Oracle 数据库软件发布序列

对于 Oracle 数据库软件，其软件及补丁共分为以下几种类型。

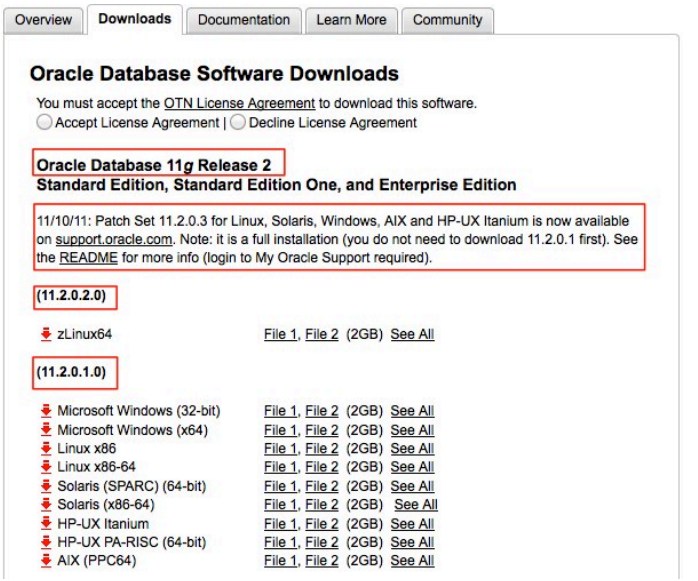
名　　称	说　　明
Release	标准产品发布。如 11g R2 的第一个发布版本为 11.2.0.1，在 OTN 可以下载
PSU（Patch Set Update）	补丁集更新。Oracle 选取在每一季用户下载数量多，并且得到验证具有较低风险的补丁放入到每个季度的 PSU 中
CPU（Critical Patch Update）	紧急补丁更新。Oracle 在 2005 年开始引入的产品安全更新策略，一般来说 CPU 包含了 Oracle 产品安全漏洞的修复补丁集
PSR（Patch Set Release）	包含 PSU 和 CPU，以及其他 Bug 修正的补丁集合
Patch	对于特定 Bug 的单独修复补丁

以上类型中，公开的 Release 版本可以从 OTN 上下载，但是其余修正内容，则只能从 MOS（My Oracle Support）中下载。MOS 只对购买了 Oracle 产品服务的用户开放。

从 Oracle 11g 开始，一个新的补丁策略被引入，11.2.0.1 之后发布的 Patch Set 本身就是一个完整的安装包，不再需要基础的发布版本安装，在 OTN 上可以看到重要的提示信息。比如下页图中 2011 年 11 月 11 日给出的信息是：Patch Set 11.2.0.3 的 Linux/Solaris/Windows/AIX/HP-UX Itanium 版本在 MOS 上已经可用，其为完整的安装版本，不需要预先下载标准发布版本。

对于 Oracle 数据环境的部署，一定要选择当前最为成熟稳定的发布版本，并仔细阅读补丁修正中的修复描述。比如对于 Oracle Database 11g 的部署，目前最为合适的版本就是 11.2.0.3，而对于 Oracle Database 10g 的部署，就应当优先选择 10.2.0.5 版本，这是 Oracle 10g 最后的 Patch Set 发布版本。

在基础的软件发布和补丁集之外，Oracle 对于其产品每个季度发行一次的安全补丁包 CPU 与 PSU 补丁，通常是为了修复产品中的安全隐患，并可能包含对一些严重 Bug 及功能组件的修复。



Oracle 的 CPU/PSU 一般固定在下表所列 4 个月份发布，发布日期为最接近 17 号的星期二。2012 年的几个发布时间分别如下。

月　份	内　容	日　期
January	CPU/PSU	17 January 2012
April	CPU/PSU	17 April 2012
July	CPU/PSU	17 July 2012
October	CPU/PSU	16 October 2012

CPU 是 Oracle 在 2005 年开始引入的产品安全更新策略，是 Oracle 关于安全方面的唯一修正来源。Oracle 致力于通过 CPU 为客户周期性地提供累积性补丁以修复安全漏洞。CPU 中修正的多数是已批露和报告的高危安全漏洞，Oracle 强烈推荐用户应用 CPU 补丁集。

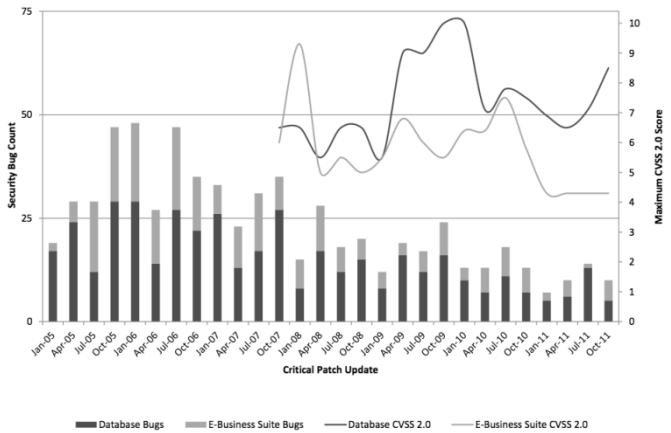
Oracle 在每个季度会将用户下载数量多，并且经过验证具有较低风险的补丁放入到 PSU 中。在 PSU 中，不但包含 Bug 修复，而且包含最新的 CPU。PSU 主要用于解决 PSR 周期长更新不及时的问题，同时用于处理补丁冲突等问题。

下面列举了一些近期 Oracle 发布的高危漏洞说明，其中的很多漏洞和 Oracle 的一些产品组件相关。

- CVE-2011-3525：这个安全漏洞将导致任意 APEX 用户有机会全权控制主机服务器。这个安全漏洞的风险性极高，建议任何安全部署 APEX 的用户立即修正这一安全漏洞。
- CVE-2011-3512：对于所有的 Oracle RDBMS 用户来说，这都是最严重的安全风险。除安装补丁外，没有任何的临时方案。这一漏洞不需要任何特别的 PACKAGE 权限，利用 Spatial 对象进行 SQL 注入，任何用户都可以获得数据库的控制权限。
- CVE-2011-2301：这个风险会导致任何有组件权限的用户获得权限提升，默认的具有 EXECUTE ANY PROCEDURE 权限的用户或者 CTXSYS 用户获得对数据库服务器的完全控制，任何用户都应当立即应用这个修正。
- CVE-2011-3511：这个漏洞会允许任何拥有 DB_ACTMGR（Account Manager users）权限的用户绕过 Database Vault 的保护去修改 VALUT 用户密码，使用 Database Vault 的用户都应该立即修正该 Bug。Oracle Database Vault 在保护用户账户时采取了分离权限的措施，带有 SYSDBA 权限（CVE-2011-2322）或 DV_ACCTMGR 能力（CVE-2011-3511）的用户可绕过这些保护措施，更改用户调用 OCIPasswordChange 客户端 API 的密码。
- CVE-2011-2322：这是另外一个会导致权限提升的漏洞，使用 Database Vault 会使用户获得 SYSDBA 超级权限。这个问题曾在 April 2011 CPU 中得到部分修正。

可以看到，在以上的安装部署中，受到影响的组件有 Spatial、TEXT、Valut 等，这些组件使数据库遭受到了严重的安全风险。所以通常建议用户在确定不需要某些组件时，不要安装相应的数据库模块，以减少数据库的安全风险。

下图列举了 Oracle 安全策略制订以来，定期发布的安全补丁数量。图中，深色部分标示的是数据库的安全补丁数量。



正确评估这些风险，选择性应用相应修正，是数据库管理和服务的一大重要工作内容。

一个逻辑坏块引发的灾难

作为重要的核心数据库系统，应当密切关注 Oracle 的补丁修正和同行业的风险报告，以及社区事件报告。Oracle 官方也会根据需要发布补丁或安全预警，提醒用户及时修正已知的重要缺陷或 Bug。这些内容十分重要，应当预先关注而不是等数据库遇到这些问题时才去了解。

案例警示

这个案例给了我们以下的教训。

1. 对于软件及更新的持续关注不可懈怠。

在这个案例中，数据库因为已知的 Bug 而遭遇严重的事故。这告诉我们，应当对基础软件保持持续的关注，同时参考行业的分析警示公告，定期评估基础软件中可能存在的 Bug 及安全风险，并认真决策软件的修正与维护操作。

很多企业对于稳定运行的系统存在心理麻痹，想当然地认为系统仍然可以继续稳定运行下去。而事实上，很多 Bug 或漏洞，仅在特定的条件下触发，安全的漏洞更需要一段时间去暴露。如果我们掉以轻心，风险也许就会在不久后不期而遇。

2. 对于数据库系统的使用方式应当明确。

客户对于自有的业务系统，应当列表总结明确数据库的访问和使用方式，针对特殊的访问和使用方式，如 DB Link、CTAS、NOLOGGING 等，需要定期跟踪 Oracle 发布的安全预警和 Bug 修正，确认自己的系统不会遭受同样问题的影响。

越是清晰地了解自己的系统，就越可能及早发现问题，规避风险。

3. 储备和维系快速反应的技术专家团队。

很多时候故障发生在一瞬间，及时地判断和响应对于解决故障至关紧要。所以维系、储备专家资源和技术团队对于核心数据库应用必不可少。在关键时刻，通过既定的职责和团队分工、内容分解，可以大幅加快分析和处理速度。

同时保障一个可以用于测试实验的环境也同样重要，很多想法和判断只有在快速测试验证之后，才能够支

持我们去应用到生产环境中。

当一个陌生的故障或异常出现时，解决时间是 2 小时、4 小时还是更久，取决于我们事前做了什么样的准备工作，一个充满活力的学习型团队和组织是企业数据环境的强大支撑。

技术回放

如果一个已知问题没有被注意到，则有可能在核心系统中埋下巨大的安全隐患。作为数据库从业人员，应当知道数据库只有两类，一类是已经崩溃的，另一类是即将崩溃的，舍此无他。

某客户的核心数据库系统，曾经因为一个已知的 Bug 反复遭遇数据灾难，这个著名的 Bug 就是

`Bug 5386204  Block corruption / OERI[kddummy_blkchk] after direct load of ASSM segment`。

这个 Bug 出现在对于 ASSM 存储表的直接路径加载后，其影响范围极其广泛。在以下 Bug 描述列出的影响范围中，可以看到 10.2.0.3 和 10.2.0.4 都在其列。这两个版本应用极其广泛，而很多用户都在数据环境中使用 Direct Load 方式加载数据，可以想象有多少风险存在。很多用户在 10.2.0.5 版本发布之后，迟迟不愿应用这个重要补丁集，但是至少应该关注在这个补丁集中修正了哪些可能遇到的 Bug。

Product (*Component*)	Oracle Server (Rdbms)
Range of versions *believed* to be affected	Versions < 11
Versions *confirmed* as being affected	• 9.2.0.8 • 10.2.0.1 • 10.2.0.2 • 10.2.0.3 • 10.2.0.4
Platforms affected	Generic (all / most platforms affected)

更惨痛的是，这个 Bug 在 10.2.0.4 版本中曾经声明已修正，但事实上直到 10.2.0.5 版本才彻底排除。

Description

Block corruption / ORA-600 [kddummy_blkchk][file#] [block#] [18038]
can occur on a segment which has been direct loaded.

(The corruption shows as a PAGETABLE SEGMENT HEADER
having blocks in the "Auxillary Map" outside of the "Extent Map"
range)

Note:
This bug was previously incorrectly listed as fixed in 10.2.0.4

这个 Bug 在某些金融客户系统中曾多次遭遇，损失惨重。

一个硬盘坏块引发的灾难

以下这次数据库故障非常曲折，但究其根本原因，竟然仅仅是一个硬盘坏块。

灾难描述

以下是一次惨痛的多层次数据库灾难。

1. 客户为了增强系统安全性，对系统根卷进行镜像。
2. 镜像后出现错误，系统重启后需要恢复，于是执行了操作系统级别的恢复。
3. 恢复使得某磁盘组重新生成了 PVID。
4. 该卷原为 ASM 磁盘的裸分区。
5. ASM 磁盘头覆盖损坏，ASM 磁盘无法加载。
6. 灾难形成。

这次复杂和曲折的数据库灾难，客户原本为了安全而采取的措施，最终却引发了一次故障。即便在本书中，这也已经不是第一次出现。这个案例最终找出的原因是镜像盘存在一个坏块，这个坏块引发的修复及后续操作破坏了 ASM 磁盘。

案例警示

通过这个案例的详细发生过程，我们获得了以下启示。

1. **重要维护应当详细检测系统各方面的安全与稳定性。**

在这个案例中，最后发现问题的根源在于，加入根卷镜像的磁盘存在坏块错误，镜像后，原有的根卷盘也出现了同步坏块错误，使得系统提示需要进行恢复。

由于 ODM 存储于根盘，ODM 在恢复时进行了一系列的操作，这其中包括了对于磁盘卷组的 PVID 重算，并最终导致了故障。这场无妄之灾仅仅是因为一个坏块。

所以我们建议大家，在进行重要的维护操作时，应当详细检查相关资源的运行状况，确保系统当前状态的

稳定稳固，确保新增加的设备、配件的健康安全，确保不要将问题引入到当前的系统中来。

2. 复杂的维护需要选择停机窗口长的时段进行。

即便有非常乐观的估计，在进行复杂维护时，也要尽量争取充足的停机窗口。因为隐藏的故障可能大大增加维护的复杂度，一个意外就可能导致几倍的时间需求，所以对维护进行充足的时间考量，是我们重要的维护计划内容。

这个案例中客户选择了在节假日进行，使得为故障处理赢得了充足的时间窗口，未对业务产生不利影响。

3. 做好最坏的打算才能有最好的结局。

在进行数据运维或者系统维护、变更时，一定要做最坏的打算，这样才能有最好的结局。通常很多工程师都会下意识地假设，进行操作系统级别维护或者网络维护跟数据库无关，非常简单，等等。但是注意，这种种因素都和数据库有关，任何一个环节出现问题都可能导致数据库故障。

所以，在进行数据库系统运维时，最好先假定各个环节都可能出现问题，然后进行逐层分析，做好可能性防范，这样才能最大程度地降低维护的风险。在当今高度耦合的计算环境中，系统的相关性使得任何一个环节都不是孤立存在的，必须充分认识这种复杂性从而有效地保障安全。

4. 项目计划要做好充分的资源准备。

在进行维护计划制订时，要做好充足的资源准备，以便在需要时可以快速获得技术支援，做出正确判断。充足的资源准备对于应对故障、快速解决疑难具有决定性作用。

良好的计划、充足的准备是信心的保证，必须具备十足的把握和信心，才能够在维护变更和应对异常时游刃有余，应付自如。

5. 数据库的重要补丁和版本升级非常重要

ASM 磁盘头损坏的情况，一直以来困扰了很多企业和 DBA。在缺少备份情况下，修复 ASM 磁盘头可能出现的种种损坏是相当复杂的。但是从 Oracle 10g 10.2.0.5.0 版本开始，一个自动备份块的引入，使得这种情况下的恢复变得极其简单。

所以，及时准确地获得正确的软件版本，及时应用重要的补丁修正，对于数据库稳定运行和安全具有重大影响。我们希望用户都能够随时关注软件版本情况，为数据库运行构建稳定的基础支持。万丈高楼平地起，关于基础维护，不能出现任何疏忽。

这就是这个案例带给我们的经验与教训。

技术回放

AIX 系统 ODM 简介

对于这个案例，我们需要首先了解一下 ODM 的概念。

在 AIX 操作系统上，内置了一个用于管理的小型关系型数据库，叫做 ODM 数据库（Object Data Manager）。AIX 的许多特性与功能的配置信息都存储在 ODM 库中，如 SMIT、系统配置信息、动态配置信息、磁盘配置信息等。

在 ODM 的设备配置（Device Configuration）区中包含了所有配置的物理卷、卷组和逻辑卷的相关信息。这些信息是对 VGDA（卷组描述区）中信息的镜像。常规的操作，当我们导入（import）一个 VGDA 的过程包括把导入卷组的 VGDA 数据拷贝到 ODM 中。当一个卷组被导出（export）时，保存在 ODM 中的有关该卷组的数据被从 ODM 数据库中删除。

ODM 数据库信息存放于以下三个目录中。

目录结构	记录内容
/etc/objrepos	主要存储不能网络共享的数据，如用户设备定义类等
/usr/lib/objrepos	主要存储只能由 AIX 系统只读共享的数据
/usr/share/lib/objrepos	主要存储可共享，不依赖于 AIX 的数据

在这个案例中，由于 AIX ODM 的损坏与恢复，导致磁盘组的 PVID 重新计算并写入了 ASM 磁盘组，最终引发了故障。

如果没有 ASM 磁盘头信息的备份，手工去修复 ASM 磁盘头是非常困难的。客户的系统存在备份，因而通过备份快速恢复了数据，但是我们仍然要探讨一下 ASM 磁盘头的问题。

由于 ASM 使用裸盘进行数据存储，而操作系统也经常会尝试占用第一个数据块，因而系统工程师的一个处置不当，就可能导致严重的数据库故障。

ASM 头块备份机制

在 Oracle Database Patch Set 10.2.0.5.0 发布之后，我们强烈建议用户升级到该版本上来。因为在这一版本上（头块备份功能最初在 Oracle 11g R1 中引入，随后被引入到 10.2.0.5 版本中），Oracle 会对 ASM 磁盘头自动完成一个镜像备份，这个镜像备份使得常规的磁盘头块损坏的恢复变得易如反掌。

以下是一个常规的 ASM 头块的主要信息，通过 kfed 可以读取，其 Oracle 数据库版本为 10.2.0.5。

```
p55a@/u01/admin/cnd> kfed read /dev/rhdisk3
kfbh.endian:                          0 ; 0x000: 0x00
kfbh.hard:                          130 ; 0x001: 0x82
kfbh.type:                            1 ; 0x002: KFBTYP_DISKHEAD
kfbh.datfmt:                          1 ; 0x003: 0x01
kfbh.block.blk:                       0 ; 0x004: T=0 NUMB=0x0
kfbh.block.obj:              2147483648 ; 0x008: TYPE=0x8 NUMB=0x0
kfbh.check:                  3015734793 ; 0x00c: 0xb3c07609
kfbh.fcn.base:                       44 ; 0x010: 0x0000002c
kfbh.fcn.wrap:                        0 ; 0x014: 0x00000000
kfbh.spare1:                          0 ; 0x018: 0x00000000
kfbh.spare2:                          0 ; 0x01c: 0x00000000
kfdhdb.driver.provstr:         ORCLDISK ; 0x000: length=8
kfdhdb.driver.reserved[0]:            0 ; 0x008: 0x00000000
kfdhdb.driver.reserved[1]:            0 ; 0x00c: 0x00000000
kfdhdb.driver.reserved[2]:            0 ; 0x010: 0x00000000
kfdhdb.driver.reserved[3]:            0 ; 0x014: 0x00000000
kfdhdb.driver.reserved[4]:            0 ; 0x018: 0x00000000
kfdhdb.driver.reserved[5]:            0 ; 0x01c: 0x00000000
kfdhdb.compat:                168820736 ; 0x020: 0x0a100000
kfdhdb.dsknum:                        0 ; 0x024: 0x0000
kfdhdb.grptyp:                        1 ; 0x026: KFDGTP_EXTERNAL
kfdhdb.hdrsts:                        3 ; 0x027: KFDHDR_MEMBER
kfdhdb.dskname:              DATA_0000 ; 0x028: length=9
kfdhdb.grpname:                   DATA ; 0x048: length=4
kfdhdb.fgname:               DATA_0000 ; 0x068: length=9
```

```
kfdhdb.capname:                            ; 0x088: length=0
kfdhdb.crestmp.hi:              32890869 ; 0x0a8: HOUR=0x15 DAYS=0x1f MNTH=0x7
YEAR=0x7d7
kfdhdb.crestmp.lo:             256121856 ; 0x0ac: USEC=0x0 MSEC=0x107 SECS=0x34
MINS=0x3
kfdhdb.mntstmp.hi:              32956086 ; 0x0b0: HOUR=0x16 DAYS=0x15 MNTH=0x7
YEAR=0x7db
kfdhdb.mntstmp.lo:            2504303616 ; 0x0b4: USEC=0x0 MSEC=0x129 SECS=0x14
MINS=0x25
kfdhdb.secsize:                     512 ; 0x0b8: 0x0200
kfdhdb.blksize:                    4096 ; 0x0ba: 0x1000
kfdhdb.ausize:                  1048576 ; 0x0bc: 0x00100000
kfdhdb.mfact:                    113792 ; 0x0c0: 0x0001bc80
kfdhdb.dsksize:                   51200 ; 0x0c4: 0x0000c800
kfdhdb.pmcnt:                         2 ; 0x0c8: 0x00000002
kfdhdb.fstlocn:                       1 ; 0x0cc: 0x00000001
kfdhdb.altlocn:                       2 ; 0x0d0: 0x00000002
kfdhdb.f1b1locn:                      2 ; 0x0d4: 0x00000002
kfdhdb.redomirrors[0]:                0 ; 0x0d8: 0x0000
kfdhdb.redomirrors[1]:            65535 ; 0x0da: 0xffff
kfdhdb.redomirrors[2]:            65535 ; 0x0dc: 0xffff
kfdhdb.redomirrors[3]:            65535 ; 0x0de: 0xffff
kfdhdb.dbcompat:              168820736 ; 0x0e0: 0x0a100000
kfdhdb.grpstmp.hi:              32890869 ; 0x0e4: HOUR=0x15 DAYS=0x1f MNTH=0x7
YEAR=0x7d7
kfdhdb.grpstmp.lo:            255995904 ; 0x0e8: USEC=0x0 MSEC=0x8c SECS=0x34
MINS=0x3
```

在随后的第 510 块上（具体的块号根据版本不同会有所不同），存储着一个备份块，这个块的内容和头块完全相同。在遇到故障时，可以通过 kfed 读取和恢复这个数据块，因而使得传统的 ASM 头块恢复变得非常简单。这是 Oracle 在多个版本的总结之后做出的一个重要改进，也是 DBA 们期待已久的一个重要功能。

```
p55a@/u01/admin/cnd> kfed read /dev/rhdisk3 blkn=510
kfbh.endian:                          0 ; 0x000: 0x00
kfbh.hard:                          130 ; 0x001: 0x82
```

```
kfbh.type:                           1 ; 0x002: KFBTYP_DISKHEAD
kfbh.datfmt:                         1 ; 0x003: 0x01
kfbh.block.blk:                      0 ; 0x004: T=0 NUMB=0x0
kfbh.block.obj:             2147483648 ; 0x008: TYPE=0x8 NUMB=0x0
kfbh.check:                 3015734793 ; 0x00c: 0xb3c07609
kfbh.fcn.base:                      44 ; 0x010: 0x0000002c
kfbh.fcn.wrap:                       0 ; 0x014: 0x00000000
kfbh.spare1:                         0 ; 0x018: 0x00000000
kfbh.spare2:                         0 ; 0x01c: 0x00000000
kfdhdb.driver.provstr:         ORCLDISK ; 0x000: length=8
kfdhdb.driver.reserved[0]:           0 ; 0x008: 0x00000000
kfdhdb.driver.reserved[1]:           0 ; 0x00c: 0x00000000
kfdhdb.driver.reserved[2]:           0 ; 0x010: 0x00000000
kfdhdb.driver.reserved[3]:           0 ; 0x014: 0x00000000
kfdhdb.driver.reserved[4]:           0 ; 0x018: 0x00000000
kfdhdb.driver.reserved[5]:           0 ; 0x01c: 0x00000000
kfdhdb.compat:               168820736 ; 0x020: 0x0a100000
kfdhdb.dsknum:                       0 ; 0x024: 0x0000
kfdhdb.grptyp:                       1 ; 0x026: KFDGTP_EXTERNAL
kfdhdb.hdrsts:                       3 ; 0x027: KFDHDR_MEMBER
kfdhdb.dskname:              DATA_0000 ; 0x028: length=9
kfdhdb.grpname:                   DATA ; 0x048: length=4
kfdhdb.fgname:               DATA_0000 ; 0x068: length=9
kfdhdb.capname:                        ; 0x088: length=0
kfdhdb.crestmp.hi:            32890869 ; 0x0a8: HOUR=0x15 DAYS=0x1f MNTH=0x7
YEAR=0x7d7
kfdhdb.crestmp.lo:           256121856 ; 0x0ac: USEC=0x0 MSEC=0x107 SECS=0x34
MINS=0x3
kfdhdb.mntstmp.hi:            32956086 ; 0x0b0: HOUR=0x16 DAYS=0x15 MNTH=0x7
YEAR=0x7db
kfdhdb.mntstmp.lo:          2504303616 ; 0x0b4: USEC=0x0 MSEC=0x129 SECS=0x14
MINS=0x25
kfdhdb.secsize:                    512 ; 0x0b8: 0x0200
```

```
kfdhdb.blksize:                    4096 ; 0x0ba: 0x1000
kfdhdb.ausize:                  1048576 ; 0x0bc: 0x00100000
kfdhdb.mfact:                    113792 ; 0x0c0: 0x0001bc80
kfdhdb.dsksize:                   51200 ; 0x0c4: 0x0000c800
kfdhdb.pmcnt:                         2 ; 0x0c8: 0x00000002
kfdhdb.fstlocn:                       1 ; 0x0cc: 0x00000001
kfdhdb.altlocn:                       2 ; 0x0d0: 0x00000002
kfdhdb.f1b1locn:                      2 ; 0x0d4: 0x00000002
kfdhdb.redomirrors[0]:                0 ; 0x0d8: 0x0000
kfdhdb.redomirrors[1]:            65535 ; 0x0da: 0xffff
kfdhdb.redomirrors[2]:            65535 ; 0x0dc: 0xffff
kfdhdb.redomirrors[3]:            65535 ; 0x0de: 0xffff
kfdhdb.dbcompat:              168820736 ; 0x0e0: 0x0a100000
kfdhdb.grpstmp.hi:             32890869 ; 0x0e4: HOUR=0x15 DAYS=0x1f MNTH=0x7
YEAR=0x7d7
kfdhdb.grpstmp.lo:            255995904 ; 0x0e8: USEC=0x0 MSEC=0x8c SECS=0x34
MINS=0x3
```

kfed 工具编译与使用

修复 ASM 故障需要用到 Oracle 的 kfed 工具，该工具默认未编译。对于 Oracle 10g R2 版本，可以通过如下步骤获得 kfed 工具。

```
cd $ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk ikfed
```

以下是可能的编译输出（引自 Oracle 10g R2+Linux 环境）。

```
[root@danaly ~]# su - oracle
[oracle@danaly ~]$ cd $ORACLE_HOME/rdbms/lib
[oracle@danaly lib]$ make -f ins_rdbms.mk ikfed
Linking KFED utility (kfed)
rm -f /opt/oracle/product/10.2.0/rdbms/lib/kfed
gcc -o /opt/oracle/product/10.2.0/rdbms/lib/kfed ...
mv -f /opt/oracle/product/10.2.0/bin/kfed /opt/oracle/product/10.2.0/bin/kfedO
```

```
mv: cannot stat `/opt/oracle/product/10.2.0/bin/kfed': No such file or directory
make: [ikfed] Error 1 (ignored)
mv /opt/oracle/product/10.2.0/rdbms/lib/kfed /opt/oracle/product/10.2.0/bin/kfed
chmod 751 /opt/oracle/product/10.2.0/bin/kfed
```

[oracle@danaly lib]$ **which kfed**

```
~/product/10.2.0/bin/kfed
```

在 Oracle 10g 中，kfed 工具的常用参数如下。

[oracle@bossdb-rac2 lib]$ kfed -help

```
as/mlib         ASM Library [asmlib='lib']
aun/um          AU number to examine or update [AUNUM=number]
aus/z           Allocation Unit size in bytes [AUSZ=number]
blkn/um         Block number to examine or update [BLKNUM=number]
blks/z          Metadata block size in bytes [BLKSZ=number]
ch/ksum         Update checksum before each write [CHKSUM=YES/NO]
cn/t            Count of AUs to process [CNT=number]
d/ev            ASM device to examine or update [DEV=string]
o/p             KFED operation type [OP=READ/WRITE/MERGE/NEW/FORM/FIND/STRUCT]
p/rovnm         Name for provisioning purposes [PROVNM=string]
s/eek           AU number to seek to [SEEK=number]
te/xt           File name for translated block text [TEXT=string]
ty/pe           ASM metadata block type number [TYPE=number]
```

通过以上帮助说明可以看到，kfed 支持对于 ASM 信息的 READ/WRITE/MERGE/NEW/FORM/FIND/STRUCT 等操作，大体上非常完备了，只是还缺少一个重要选项。这个选项在 Oracle 11g 中加入了进来，它就是 REPAIR（修复），以下是引自 Oracle Database 11g 的 kfed 帮助信息输出。

[ora11g@hpserver2 ~]$ kfed

```
as/mlib         ASM Library [asmlib='lib']
aun/um          AU number to examine or update [AUNUM=number]
aus/z           Allocation Unit size in bytes [AUSZ=number]
blkn/um         Block number to examine or update [BLKNUM=number]
blks/z          Metadata block size in bytes [BLKSZ=number]
ch/ksum         Update checksum before each write [CHKSUM=YES/NO]
cn/t            Count of AUs to process [CNT=number]
```

```
de/v            ASM device to examine or update [DEV=string]
dm/pall         Don't suppress repeated lines when dumping corrupt blocks
[DMPALL=YES/NO]
o/p             KFED operation type [OP=READ/WRITE/MERGE/REPAIR/NEW/FORM/FIND/STRUCT]
p/rovnm         Name for provisioning purposes [PROVNM=string]
s/eek           AU number to seek to [SEEK=number]
te/xt           File name for translated block text [TEXT=string]
ty/pe           ASM metadata block type number [TYPE=number]
```

这是一个期待已久的功能，有了备份块和 kfed 的修复功能，在磁盘头损坏之后，只需一个 repair 命令就可以完成恢复。

如下这条命令就能够恢复一个损坏的 ASM 磁盘头。

```
kfed repair /dev/rhdisk3
```

手工修复 ASM 案例一则

如果没有 ASM 自动的头块备份,也没有用户的主动备份,那么恢复 ASM 头块损坏将是非常复杂和困难的。以下就是这样一则恢复案例。

灾难描述

以下是这次数据库故障的大致情形。

1. 客户 RAC 系统的一个节点出现故障。
2. 重新安装后，加入 Cluster 成功。
3. 添加 ASM 实例出现错误，无法正确识别磁盘组。
4. 故障形成。

在这个案例中，用户的数据库版本为 10.2.0.4，无法利用到 Oracle 自动备份数据块的特性，也就无法直接恢复。由于没有头块备份，因此只能通过手工方式进行恢复。

技术回放

在类似的案例中，一般来说，重装过程中最复杂的是 Cluster 的清除和添加。如果 Cluster 已经添加成功，那么剩下的就是数据库级别的操作，相对来说会简单得多。根据上面的信息，初步判断问题可能发生在以下几个方面。

- 新节点的 ORACLE_HOME 的版本和旧节点不一致。
- 10204 补丁没有在 Cluster 上安装。
- 在新节点上没有给 Oracle 用户授权。
- 共享存储在新节点上挂载存在问题。

PROVISIONED 磁盘状态分析

到了客户现场后才发现，问题和我们的初始判断大相径庭，数据库的版本、权限等都不存在问题。事实上，新节点上的 ASM 实例已经启动，且两个磁盘组中的一个已经 MOUNT，但另外一个磁盘组无法 MOUNT。

```
SQL> alter diskgroup datag1 mount;
alter diskgroup datag1 mount
*
ERROR at line 1:
ORA-15032: not all alterations performed
ORA-15063: ASM discovered an insufficient number of disks for diskgroup "DATAG1"
```

ORA-15032 只是一个描述性错误，而关键的 ORA-15063 错误比较少见。

由于另外一个节点从 RAC 环境出现故障后一直没有停机，因此可以从这个节点的 ASM 实例上检查磁盘组和磁盘的状态。这又一次给了我们一个提醒，在故障时不要贸然关闭或者重启数据库，否则可能使问题变得更加复杂。

```
SQL> select group_number, name, state, total_mb, free_mb from v$asm_diskgroup;
GROUP_NUMBER NAME                           STATE           TOTAL_MB   FREE_MB
------------ ---------------------------------------- ---------- ----------
1            DATAG1                         MOUNTED         512000     217396
2            DATAG2                         MOUNTED         1024000    347457
SQL> select group_number, disk_number, mount_status, header_status, name, path
  2  from v$asm_disk where group_number = 1;
GROUP_NUMBER DISK_NUMBER MOUNT_STATUS HEADER_STATUS  NAME         PATH
------------ ----------- ------------ -------------- ----------- -------------
1            0           CACHED       PROVISIONED    DATAG1_0000 /dev/raw/raw4
1            1           CACHED       MEMBER         DATAG1_0001 /dev/raw/raw5
SQL> select instance_number, instance_name from v$instance;
INSTANCE_NUMBER INSTANCE_NAME
--------------- --------------------------------
2               +ASM2
```

可以看到，磁盘组 DATAG1 中的磁盘 DATAG1_0000 的文件头状态不正确，其状态为 PROVISIONED。

从新添加的实例 1 上检查 ASM 磁盘信息。

```
SQL> select group_number, name, state, total_mb, free_mb from v$asm_diskgroup;
GROUP_NUMBER NAME          STATE          TOTAL_MB    FREE_MB
------------ ------------ ------------- ---------- ----------
1            DATAG1        DISMOUNTED    0          0
2            DATAG2        MOUNTED       1024000    347457
SQL> select group_number, disk_number, mount_status, header_status, name, path
  2  from v$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       PROVISIONED                /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/raw6
SQL> select instance_number, instance_name from v$instance;
INSTANCE_NUMBER INSTANCE_NAME
--------------- --------------------------------
1               +ASM1
```

由于节点 1 上的 ASM 磁盘组 1 无法 MOUNT，因此可以看到/dev/raw/raw4 和/dev/raw/raw5 两个磁盘对应的 GROUP_NUMBER 是 0，且 MOUNT_STATUS 是 CLOSED，而 raw4 对应的 HEADER_STATUS 也是 PROVISIONED，显然就是这个错误的状态值导致当前节点无法 MOUNT 磁盘组。

由于另外一个实例一直没有关闭过，因此虽然磁盘头的状态不正常，但是并不影响这个磁盘组的正常使用，而如果这时对 ASM 实例进行重启，则这个磁盘组必然无法再次 MOUNT。

根据 Oracle 文档的描述，PROVISIONED 和 CANDIDATE 状态有所区别。二者虽然都表示磁盘为候选磁盘，不属于任何一个磁盘组，不过 CANDIDATE 是正常的候选状态，而 PROVISIONED 则表示磁盘经过特殊的处理，比如操作系统工具对磁盘进行过特殊的操作。

既然这个磁盘的状态是 PROVISIONED，那么能否通过再次添加这个磁盘到磁盘组，使得磁盘头的状态变成正常？于是我们尝试在问题节点再次添加这个磁盘，但是由于磁盘组处于 NOMOUNT 状态，因而添加操作

失败，而尝试在目前正常工作的节点添加磁盘，结果同样报错。

```
SQL> alter diskgroup datag1 add disk '/dev/raw/raw4';
alter diskgroup datag1 add disk '/dev/raw/raw4'
*
ERROR at line 1:
ORA-15032: not all alterations performed
ORA-15029: disk '/dev/raw/raw4' is already mounted by this instance
```

看来常规手段已经无法解决这个问题，于是只剩两个办法：一是重建 ASM 磁盘，二是直接修改 ASM 磁盘头信息。

重建意味着较长的停机时间，而且客户当前环境中配置了 DATA GUARD，重建 ASM 磁盘组也不是一件轻松的事情。当前的 PRIMARY 数据库和 STANDBY 数据库都是两节点的 RAC 环境，在数据库中还配置了 STREAM 的 CAPTURE，这使得数据库架构异常复杂，进行 SWITCHOVER 的难度也相对比较大。而且对于当前运行的节点而言，一旦关闭，很有可能导致 ASM 磁盘无法 MOUNT，从而导致原本计划中的 SWITCHOVER 变成 FAILOVER，甚至有可能损失 ONLINE REDO LOGFILE 的风险。因此，无论从哪个角度看，重建都不是一个最佳的解决方案，如果能直接将 ASM 磁盘的头信息改写正确，无疑是最直接、代价最小的解决方案。

使用 kfed 修改 ASM 磁盘头信息

修改 ASM 磁盘头信息,需要借助 Oracle 的工具 kfed。以下用 kfed 来检查 ASM 磁盘裸设备的文件头信息。

```
[oracle@node1 data]$ kfed read /dev/raw/raw4 > raw4.txt
[oracle@node1 data]$ kfed read /dev/raw/raw5 > raw5.txt
[oracle@node1 data]$ kfed read /dev/raw/raw6 > raw6.txt
```

将磁盘组 DATAG1 的两个磁盘头和另一个磁盘组 DATAG2 的第一个文件的头信息输出到文本文件，对这三个文件进行比对，找到 raw4 文件中异常的部分。

由于每个磁盘组中有两个文件，且有两个磁盘组，因而比对的工作非常顺利，定位到 raw4 中存在两个标志位异常。以下 kfed 输出标示出了两个特殊位置。

```
[oracle@nccpxdb1 lib]$ kfed read /dev/raw/raw4
kfbh.endian: 1 ; 0x000: 0x01
kfbh.hard: 130 ; 0x001: 0x82
kfbh.type: 1 ; 0x002: KFBTYP_DISKHEAD
kfbh.datfmt: 1 ; 0x003: 0x01
```

```
kfbh.block.blk: 0 ; 0x004: T=0 NUMB=0x0
kfbh.block.obj: 2147483648 ; 0x008: TYPE=0x8 NUMB=0x0
kfbh.check: 4024565597 ; 0x00c: 0xefe1ff5d
kfbh.fcn.base: 952047 ; 0x010: 0x000e86ef
kfbh.fcn.wrap: 0 ; 0x014: 0x00000000
kfbh.spare1: 0 ; 0x018: 0x00000000
kfbh.spare2: 0 ; 0x01c: 0x00000000
kfdhdb.driver.provstr: ORCLDISK ; 0x000: length=8
kfdhdb.driver.reserved[0]: 0 ; 0x008: 0x00000000
kfdhdb.driver.reserved[1]: 0 ; 0x00c: 0x00000000
kfdhdb.driver.reserved[2]: 0 ; 0x010: 0x00000000
kfdhdb.driver.reserved[3]: 0 ; 0x014: 0x00000000
kfdhdb.driver.reserved[4]: 0 ; 0x018: 0x00000000
kfdhdb.driver.reserved[5]: 0 ; 0x01c: 0x00000000
kfdhdb.compat: 168820736 ; 0x020: 0x0a100000
kfdhdb.dsknum: 0 ; 0x024: 0x0000
kfdhdb.grptyp: 1 ; 0x026: KFDGTP_EXTERNAL
kfdhdb.hdrsts: 3 ; 0x027: KFDHDR_MEMBER
kfdhdb.dskname: DATAG1_0000 ; 0x028: length=11
kfdhdb.grpname: DATAG1 ; 0x048: length=6
kfdhdb.fgname: DATAG1_0000 ; 0x068: length=11
kfdhdb.capname: ; 0x088: length=0
kfdhdb.crestmp.hi: 32928501 ; 0x0a8: HOUR=0x15 DAYS=0x17 MNTH=0xc YEAR=0x7d9
kfdhdb.crestmp.lo: 2195144704 ; 0x0ac: USEC=0x0 MSEC=0x1d0 SECS=0x2d MINS=0x20
kfdhdb.mntstmp.hi: 32940275 ; 0x0b0: HOUR=0x13 DAYS=0x7 MNTH=0x8 YEAR=0x7da
kfdhdb.mntstmp.lo: 3201116160 ; 0x0b4: USEC=0x0 MSEC=0x34a SECS=0x2c MINS=0x2f
kfdhdb.secsize: 512 ; 0x0b8: 0x0200
kfdhdb.blksize: 4096 ; 0x0ba: 0x1000
kfdhdb.ausize: 1048576 ; 0x0bc: 0x00100000
kfdhdb.mfact: 113792 ; 0x0c0: 0x0001bc80
kfdhdb.dsksize: 512000 ; 0x0c4: 0x0007d000
kfdhdb.pmcnt: 6 ; 0x0c8: 0x00000006
kfdhdb.fstlocn: 1 ; 0x0cc: 0x00000001
```

```
kfdhdb.altlocn: 2 ; 0x0d0: 0x00000002
kfdhdb.f1b1locn: 2 ; 0x0d4: 0x00000002
kfdhdb.redomirrors[0]: 0 ; 0x0d8: 0x0000
kfdhdb.redomirrors[1]: 65535 ; 0x0da: 0xffff
kfdhdb.redomirrors[2]: 65535 ; 0x0dc: 0xffff
kfdhdb.redomirrors[3]: 65535 ; 0x0de: 0xffff
kfdhdb.dbcompat: 168820736 ; 0x0e0: 0x0a100000
kfdhdb.grpstmp.hi: 32928501 ; 0x0e4: HOUR=0x15 DAYS=0x17 MNTH=0xc YEAR=0x7d9
kfdhdb.grpstmp.lo: 2195053568 ; 0x0e8: USEC=0x0 MSEC=0x177 SECS=0x2d MINS=0x20
kfdhdb.ub4spare[0]: 0 ; 0x0ec: 0x00000000
kfdhdb.ub4spare[1]: 0 ; 0x0f0: 0x00000000
……
kfdhdb.ub4spare[40]: 0 ; 0x18c: 0x00000000
kfdhdb.ub4spare[41]: 0 ; 0x190: 0x00000000
kfdhdb.ub4spare[42]: 0 ; 0x194: 0x00000000
kfdhdb.ub4spare[43]: 104436 ; 0x198: 0x000197f4
kfdhdb.ub4spare[44]: 0 ; 0x19c: 0x00000000
……
kfdhdb.ub4spare[57]: 0 ; 0x1d0: 0x00000000
kfdhdb.acdb.aba.seq: 0 ; 0x1d4: 0x00000000
kfdhdb.acdb.aba.blk: 0 ; 0x1d8: 0x00000000
kfdhdb.acdb.ents: 0 ; 0x1dc: 0x0000
kfdhdb.acdb.ub2spare: 43605 ; 0x1de: 0xaa55
```

而正常磁盘 raw5 的 kfdhdb.ub4spare[43]和 kfdhdb.acdb.ub2spare 相关标志位全为 0，是一致的状态。

```
[oracle@nccpxdb1 lib]$ kfed read /dev/raw/raw5
kfbh.endian: 1 ; 0x000: 0x01
kfbh.hard: 130 ; 0x001: 0x82
kfbh.type: 1 ; 0x002: KFBTYP_DISKHEAD
kfbh.datfmt: 1 ; 0x003: 0x01
kfbh.block.blk: 0 ; 0x004: T=0 NUMB=0x0
kfbh.block.obj: 2147483649 ; 0x008: TYPE=0x8 NUMB=0x1
kfbh.check: 1802212223 ; 0x00c: 0x6b6b937f
kfbh.fcn.base: 0 ; 0x010: 0x00000000
```

```
kfbh.fcn.wrap: 0 ; 0x014: 0x00000000
kfbh.spare1: 0 ; 0x018: 0x00000000
kfbh.spare2: 0 ; 0x01c: 0x00000000
kfdhdb.driver.provstr: ORCLDISK ; 0x000: length=8
kfdhdb.driver.reserved[0]: 0 ; 0x008: 0x00000000
kfdhdb.driver.reserved[1]: 0 ; 0x00c: 0x00000000
kfdhdb.driver.reserved[2]: 0 ; 0x010: 0x00000000
kfdhdb.driver.reserved[3]: 0 ; 0x014: 0x00000000
kfdhdb.driver.reserved[4]: 0 ; 0x018: 0x00000000
kfdhdb.driver.reserved[5]: 0 ; 0x01c: 0x00000000
kfdhdb.compat: 168820736 ; 0x020: 0x0a100000
kfdhdb.dsknum: 1 ; 0x024: 0x0001
kfdhdb.grptyp: 1 ; 0x026: KFDGTP_EXTERNAL
kfdhdb.hdrsts: 3 ; 0x027: KFDHDR_MEMBER
kfdhdb.dskname: DATAG1_0001 ; 0x028: length=11
kfdhdb.grpname: DATAG1 ; 0x048: length=6
kfdhdb.fgname: DATAG1_0001 ; 0x068: length=11
kfdhdb.capname: ; 0x088: length=0
kfdhdb.crestmp.hi: 32928501 ; 0x0a8: HOUR=0x15 DAYS=0x17 MNTH=0xc YEAR=0x7d9
kfdhdb.crestmp.lo: 2195144704 ; 0x0ac: USEC=0x0 MSEC=0x1d0 SECS=0x2d MINS=0x20
kfdhdb.mntstmp.hi: 32940275 ; 0x0b0: HOUR=0x13 DAYS=0x7 MNTH=0x8 YEAR=0x7da
kfdhdb.mntstmp.lo: 3201116160 ; 0x0b4: USEC=0x0 MSEC=0x34a SECS=0x2c MINS=0x2f
kfdhdb.secsize: 512 ; 0x0b8: 0x0200
kfdhdb.blksize: 4096 ; 0x0ba: 0x1000
kfdhdb.ausize: 1048576 ; 0x0bc: 0x00100000
kfdhdb.mfact: 113792 ; 0x0c0: 0x0001bc80
kfdhdb.dsksize: 512000 ; 0x0c4: 0x0007d000
kfdhdb.pmcnt: 6 ; 0x0c8: 0x00000006
kfdhdb.fstlocn: 1 ; 0x0cc: 0x00000001
kfdhdb.altlocn: 2 ; 0x0d0: 0x00000002
kfdhdb.f1b1locn: 0 ; 0x0d4: 0x00000000
kfdhdb.redomirrors[0]: 0 ; 0x0d8: 0x0000
kfdhdb.redomirrors[1]: 0 ; 0x0da: 0x0000
```

```
kfdhdb.redomirrors[2]: 0 ; 0x0dc: 0x0000
kfdhdb.redomirrors[3]: 0 ; 0x0de: 0x0000
kfdhdb.dbcompat: 168820736 ; 0x0e0: 0x0a100000
kfdhdb.grpstmp.hi: 32928501 ; 0x0e4: HOUR=0x15 DAYS=0x17 MNTH=0xc YEAR=0x7d9
kfdhdb.grpstmp.lo: 2195053568 ; 0x0e8: USEC=0x0 MSEC=0x177 SECS=0x2d MINS=0x20
kfdhdb.ub4spare[0]: 0 ; 0x0ec: 0x00000000
kfdhdb.ub4spare[1]: 0 ; 0x0f0: 0x00000000
.....
kfdhdb.ub4spare[38]: 0 ; 0x184: 0x00000000
kfdhdb.ub4spare[39]: 0 ; 0x188: 0x00000000
kfdhdb.ub4spare[40]: 0 ; 0x18c: 0x00000000
kfdhdb.ub4spare[41]: 0 ; 0x190: 0x00000000
kfdhdb.ub4spare[42]: 0 ; 0x194: 0x00000000
kfdhdb.ub4spare[43]: 0 ; 0x198: 0x00000000
kfdhdb.ub4spare[44]: 0 ; 0x19c: 0x00000000
kfdhdb.ub4spare[45]: 0 ; 0x1a0: 0x00000000
.....
kfdhdb.acdb.ents: 0 ; 0x1dc: 0x0000
kfdhdb.acdb.ub2spare: 0 ; 0x1de: 0x0000
```

而磁盘组 DATAG2 的两个磁盘 raw6 和 raw7 的标志位也都是全 0。

确定了问题后，手工修改这个文本文件，将 kfdhdb.ub4spare[43]和 kfdhdb.acdb.ub2spare 均改为 0。

```
kfdhdb.ub4spare[43]: 0 ; 0x198: 0x00000000
kfdhdb.ub4spare[44]: 0 ; 0x19c: 0x00000000
.
.
.
kfdhdb.acdb.ents: 0 ; 0x1dc: 0x0000
kfdhdb.acdb.ub2spare: 0 ; 0x1de: 0x0000
```

然后利用 kfed 的 merge 命令将其写回磁盘头。

```
[oracle@node1 data]$ kfed merge /dev/raw/raw4 text=raw4_new.txt
[oracle@node1 data]$ export ORACLE_SID=+ASM1
[oracle@node1 data]$ sqlplus / as sysdba
```