行恢复。

AMDU 工具

在 Oracle 10g 中，ASM 磁盘组的信息需要在 Mount 之后才能通过内部视图查询，如果磁盘组因为故障无法正常加载，那么信息将不可用，这为诊断带来了诸多不便。

从 Oracle 11g 开始，Oracle 提供了一个工具 AMDU 用于协助诊断。通过这个工具可以在磁盘组加载之前将 ASM 的元数据抽取出来，用于数据库诊断。这个工具可以向后兼容，引入到 Oracle 10g 中。

通过 amdu -h 可以查看详细的帮助说明。直接调用 amdu，将自动生成一个以时间命名的目录，其下的报告文件记录了磁盘组的相关信息。

```
[oracle@enmou1 ~]$ amdu
amdu_2011_03_29_10_28_41/
[oracle@enmou1 ~]$ cd amdu_2011_03_29_10_28_41/
[oracle@enmou1 amdu_2011_03_29_10_28_41]$ ls
report.txt
```

该报告的主要内容如下。

```
[oracle@enmou1 amdu_2011_03_29_10_28_41]$ more report.txt
-*-amdu-*-
```

```
****************************** AMDU Settings ********************************
ORACLE_HOME = /u01/app/db/11.2.0
System name:    Linux
Node name:      enmou1
Release:        2.6.18-128.el5
Version:        #1 SMP Wed Dec 17 11:41:38 EST 2008
Machine:        x86_64
amdu run:       29-MAR-11 10:28:41
Endianess:      1
-------------------------------- Operations --------------------------------
******************************** DISCOVERY **********************************

----------------------------- DISK REPORT N0001 -----------------------------
                Disk Path: ORCL:VOL1
           Unique Disk ID:
               Disk Label: VOL1
     Physical Sector Size: 512 bytes
                Disk Size: 1954 megabytes
               Group Name: CRSDG
                Disk Name: VOL1
       Failure Group Name: VOL1
              Disk Number: 0
            Header Status: 3
       Disk Creation Time: 2011/03/17 11:39:10.772000
          Last Mount Time: 2011/03/29 09:21:38.608000
    Compatibility Version: 0x0b200000(11020000)
         Disk Sector Size: 512 bytes
         Disk size in AUs: 1954 AUs
         Group Redundancy: 1
      Metadata Block Size: 4096 bytes            →元数据块大小，4KB
                  AU Size: 1048576 bytes         →AU 大小：1MB
                   Stride: 113792 AUs
```

```
      Group Creation Time: 2011/03/17 11:39:10.671000
  File 1 Block 1 location: AU 2                  →文件使用，从 AU 2 开始
              OCR Present: NO

**************************** END OF REPORT ******************************
```

定义特定的参数可以获得 ASM 磁盘组内部的区间分配等详细信息。以下命令指定转储 CRSDG 的磁盘组信息，除报告文件外，还生成了 map 和 img 信息文件。

```
[oracle@enmou1 ~]$ amdu -diskstring '/dev/oracleasm/disks/VOL*' -dump 'CRSDG'
amdu_2011_03_29_10_36_03/
[oracle@enmou1 ~]$ cd amdu_2011_03_29_10_36_03/
[oracle@enmou1 amdu_2011_03_29_10_36_03]$ ls
CRSDG_0001.img  CRSDG.map  report.txt
```

这里 map 文件的信息如下，其中描述了 ASM 元数据在磁盘组中的位置，最后部分就是指针信息。

```
[oracle@enmou1 amdu_2011_03_29_10_36_03]$ more CRSDG.map
N0001 D0000 R00 A00000000 F00000000 I0 E00000000 U00 C00256 S0001 B0000000000
N0001 D0000 R00 A00000001 F00000000 I0 E00000000 U00 C00256 S0001 B0001048576
N0001 D0000 R00 A00000002 F00000001 I0 E00000000 U00 C00256 S0001 B0002097152
N0001 D0000 R00 A00000003 F00000002 I0 E00000000 U00 C00256 S0001 B0003145728
N0001 D0000 R00 A00000004 F00000003 I0 E00000000 U00 C00256 S0001 B0004194304
```

而 img 文件则是元数据块的镜像转储，为二进制文件。这些文件在 ASM 出现故障时，可用于收集信息，分析故障。

AMDU 有一个重要参数 extract，该参数可用于从 ASM 磁盘组中抽取数据文件。以下是 AMDU 的帮助信息摘录。

```
extr/act        Files to extract
-extract <diskgroup>.<file_number>: This extracts the numbered file
    from the named diskgroup, case insensitive. This option may be
    specified multiple times to extract multiple files. The extracted
    file is placed in the dump directory under the name
    <diskgroup>_<number>.f  where <diskgroup> is the diskgroup name
    in uppercase, and <number> is the file number. The -output option
    may be used to write the file to any location. The extracted file
```

```
will appear to have the same contents it would have if accessed
through the database. If some portion of the file is unavailable
then that portion of the output file will be filled with
0xBADFDA7A, and a message will appear on stderr.
```

这个选项可用于直接从 ASM 磁盘组中抽取数据文件。

文件分析

由于磁盘组不能 Mount，控制文件也无法访问，因此需要首先分析数据库的文件分布情况，进而通过文件的 ASM 存储序号来进行文件抽取。

通过告警日志，可以找到数据库的控制文件信息。如下所示，控制文件的 ASM 文件号是 270。

```
System parameters with non-default values:
  processes                = 150
  spfile                   = "+DG_DATA/proda02/spfileproda02.ora"
  memory_target            = 4G
  control_files            = "+DG_REDO/proda02/controlfile/current.270.768047731"
  db_block_size            = 8192
  compatible               = "11.2.0.0.0"
  cluster_database         = TRUE
  db_create_file_dest      = "+DG_DATA"
  db_create_online_log_dest_1= "+DG_REDO"
  thread                   = 2
  undo_tablespace          = "UNDOTBS2"
  instance_number          = 2
  remote_login_passwordfile= "EXCLUSIVE"
  db_domain                = ""
  remote_listener          = "rac03-cluster-scan:1521"
  audit_file_dest          = "/u01/oracle/admin/proda02/adump"
  audit_trail              = "DB"
  db_name                  = "proda02"
  open_cursors             = 300
  diagnostic_dest          = "/u01/oracle"
```

随后即可通过 amdu 提取控制文件。

```
su - oracle
mkidr –p /u01/d02
cd /u01/d02
amdu -extract DG_REDO.270
```

取得控制文件之后，可以通过控制文件内容获得数据库的数据文件及日志文件分布情况。以下是从控制文件中获得的信息输出。

```
strings DG_REDO.270 | grep +DG_DATA >02_DATAFILE.TXT
strings DG_REDO.270 | grep +DG_REDO >> 02_DATAFILE.TXT
cat ./02_DATAFILE.TXT
+DG_DATA/proda02/datafile/system.282.769365511
+DG_DATA/proda02/datafile/sysaux.278.769365517
+DG_DATA/proda02/datafile/undotbs1.281.769365521
+DG_DATA/proda02/datafile/undotbs2.279.769365529
+DG_DATA/proda02/datafile/undotbs3.277.769365529
+DG_DATA/proda02/datafile/users.276.769365531

+DG_REDO/proda02/onlinelog/group_1.274.769365509
+DG_REDO/proda02/onlinelog/group_2.273.769365509
+DG_REDO/proda02/onlinelog/group_5.276.769366163
+DG_REDO/proda02/onlinelog/group_6.275.769366163
+DG_REDO/proda02/onlinelog/group_3.272.769366163
+DG_REDO/proda02/onlinelog/group_4.271.769366165
```

有了文件分布信息，接下来的恢复工作就大大简化了。

AMDU 文件恢复

获得文件的分布信息之后，即可使用 amdu 工具进行文件提取工作。根据如上的数据文件和日志文件信息，创建如下脚本以抽取对应的日志文件和数据文件。

```
amdu -extract DG_DATA.282
amdu -extract DG_DATA.278
amdu -extract DG_DATA.281
```

```
amdu -extract DG_DATA.280
amdu -extract DG_DATA.279
amdu -extract DG_DATA.277
amdu -extract DG_DATA.276
amdu -extract DG_DATA.284
amdu -extract DG_DATA.283
amdu -extract DG_REDO.274
amdu -extract DG_REDO.273
amdu -extract DG_REDO.276
amdu -extract DG_REDO.275
amdu -extract DG_REDO.272
amdu -extract DG_REDO.271
```

运行以上脚本，就可以将相应的数据文件和日志文件从磁盘组中提取出来，然后将这些文件整合到一个统一的目录中。

通过配置好的参数文件和抽取的控制文件，可以将数据库启动到 mount 状态。具体过程如下。

```
export ORACLE_SID=d02
startup nomount pfile='/u01/d02/ora.pfile'
alter database mount;
```

接下来编写如下文件重命名脚本，将文件指向统一的存储目录。

```
alter database rename file '+DG_DATA/proda02/datafile/system.265.768047733'
   to '/u01/d02/DG_DATA_265.f';
alter database rename file '+DG_DATA/proda02/datafile/sysaux.266.768047739'
   to '/u01/d02/DG_DATA_266.f';
alter database rename file '+DG_DATA/proda02/datafile/undotbs1.267.768047743'
     to '/u01/d02/DG_DATA_267.f';

alter database rename file '+DG_DATA/proda02/datafile/undotbs2.269.768047751'
  to '/u01/d02/DG_DATA_269.f';
alter database rename file '+DG_DATA/proda02/datafile/undotbs3.270.768047753'
  to '/u01/d02/DG_DATA_270.f';
alter database rename file '+DG_DATA/proda02/datafile/users.271.768047753'
  to '/u01/d02/DG_DATA_271.f';
```

在这个案例中，由于文件和日志完好，数据库随后就可以成功打开。

对于特定的文件，通过以下测试可以验证 amdu 的恢复过程和文件完好性。

```
+DG_DATA/proda02/datafile/users.271.768047753'
```

通过 amdu 提取文件。

```
[oracle@oracle]$ amdu -diskstring '/dev/oracleasm/disks/DISK*' -extract 'DG_DATA.271'
    amdu_2012_02_22_02_02_09/
```

通过更名来重定向数据文件。

```
SQL> ALTER DATABASE RENAME FILE '+DG_DATA/proda02/datafile/users.271.768047753'
     to '/u01/amdu_2012_02_22_02_02_09/DG_DATA_271.f';
Database altered.
SQL> alter database open;
Database altered.
SQL> select name from v$datafile where name like 'DG%';
NAME
--------------------------------------------------------------------------------
/u01/amdu_2012_02_22_02_02_09/DG_DATA_271.f
```

这个案例的幸运之处在于磁盘组未发生更为严重的损坏，数据文件和日志文件都是完好的，而 Oracle 的 AMDU 工具在这种情况下为我们提供了便利的恢复手段。

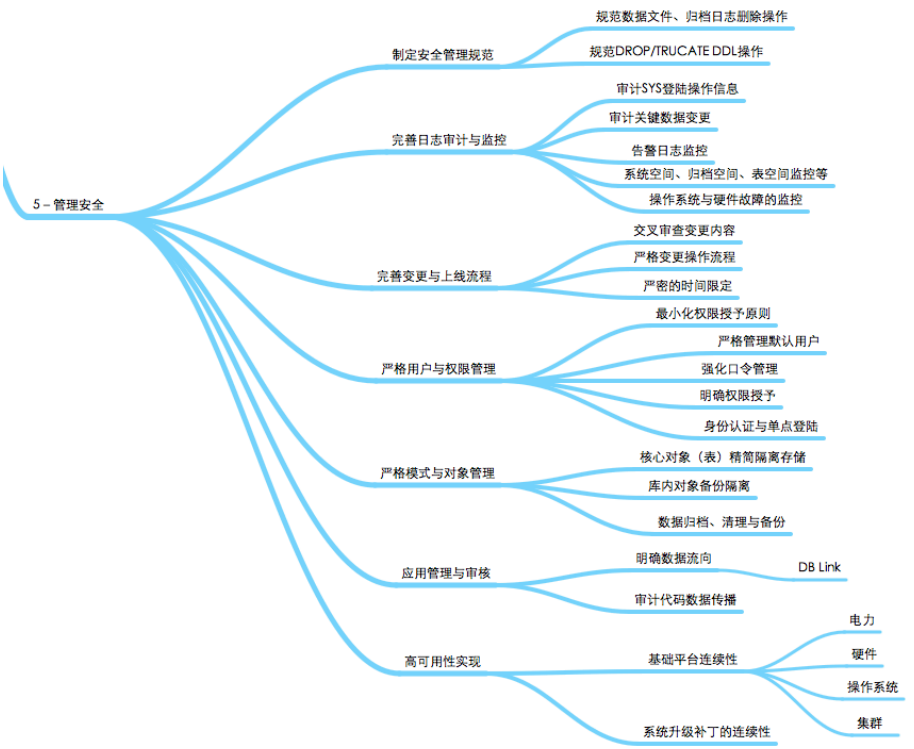
未雨绸缪，防患未然

宜未雨而绸缪，毋临渴而掘井。

——朱用纯《治家格言》

君子以思患而豫防之。

——《周易·既济》



在企业数据环境中，已经有很多 DBA 因为疏忽而遭受了惨痛的教训。通过了解和学习这些案例，可以帮助我们熟悉常见的错误种类和错误条件，进而增强规范、建立制度约束和技术防范，规避和减少技术风险，实现数据环境的安全运行和保障。

在数据管理阶段出现的问题，我们定义为管理安全范畴。在这一阶段，如果企业缺乏足够的管理规范，就可能出现很多安全事故，如误删除、Trucate 数据表等。

篇首图列举了管理安全范畴中可能涉及的诸多方面。

在本篇中，我们将主要描述一下各行业 DBA 遭遇的管理安全问题。从这些现实案例中总结经验，吸取教训。从灾难中学习，进而未雨绸缪，防患于未然，这也是每个企业数据运维人员的职责所在。

DBA 四大守则

通过分析所经历和遭遇到的各种案例，我曾总结出 DBA 生存四大守则，用于提醒 DBA 们在工作中应当注意和遵循的内容，这些守则直至今天仍然具有其现实意义。

以下是我认为极其重要的、需要铭记于心的 DBA 生存守则。

1. **备份重于一切。**

我们必须知道，系统总是要崩溃的，硬盘总是要损坏的；没有有效的备份只是在等哪一天死！我经常开玩笑地说，唯一会使 DBA 在梦中惊醒的就是：**没有有效的备份**。

如果你睡前想一想，“那个没有备份的数据库，如果今晚硬盘损毁，明天如何去恢复数据和业务？”，如果真的没有备份，恐怕没有 DBA 能够安然入睡。

2. **三思而后行。**

Think thrice before you act.

任何时候都要清楚你所做的一切，否则宁可不做！

有时候一个回车、一条命令就会造成不可恢复的灾难，所以，你必需清楚确认自己所做的一切，并在必要时保护现场，保证所做操作至少不会使事情变得更糟。

我们已见过太多草率的操作导致的数据灾难。

3. **rm 是危险的。**

要知道在 UNIX/Linux 下，这个操作意味着你可能将永远失去这个命令后面的东西，所以，请确认你的操作！

已经有太多的人在“rm -rf”上悲痛欲绝。当年写下这条守则时是一个凌晨，一个朋友将我吵醒，说他误操作 rm -rf 删除掉了 200GB 的数据库，并且没有备份。

我当时能告诉他的只有一句话：要保持冷静，离开键盘，然后让你的领导知道这件事。

作为一个技术人员，当事情超出了你的掌握，最好的决定就是让更多的人知道这件事，通过共同的决策来制订恢复方案，防止因为个人的再次判断失误造成进一步的损失。

rm 是危险的，以此类推，在数据库内部执行 DROP/TRUNCATE 等破坏性操作时，同样应当谨慎，**小心一万次也不多，疏忽一次就可能致命**。

4. 你来制订规范。

良好的规范是减少故障的基础。所以，作为一个 DBA，你需要来制订规范，规范开发甚至系统人员，这样甚至可以规避有意或是无意的误操作，减少数据库的风险。

而作为企业数据环境的管理人员，更应该从管理、流程上制订规范，防止因为管理流程不当导致的数据安全、数据灾难出现。

我们知道，在管理良好的数据库服务器上，rm -rf 甚至都可能是不允许使用的。

我们需要遵守的可能更多，所以我一直强调 DBA 一定要严谨专注，在管理数据库的同时，要承担起数据责任，不能有丝毫的马虎和大意，草率的判断和轻忽的选择对数据来说很可能是致命的。当然我也非常喜欢另外一句话：坚韧卓绝之人，必能成就万事。如果有兴趣、有毅力，就一定能够做好数据的守护者。

以上四大守则，愿与诸位 DBA 朋友共勉。

DBA 守则外两则

在《Oracle DBA 手记 2》一书中，我的朋友、作者之一郭岳，曾经提出了几点 DBA 守则，我深以为然，并对其中几点略作引申，收录在这里供大家参考。

1. **高度的信息安全意识。**

本书的宗旨是安全，希望能够对大家的数据库安全起到帮助。在数据安全的诸多方面中，信息安全是核心环节。DBA 或相关技术人员能够接触到核心数据（虽然各种技术手段不断发展衍生出来用于防范 DBA 的数据获取），但是接触也就意味着责任，权利与责任从来都是如影随形的。能够访问获取数据，并不意味着你就可以去利用和传播这些数据，DBA 的职业道德要求你仅仅完成责任范围之内的数据库维护，对于业务数据应当敬而远之，视而不见。

在 2011 年，一则新闻触动了很多数据工作者的神经：陕西 1400 多万的手机用户信息被泄露，不法分子利用这些信息以短信推广等方式非法牟利。对于运营商来说，保有用户信息也意味着责任，如果不能保障这些数据的安全，就可能被不法分子利用，侵犯公民的个人隐私甚至经济利益；而对于数据工作人员，接触数据的同时也要遵守职业道德和法律法规，严格避免不必要的信息泄露和传播。

以下是这则案件的摘要信息。

> 根据 2011 年 09 月 23 日的新闻报道，西安警方破获了全国首例非法出售、获取公民个人信息的系列案件。
>
> 据了解，这一系列案件导致陕西省近 1400 万手机用户的个人信息被泄露。
>
> 根据最后的破获结果，犯罪嫌疑人是一家科技公司的技术人员，他代表公司负责研发和维护陕西省某通信公司的计费经营系统。
>
> 2011 年 3 月以来，他利用工作便利，多次侵入这家通信公司的用户数据库，盗取手机用户个人信息。
>
> 2011 年 4 月，他应另一嫌疑人的要求，再次侵入某通信公司客户数据库，窃取了西安、咸阳、铜川等 7 个地市 1394 万手机用户的个人信息。
>
> 据警方介绍，这 1394 万手机用户的个人信息，占全省手机用户的 60%~70%。这些信息被非法交易的背后，是利益的驱动，使得 4 名犯罪嫌疑人形成承上启下的非法利益链条。警方同时指出，

该案件的侦破，也**暴露出通信运营商和为通信运营商提供技术支持的科技企业，都存在监督职责的缺失**。

律师认为，盗取手机信息的技术人员，**首先违反了民事法律规定，需承担相应民事赔偿责任**。刑法修正案中有涉及泄漏、盗取私人信息的相关规定：**非法泄露个人信息，最高可判三年有期徒刑**。

国家机关或者金融、电信、交通等单位的工作人员，违反国家规定，将本单位在履行职责或者提供服务过程中获得的公民个人信息，出售或者非法提供给他人，情节严重的，处三年以下有期徒刑或者拘役，并处或者单处罚金。窃取或者以其他方法获取上述信息，情节严重的，依照前款的规定处罚。

以下文字节选自《Oracle DBA 手记 2》。

运营商最为看重的事情是什么？如果我问你，你对运营商最不满意的是什么？或许你会说，是网络质量，是资费，是服务，等等。其实这些都不是运营商最为看重的事情。假设你的通话记录会被泄密，我想，即便这家运营商的网络质量好到你在海底 10000 米都能流畅通话，资费低到打长途比市话还便宜，服务好到给你配个一对一服务的美女客户经理，你还是会选择离开这家运营商。

既然客户这么看重信息安全，那么，运营商最为看重的，必然也是信息安全。作为运营商的 DBA，由于工作的需要，能接触到很多核心的信息，例如，运营商最终用户的身份信息，什么姓名、家庭住址，甚至是身份证号码，还有通话的详单等等。人都是有好奇心的，当你开始做运营商的 DBA 时，你会发现，以前怎么都不可能得到的信息，现在，只需举手之劳就能偷窥到。可能仅仅是为了在朋友面前证明下自己多么强大，多么厉害，赢取朋友羡慕的眼神，然后，你就偷窥了这些信息。记住，这样的事情，不仅仅会毁了你自己的 DBA 职业生涯，甚至会毁掉你的公司。

当经历了无数次的信息安全泄密事件的通报，经历了领导的一次一次的安全教育以后，你已经不会再主动地为了赢取朋友羡慕的眼神而去偷窥不该看到的信息，这个时候，最为严重的可能就是无意的泄密了。

关于信息安全泄密的事件，在我工作的运营商那里，有两个典型的案例，一次发生在我去那里干活之前，另外一次，很巧合地发生在我离开那里之后。

先说说第一次吧，在我去那里干活以前，一家厂商的工程师，一时迷糊窃取了用户的一些信息泄密给外面很有名的调查公司。在查出来以后，他受到的惩罚是开除、赔偿，同时，去其他任何一家 IT 公司，都会被原来公司发律师函，说明其为什么被开除。我相信，从此以后，他再也不用从事 IT 行业的工作了。

另外一次，是某技术人员在网上论坛发帖，为了演示某个 SQL 功能，查询显示了部分人员的姓名、手机号码和 E-mail 地址等信息。这引发了一次安全危机，以致动用了大量的人力物力去追查。

追查的结果是一个做系统监控的工作人员，无意中泄露了信息，我不知道最后会如何处理这个人，但是估计不会很轻。

其实这样的泄密，一看就是无意识的。**很多时候，DBA 就是要对无意识的事情，保持高度的警惕**。为了这样的事情，影响到自己的职业生涯，是绝对不划算的。建议 DBA 在发布任何会有第三方知道的文档之前，先问问自己，这份东西会导致泄密么？多问自己几次，多确认几次，如果不能确认，那就和你的主管确认吧。我还记得，以前我们有一条铁的规定，非项目经理及以上级别，不得公开对甲方发布任何技术文档及承诺。这条规定就很好，可以避免很多不必要的麻烦。

2. 忘记你的系统有备份。

虽然我曾经反复强调，备份重于一切，但是我不得不承认，有时候，忘记你的系统有备份的确是一个重要的提示。

在本书部分案例中，客户存在备份，但是由于数据量庞大或者空间有限，使用备份进行恢复的成本很高或者不现实，于是客户不得不选择采取一些较为极端的方式来进行异常修复。

而很多 DBA 之所以犯下错误，也正是因为觉得别人做过备份，已经有了备份。这些想当然的前提在故障出现之后可能不再成立，而这正是很多经典故障的根源。

所以，我非常欣赏郭岳在书中提出的观点：**在有些时候，忘记你的系统有备份**。

很多人，看到这个标题，可能觉得十分诧异，作为一个 DBA，需要遵守的守则中的第一条，就是要备份。

在 eygle 的 DBA 四大守则中的第一条，写的就是备份重于一切，并且很明白地说明，唯一能使 DBA 半夜惊醒的事情，就是系统没有备份。首先不得不说明，我完全同意这个观点，但是在维护电信运营商系统的时候，请你一定要忘记你的系统是有备份的。

我提出这个观点，基于以下两个原因。

首先，对于运营商级别的系统，数据量之大，如果没有到达非常让人崩溃的情况（例如，生产系统崩溃，容灾系统无法启动，并且应急系统也无法运行这样极端），是不会去采用恢复数据的方案的，因为数据量太大，恢复的时间太长，而运营商是要求业务能运行为第一要素的。

其次，人都是有依赖性的，当你知道你的系统有备份的时候，你的操作就开始不那么如履薄冰了，就开始毛躁了，因为你的潜意识会认为，反正我的系统有备份，大不了恢复回来。

忘记你的系统有备份吧，忘记它吧。

可能是我个人的习惯，如果一项特性对于我来说是绝对弊大于利，比如 RAC 的 load balance 这

样的特性，我一般会选择忘记它。记得自己的系统有备份，对平时的工作没有任何的帮助，只会让自己的潜意识操作不那么精细，不那么谨慎，所以，请忘记系统有备份。

但是，请不要忘记，定期检查你的备份是否有效！如果你连定期检查系统的备份是否有效也一起忘记了，那就有点过犹不及，得不偿失了。嗯，忘记你的系统有备份，但是别忘记定期检查系统备份的有效性。

各种惨痛的案例

我们曾经在网络上收集过大家曾经犯过的错误信息，这些信息千奇百怪、五花八门，对于管理者，了解一下这些错误有助于制定规范、防范错误；对于 DBA 则有助于吸取教训，增长经验，防患于未然。

对于这些案例，我多数保留了原始的描述和记录，做了一些简单的编辑整理和分类工作，这些案例收集自网络，与大家共为警醒。

系统级误删除案例

以下一组案例与删除有关，在操作系统上执行 rm 命令应当非常谨慎，重要的生产环境，应当将 RM 命令进行别名重定义。在我的 DBA 生存四大守则里，“rm 是危险的”就是其中非常重要的一条。本书中就有多个案例与 RM 删除数据文件有关。

在这些案例之前，我想引用一下在程序员的世界里，2011 年影响深远的一个 BUG，这个 BUG 被戏称为“一个空格引发的惨剧”：

> bumblebee 是一个开源项目，这个名字也就是变形金刚里的大黄蜂，这个项目是这样介绍自己的。
>
> bumblebee is Optimus support for Linux, with real offloading, and not switchable graphics.. More important.. it works on Optimus Laptops without a graphical multiplexer..
>
> Optimus 是 NVIDIA 的“优驰”技术，其可以将您的笔记本电脑 PC 提升到绝佳状态，提供出色的图形性能，并在需要时延长电池续航时间。这个项目是把这个技术移到 Linux 上来。
>
> 这个项目本来不出名，不过，程序在其安装脚本 install.sh 里的一个 bug 让这个项目一下子成了全世界最瞩目的项目，这个 bug 的 fix 如下：

```
    @@ -348,7 +348,7 @@ case "$DISTRO" in
-  rm –rf /usr /lib/nvidia-current/xorg/xorg
+  rm –rf /usr/lib/nvidia-current/xorg/xorg
```

> （引自 http://coolshell.cn/articles/4875.html）

注意到么，在 rm -rf /usr 后面多了一个空格，这会导致用户的/usr 下所有文件被删除，一个简单的空格造就了一个伟大的 BUG，并且引发了全球数以万计的程序员前往围观留言。

而关于这样一个空格引发的灾难在数据库 DBA 的世界里，也层出不穷。

根据这些灾难和经验总结，我们建议所有重要的数据环境中可以参考以下建议：

1. 通过别名或重定义方式提示或禁用 rm 操作

或者制定一个规范，通过 mv 的方式进行文件转移，通过一定时间段（如一周）观察无误后，再彻底清除数据文件。

rm 操作的危险性必须得到技术人员的充分重视。

2. 加强数据环境的空间监控

很多用户是在空间达到 100%之后才去匆忙进行空间清理，匆忙常常会带来考虑不周，误操作等意外发生。

所以我们建议加强数据环境的存储空间监控，不要等到 100%再去应急，应当总是使空间留有阈量，提前进行空间维护，避免手忙脚乱的应急处理。

3. 在紧急删除之前做好备份

如果不可避免的要进行紧急的文件删除工作，那么在条件允许的情况下，应当做好备份转移到其他主机或存储，避免无法回退恢复的灾难。

通常文件的转移并不会花费太多的时间，在可能情况下用转移替代删除，在必须删除时，也要考虑能否保留最后一个备份。

4. 避免在持续工作或者凌晨仓促的进行文件删除等工作

人在疲劳和不清醒的状态下极易犯下错误，所以应当尽量避免在连续工作的疲惫状态下，或者在梦中惊醒的凌晨迷糊状态下进行维护工作，比如文件删除，在这种状态下，极易出现误判，造成误操作。

另外，在操作之前确认你的当前路径，很多灾难是由于当前路径错误导致的，在 Unix/Linux 下，可以通过 pwd 命令来确认。

5. 重要的操作实现人员备份

前面提到过这点建议，再次重申，在执行重要操作时，最好有两个人同时在场，互相监督审核，避免一个人草率或者考虑不周导致的误操作。

这些系统级别的误删除操作，导致了一次又一次的数据灾难，这些灾难极其相似，也极其惨痛，我们每个人在学习之后都不应该再犯下这样的错误。

这些案例中，也有很多空格在一些 DBA 的职业生涯中刻骨铭心：

案例概述	案例详情
误删除 Oracle 软件	硬件维护人员删除归档日志的时候,把节点 2 的整个 ORACLE_HOME 都删除了。 ...在删除的时候没有注意到目录改变了,还键盘做了一个向上的动作，刚好就是刚刚使用的 rm -rf *，然后一个**下意识的动作回车**就这么按下去了
误删除 Oracle 软件	曾经犯过 update 和 rm 两个错误、哎~~~，update 了一大片数据,rm 了整个$ORACLE_HOME 目录,后者那个错误是在测试库，前者那个错误是在生产库
误删除 Oracle 软件	rm -rf /opt/ora92/* 在测试库中本来想删除数据库，结果错误的把 ORACLE 软件删除了．郁闷啊,幸好不是生产库
误删除所有数据文件	在 linux 平台上，一次不小心操作，把 oradata 下所有的东西全删除了。至今**铭刻于心**
误删除软件及数据	不小心用 rm -rf /home 删除了目录下的所有文件 /home 目录下放的账务系统的 app。一看删除的路径的不对，已经来不及了
误删除所有文件	一次在一个目录下发现有个*开头的文件,就写了个 rm -rf *.后缀，结果可想而知了吧
误删除软件及数据	在生产环境目录下，执行 rm -R *.*，结果数据库、应用全部 over。停机 5 天，叫了 N 个人才搞好

空格导致的误删除	我最难忘的：root 用户 在根目录下 rm -rf abc *，abc 和*之间有个空格，结果把 os 删除了。 已经成为佳话。什么事情都可能发生的。 从此，整个人好像变了一样，**做什么事情，都三思而后行**了
空格导致的误删除	在测试环境下遇到过在删除一些临时文件时，在 * 和 .tmp 中间多加了空格. 后果很严重啊! 想想都后怕,如果是在生产环境下，一身冷汗! rm * .tmp
空格导致的误删除	列举一次绝对毁灭性的操作： 重建表空间后文件系统里有些数据文件没有用了，我打算清理空间本来语句类似是这样写： rm -rf ts_tab_test* 由于网络有延迟，导致了手欠，多敲了一个空格，写成了 rm -rf ts_tab_test * 结果这个目录下所有的数据文件都被我删除了，绝对崩溃
空格导致的误删除	偶的教训不是很深刻，不过意义很重大： 删除一些 trace 文件，然后就直接删除 rm orcl*，结果通过 vpn 到生产的，网络太慢，命令刚刚慢慢的显示出来，看都没看直接按回车，结果执行的命令却是 rm orcl *，因为 orcl 和星号中间有个空格，所以把这个目录下面所有的内容全部删除了。出了一身冷汗，试想，如过是删除数据文件目录下的内容，那立马死翘翘了 到现在为止，每次都要等命令完全显示出来，从

	头到尾看一遍再执行。不过，**大多数错误都是在很繁忙或者很疲劳的情况下发生的**，呵呵，看来 dba 应该多休息才是
空格导致的误删除	删除日志时，输入 rm * .log，因为星号*与 log 之间有空格，所以删除了该目录下所有的文件，吓得我出了一身汗
空格导致的误授权	安装数据库的时候 su - chmod 777 -R /oracle ,多输入一个空格 变成 chmod 777 -R / oracle ，许多系统文件属性变坏，Unix 瘫痪 这个错误犯了两次，用系统恢复磁带重做系统，幸好是测试机。 **从此以后系统部门的同事□不肯给□roo□口令**
误删除日志文件链接	我所經歷的. RAW 磁盤的链接文件, 整理的時候, 下 RM 命令加*, 把正在運行的一個實例的 REDO LOG FLIE 的連接文件刪除了. 當時就感覺出錯誤了. 那個實例當時還沒有 DOWN. 還來切換 REDO LOG 時候找不到文件就 DOWN 了. 幸好 RAC 其他實例正常運行, 用戶和其他部門都沒有感覺到. 還來把那個連接文件重新建立, 又可以啟動了. **自此下 RM 命令很小心**
误删除数据文件	當時，那幾天都是很疲勞的。在開發環境作數據文件分佈調整時，先 cp 完某個表空間所有文件到其他地方，然後作*匹配 rm 了此表空間在此目錄的數據文件。但是 rename 時發現居然有一數據文件沒 cp 過來，忘了說了，此表空間是 system 表空間。沒辦法，開發人員明天還要使用這個環境。幸虧之前有一備份，不

	過當時磁盤空間不是很充裕，足足折騰了一夜才搞定！想起來都後怕哦，幸虧不是正式環境了！再以後，就很少用 cp,rm 了，特別是 rm *..,**一般是此類操作用 mv 來完成。需要 rm 的東西，一般 mv 到一臨時目錄了，再 rm 了**！呵呵，可能都有點謹慎過頭了哦
脚本中误删除文件	自己写了个 rman 备份以及备份成功后 rm 旧 log 的 shell 脚本，log 的目录赋值給变量，结果执行時目录赋值沒成功，該变量指向了另一個目录，结果下面的东东全没了，系统立即报错（把用户的 home 目录删了）。幸好当时头脑还很清醒，也没误删什么重要的数据，很快就搞定了。以后**脚本中要 rm 某个目录的东东再也不敢用变量表示了**，直接 hardcode 进去算了，這样也放心。另外出问题后一定要冷静，定位出问题原因后再动
误删除目录中挂载	一次生产环境 linux 系统,做整个项目目录的移植,cp 一份确认正常执行后直接 rm 原来的目录,没想到.子目录中居然有 mount 到其他 server 的 XX 目录,结果可想而知...linux 啊.....
误删除自动匹配文件	由于 linux 下的 tab 可以自动出来文件名。 一次快速操作，直接 rm -rf 文件时，用 tab 键出来文件名，没有仔细看，直接把文件都删除了，吓了一跳。 后来一 pwd 看，幸好是/root/目录下。不然，要是把整个/给删除了，那就麻烦了
误删除数据文件	刚进现在的公司不久时,做一个数据仓库项目,同事周日加了一天班把数据抽到一个大表空间里,大概 100G,第二天因为临时表空间增长很快,决定重建,这个临时表空间的开头和那个大表空间名字是一样的,只是后面加了一个_temp,当时也是因为事情比较多,认为这是很简单的,结果输入名字就忘了输 入_temp,把大表