ORACLE 数据库高级应用丛书

ORACLE 数据高可用之路

贾代平 吴丽娟 著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内容简介

现代数据服务面临的两大问题是数据保障和不间断服务，即数据服务的高可用性（High Availability）。本书论述Oracle在此方面的两类解决方案：数据卫士（Data Guard）和数据集群（Real Application Cluster，RAC）。数据卫士将主数据库的数据变更通过异步或同步的方式传播到网络（局域网或广域网）上的另一台或多台主机上，从而实现对主数据库的数据保护。不仅如此，这些跟随主数据库数据变化的主机（备用数据库）还可以实现联机的只读访问或暂时的数据读/写，这就大大增强了数据卫士的应用价值。RAC数据集群则是将数据库同时运行在高速局域网的多个不同的主机上，这种处理方式不仅可以将应用系统的访问负荷分散到不同的服务器上，还可以通过多台主机服务之间的冗余来防范单节点故障，从而为用户提供不间断的数据访问。RAC和Data Guard的联合应用，可以实现当前IT业界最高水准的高可用性。

本书在阐述上述两类解决方案技术原理和关键知识点的基础上，以企业级应用环境为背景，详细解构Data Guard和RAC的体系架构与技术路线，包括软硬件的准备、Oracle系统的安装配置、系统管理与维护等内容。通过与案例实施过程相结合的讲解方式，带领读者领略精彩的Oracle高可用技术。

本书特别适合Oracle中高级系统管理人员、应用开发人员阅读，同时对关心服务器技术、关注数据高可用性与数据安全的企业技术人员、相关IT专业的高校教师和研究生也有重要的参考价值。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

ORACLE数据高可用之路 / 贾代平，吴丽娟著. —北京：电子工业出版社，2015.6
（ORACLE数据库高级应用丛书）
ISBN 978-7-121-26187-9

Ⅰ. ①O… Ⅱ. ①贾… ②吴… Ⅲ. ①关系数据库系统 Ⅳ. ①TP311.138

中国版本图书馆CIP数据核字（2015）第117524号

策划编辑：薄　宇
责任编辑：董亚峰
特约编辑：刘广钦
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：720×1000　1/16　印张：28.5　字数：729.6 千字
版　　次：2015年6月第1版
印　　次：2015年6月第1次印刷
定　　价：78.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前言

欢迎读者开启Oracle数据服务的高可用之路。

计算机系统的安全本质上是数据的安全，包括数据存储与数据访问的安全，因为硬件可以重建、软件可以重装，唯独数据不能有任何闪失（丢失或损坏）。作者在之前出版了一本技术著作——《Oracle数据存储与访问技术》，该著作专门探讨Oracle数据库在数据存储与数据访问方面的诸多技术问题。本书从另外一个角度再一次探讨数据存储与数据访问方面的问题，不过此次的落脚点不是存储与访问的技术本身，而是由此引发的数据安全方面的问题，包括数据存储的安全（可靠性）和数据访问的安全（连续性），即数据高可用性的两个侧面。

早期的Oracle数据库都是运行在单个主机上，用户为了确保数据访问的安全，往往采用第三方的高可用方案，常用的有双机热备和冷热互备方式。这种方式有两大缺点：一是对服务器资源的利用率较低；二是不能避免数据存储的单点故障。为了改变这种方式，Oracle公司在此领域做出了多种途径的探索来确保数据存储与数据访问的安全，如较早的Standby数据库、OPS（Oracle并行服务器）等。到目前为止，Oracle经历多次技术改进与升级，两类解决方案已经比较完美地解决了此类问题，一类是数据卫士（Data Guard），另一类是数据集群（RAC）。数据卫士解决的是数据存储上的安全问题，它通过数据库存储上的冗余来保护数据，而且这种保护是动态的、联机的、跨越地理位置和网络的。数据卫士有效地避免了服务器系统在数据存储上的单点故障。数据集群解决的是数据访问上的安全问题，即提供了应用系统的不间断数据访问，它通过将数据库运行在不同的主机上来避免数据访问的单点故障。不仅如此，数据集群还很好地解决了服务器的扩展性问题，用户可以通过相对廉价的普通服务器（甚至是PC）来增加服务器的访问节点、扩展服务器的容量和性能，从而方便地实现本来由昂贵的大型主机系统才能提供的数据服务。

在实际服务器部署中，如果能够将数据卫士和数据集群联合起来，就能够达到数据服务高可用性的更高境界，产生优势互补、相得益彰、珠联璧合的效果。这正是本书作者所期望达到的目标。全书以数据卫士为主线展开介绍，并介绍相关的技术背景，如数据恢复、日志挖掘、数据闪回、自动存储管理、RMAN等，并将数据集群作为数据卫士的应用环境来展开，最终将数据卫士和数据集群有机地结合在一起，为读者带来Oracle数据服务高可用性的完整解决方案。

本书的写作得到国家自然科学基金（61272244）、山东省自然科学基金（ZR2013FL018）的支持，全书由贾代平同志主笔，吴丽娟同志写作了第1章和第17章，安志勇同志为本书的写作提供了多方面的协助。本书的出版得到电子工业出版社及薄宇编辑的大力支持，在此一并表示感谢。

读者在阅读本书的过程中，若有任何问题，可以通过邮件（jiadp@oraclechina.org）与作者联系，我们将提供及时的技术服务。

作　者

2015年3月

第 1 章 Data Guard 技术概要

1.1 什么是 Data Guard 001
1.2 Data Guard 的主要功能 001
1.3 Data Guard 进程结构与环境 002
1.4 数据变更与备用方式 003
1.4.1 物理备用数据库 003
1.4.2 逻辑备用数据库 004
1.4.3 两种备用方式的比较 005
1.5 Active Data Guard 功能 006
1.6 角色转换与故障切换 007
1.7 Data Guard 的其他特性 008
1.8 Data Guard 的技术优势 009
1.9 和远程镜像方案的比较 010

第 2 章 基于事务日志的数据恢复技术

2.1 事务处理与联机日志 012
2.2 归档日志与运行模式 013
2.3 事务日志用于数据恢复的核心机制 014
2.4 Oracle 数据库介质恢复框架 015
2.5 完全数据恢复的黄金法则 017

第 3 章 重做数据与日志挖掘

3.1 Oracle 日志挖掘介绍 019
3.1.1 LogMiner 的配置框架 019
3.1.2 LogMiner 的用户接口 020
3.1.3 补充日志 Supplemental Logging 021
3.2 确定 LogMiner 字典 023
3.3 建立日志文件列表 024
3.4 启动 LogMiner 日志挖掘 026
3.5 查看日志挖掘的结果 027
3.6 控制事务输出 029
3.7 LogMiner 会话及其步骤 031
3.8 日志挖据典型案例 034
案例 1 挖掘已提交的事务 034
案例 2 限定重做数据的范围 034
案例 3 跟踪特定用户的数据处理 037
案例 4 统计特定表上的用户访问 037
3.9 日志挖掘不支持的数据类型 038

第 4 章 Data Guard 中的进程架构

4.1 DG 环境的进程架构概述 039
4.2 备用数据库的实现框架 040
4.3 日志传输服务 042
4.3.1 异步日志传输（ASYNC） 042
4.3.2 同步日志传输（SYNC） 042
4.3.3 日志间隔（Gap）的自动处理 043
4.4 事务日志的应用服务 044
4.4.1 Redo 应用 044
4.4.2 SQL 应用 044

第 5 章 构建物理备用数据库

5.1 主数据库的准备 047
5.1.1 检查并设置数据库 047
5.1.2 设置必要的主数据库参数 049
5.2 备用数据库控制文件 051
5.3 构造备用数据库运行环境 054
5.4 启动日志传输和应用服务 058
5.4.1 主数据库端启动日志传输 058
5.4.2 备用端启动 Redo 应用 058
5.4.3 数据更新测试 060
5.5 DG 后续配置的考虑 062

第 6 章 构建逻辑备用数据库

6.1 对主数据库的检查 063
6.1.1 检查不支持的模式 063
6.1.2 检查不支持的用户表 064
6.1.3 检查存在唯一性问题的表 066
6.2 准备物理备用数据库 068
6.2.1 主备用数据库参数的准备 069
6.2.2 物理存储的准备 070
6.2.3 备用实例的启动与还原 073
6.2.4 同步物理备用数据库 079
6.3 激活逻辑备用数据库 079
6.3.1 构建 LogMiner 字典 080
6.3.2 Standby 类型的转换 080
6.3.3 重置新的 Incarnation 082
6.3.4 启动 SQL Apply 084

第 7 章 日志传输与应用服务

7.1 配置日志传输服务 088
7.2 重做数据的接收 089
7.3 备用端的日志应用 092
7.4 日志间隔的手工处理 094
7.5 数据保护模式 096
7.5.1 最大保护模式（Maximum Protection） 096
7.5.2 最高可用性保护（Maximum Availability） 096
7.5.3 最高性能保护（Maximum Performance） 097
7.5.4 保护模式的实施与转换 097
7.6 监控物理备用数据库 101
7.6.1 进程名称及其状态 101
7.6.2 物理备用端的应用进展 102
7.6.3 重要的相关视图 103
7.7 重新创建物理备用控制文件 103

第 8 章 逻辑备用数据库的管理

8.1 逻辑备用的状态 105
8.2 SQL Apply 的内部设置 106
8.2.1 挖掘服务的调整 108
8.2.2 应用服务的调整 110
8.2.3 LCR 缓存的调整 110
8.3 控制逻辑备用维护的数据集 111
8.3.1 设置备用数据库的防护 111
8.3.2 逻辑备用不支持的数据集 112

8.3.3 自定义逻辑备用维护的数据集 114
8.4 判断逻辑备用不维护的表 116
8.5 逻辑备用故障排除实例 120

第 9 章
物理备用数据库的新特性

9.1 物理备用的只读模式 125
9.2 物理备用的读 / 写模式 126
9.2.1 打开前的条件准备 126
9.2.2 激活物理备用库 128
9.2.3 返回到物理备用状态 129
9.2.4 处理主库的日志中断 131
9.3 快照备用数据库 138
9.4 Active Data Guard 141
9.4.1 备用数据库的读与写 141
9.4.2 实现活动备用数据库 141
9.4.3 活动备用数据库应用 143

第 10 章
角色切换与故障转移

10.1 角色切换与故障转移技术概要 145
10.1.1 角色切换（Switch Over） 145
10.1.2 故障转移（Fail Over） 145
10.1.3 角色切换注意事项 146
10.2 故障转移与数据损失 147
10.3 数据库闪回与 Data Guard 147
10.4 闪回日志导致的数据库故障 148
10.5 角色切换前的准备工作 151
10.5.1 检查日志传输 151
10.5.2 检查备用端的日志应用 152
10.5.3 检查会话和后台作业 153
10.6 物理备用与主数据库的角色切换 155
10.6.1 网络配置与角色参数 155
10.6.2 主数据库中的工作 155
10.6.3 物理备用数据库中的工作 160
10.6.4 主数据库与物理备用 SWITCHOVER 小结 162
10.7 逻辑备用与主数据库的角色切换 164
10.7.1 准备切换阶段 164
10.7.2 执行角色切换 166
10.7.3 主数据库与逻辑备用数据库 SWITCHOVER 小结 168
10.8 故障转移至备用数据库 169
10.8.1 备用数据库的选择和处理 169
10.8.2 转移至物理备用数据库 170
10.8.3 转移至逻辑备用数据库 174
10.8.4 原有主数据库的再利用 181

第 11 章
Data Guard 的主要参数、视图与管理指令

11.1 初始化参数及其设置 189
11.1.1 Data Guard 主要参数 189
11.1.2 参数 LOG_ARCHIVE_DEST_n 190
11.1.3 参数 LOG_ARCHIVE_TRACE 191
11.2 Data Guard 视图 192
11.3 Data Guard 的管理指令 193
11.3.1 管理数据库 193
11.3.2 管理实例 195
11.3.3 管理会话 196

11.3.4 常用指令流程图 196
11.4 PL/SQL 程序包 198
11.4.1 DBMS_LOGMNR 程序包 199
11.4.2 DBMS_LOGMNR_D 程序包 200
11.4.3 DBMS_LOGSTDBY 程序包 200

第 12 章 RMAN 备份与恢复技术

12.1 熟悉 RMAN 环境 203
12.1.1 RMAN 环境的构成 203
12.1.2 目标数据库的配置 206
12.1.3 目录数据库的配置 206
12.1.4 配置 RMAN 执行环境 209
12.1.5 配置备份通道与存储 213
12.1.6 磁带通道的磁盘模拟 215
12.2 RMAN 备份的主要形式 217
12.2.1 RMAN 备份的优点 217
12.2.2 备份目标和备份结果 218
12.2.3 执行 RMAN 备份 BACKUP 218
12.2.4 Plus Archivelog 解析 220
12.3 执行 RMAN 增量备份 225
12.3.1 启动块改变跟踪 225
12.3.2 增量备份的类别 226
12.3.3 基于 SCN 的增量备份 227
12.4 RMAN 备份的其他选项 227
12.4.1 Backup 指令的其他选项 227
12.4.2 限定 RMAN 备份的过程 229
12.5 管理 RMAN 的备份结果 229
12.5.1 查看备份结果 LIST 230
12.5.2 关于备份的报告 234
12.5.3 备份的状态与交叉检验 236
12.5.4 在 RMAN 中注册备份 237

第 13 章 RMAN 与 Data Guard

13.1 RMAN 复制与备用数据库 239
13.1.1 复制的必要准备 239
13.1.2 为物理备用而复制 240
13.2 Data Guard 环境中的 RMAN 配置 245
13.2.1 唯一性标识 DB_UNIQUE_NAME 245
13.2.2 为主备库配置 RMAN 247
13.3 卸载备份至备用数据库 250
13.3.1 RMAN 备份的数据库相关性 250
13.3.2 使用备用端备份恢复主数据库 254
13.3.3 使用备用端数据文件恢复主数据库 260

第 14 章 闪回数据库与 Data Guard

14.1 数据库闪回功能 264
14.1.1 闪回日志与数据库闪回 264
14.1.2 还原点和闪回窗口 265
14.2 配置数据库闪回和还原点 267
14.2.1 闪回和保证还原点的条件 267
14.2.2 启动数据库闪回功能 267
14.3 闪回恢复区的维护 271
14.3.1 闪回恢复区的删除规则 271
14.3.2 监控与管理闪回恢复区 272
14.4 闪回数据归档 273

14.5 数据库闪回与数据恢复 274
14.5.1 当前演化路径下的闪回 276
14.5.2 闪回到 Incarnation 演化分支 278
14.6 数据库闪回在 Data Guard 中的应用 280
14.6.1 基于保证还原点的应用 281
14.6.2 使用快照备用数据库 284

第 15 章 集群数据库系统 RAC

15.1 RAC 体系结构 289
15.1.1 RAC 总体框架图 290
15.1.2 共享存储部分 291
15.1.3 集群件（Clusterware） 292
15.1.4 网络访问部分 293
15.2 OCR 与 Voting Disk 294
15.2.1 集群注册表（OCR） 294
15.2.2 表决磁盘（Voting Disk） 296
15.2.3 本地集群注册表（OLR） 297
15.3 构造 Clusterware 集群平台 298
15.3.1 主机环境概要 298
15.3.2 ASMLib 驱动与 ASM 磁盘 312
15.3.3 网格基础架构（Grid Infrastructure） 314
15.4 创建 RAC 集群数据库 317
15.4.1 选择 DBMS 软件 317
15.4.2 集群数据库的特有设置 319
15.4.3 RAC 数据库的形成过程 320
15.5 RAC 的缓存融合技术 323
15.5.1 全局资源目录（GRD） 324
15.5.2 GCS 和 GES 324
15.5.3 RAC 后台进程 325

第 16 章 RAC 环境下的 Data Guard

16.1 RAC 主机环境描述 326
16.1.1 基于 RAC 的 Dataguard 框架 326
16.1.2 主备数据库及其实例配置 327
16.1.3 建立 Dataguard 前的基础条件 328
16.2 Primary 端 RAC 数据库的准备 330
16.2.1 检查和调整主数据库的运行 330
16.2.2 备份主数据库 331
16.2.3 主数据库端的网络配置 333
16.2.4 主数据库端的参数调整 335
16.3 Standby 端 RAC 数据库的准备 335
16.3.1 准备备用数据库实例 336
16.3.2 还原并启动备用数据库 339
16.3.3 创建备用重做日志 341
16.3.4 向集群中注册 RAC 备用数据库 342
16.4 启动备用实例的托管恢复 343
16.4.1 检查日志传输 343
16.4.2 启动日志应用 344
16.5 RAC 备用数据库的应用 346
16.5.1 RAC 备用冷集群 346
16.5.2 RAC 活动备用数据库 346

第 17 章 由 ASM 到 RAC+DG 的高可用之路

17.1 GI 独立服务器基础架构及其 ASM 数据库 348

17.1.1 安装独立服务器网格基础架构 349
17.1.2 安装配置 DBMS 软件和数据库 360
17.1.3 迁移数据库至 ASM 存储 361
17.2 Data Guard 及其备用数据库诞生记 367
17.2.1 创建物理备用数据库的基本过程 367
17.2.2 创建逻辑备用数据库的基本过程 380
17.3 RAC 集群数据库的构造过程 388
17.3.1 单节点 GI for Cluster 的安装和配置 388
17.3.2 Oracle 数据库软件的安装 398
17.3.3 创建集群数据库 399
17.3.4 向 OCR 注册集群数据库 406
17.3.5 扩展集群至新节点 414
17.3.6 删除集群中的节点 425
17.4 典型的 RAC+Data Guard 高可用环境 MAA 430
17.4.1 RAC 主数据库端环境描述 430
17.4.2 为创建备用数据库做准备 433
17.4.3 启动备用端实例 436
17.4.4 在备用端还原数据库 440
17.4.5 启动托管恢复 444
17.4.6 主备用角色切换 445
参考文献 446

第1章

Data Guard 技术概要

信息系统的停机分为计划内和计划外两类。信息系统计划外停机事件可能是由硬件或系统故障、数据 / 存储故障、人为错误、计算机病毒、软件故障、自然灾害或恶意行为引起的。在数据库服务器没有异机或远程保护的情况下，信息系统的计划内停机不可避免，此时系统不得不承受因计划维护（如系统升级）而造成的业务中断。业务连续性和灾难恢复是信息系统架构设计最先需要考虑的问题。Data Guard 是保障信息系统数据库服务器连续运行和灾难恢复的综合解决方案，它能够确保数据安全并保障系统 7×24 小时运行。

1.1 什么是 Data Guard

Data Guard 是管理、监控和自动化处理 Oracle 数据库服务器的基础架构，它创建、维护和监控一个或多个备用数据库，以保护企业数据结构不受故障、灾难、错误和崩溃的影响。Data Guard 使备用数据库保持为与生产数据库在事务上一致的副本。这些备用数据库既可能位于距生产数据中心千里之外的远程灾难恢复站点，也可能位于同一城市、同一单位乃至同一建筑物内。当生产数据库由于计划中断或意外中断而变得不可用时，Data Guard 可以将任意备用数据库切换到生产角色，从而使与中断相关的停机时间减到最少甚至不停机，并可有效防止任何数据丢失。Data Guard 还能够与其他的 Oracle 高可用性（HA）解决方案（如真正应用集群（RAC）和恢复管理器（RMAN））结合使用，以提供数据库服务器前所未有的高水平数据保护和数据可用性。

1.2 Data Guard 的主要功能

Data Guard 包括一个生产数据库，也称为主数据库（Primary Database），以及一个或多个备用数据库（Standby Database），这些备用数据库是与主数据库在事务上一致的副本。Data Guard 利用重做数据保持这种事务一致性。当主数据库中发生事务时，则生成重做数据并将其写入本地重做日志文件中。通过 Data

Guard，还将重做数据传输到备用站点上，并应用到备用数据库中，从而使备用数据库与主数据库保持同步。Data Guard 允许 DBA 选择将重做数据同步还是异步地发送到备用站点上。备用数据库的底层技术是 Data Guard 重做应用（物理备用数据库）和 Data Guard SQL 应用（逻辑备用数据库）。物理备用数据库在磁盘上拥有和主数据库逐块相同的数据库结构，并且使用 Oracle 介质恢复进行更新。逻辑备用数据库是一个独立数据库，它与主数据库包含相同的数据。它使用 SQL 语句进行更新，其相对优势是能够并行用于恢复，以及诸如报表、查询等其他任务（从 Oracle 11g 开始，物理备用数据库也可以实现只读查询）。Data Guard 简化了主数据库和选定的备用数据库之间的转换和故障切换，从而减少了由计划停机和计划外故障所导致的总停机时间。主数据库和备用数据库，以及它们的各种交互可以使用 SQL*Plus 来进行管理。为了获得更高效的可管理性，Data Guard 还提供了一个分布式管理框架（称为 Data Guard Broker），它不但自动化了 Data Guard 配置的创建、维护和监控，并对这些操作进行统一管理。DBA 可以使用 Oracle Enterprise Manager 或 Broker 自己的专用命令行界面 (DGMGRL) 来管理 Data Guard 环境。

1.3 Data Guard 进程结构与环境

在主数据库上，Data Guard 使用日志写入器进程 (LGWR) 或归档器进程 (ARCH) 收集事务重做数据，并将其传输到备用数据库中；使用获取归档日志进程 (FAL) 提供一个客户服务器机制，用于在主数据库和备用数据库之间出现通信中断之后将归档日志发送到备用数据库中，以实现自动填充间隔和重新同步。

在备用数据库上，Data Guard 使用远程文件服务器 (RFS) 进程从主数据库接收重做记录；使用管理恢复进程 (MRP) 将重做信息应用到物理备用数据库中；使用逻辑备用进程 (LSP) 将重做数据经过 SQL 转换后应用到逻辑备用数据库中。如果启动了 Data Guard Broker，Oracle Data Guard 还使用 Data Guard Broker Monitor 进程将主数据库和备用数据库作为一个统一的配置进行管理和监控。

Data Guard 环境包括一个主数据库和最多 9 个备用数据库。主数据库和备用数据库可以是单实例数据库或真正应用集群 (RAC) 数据库。备用数据库使用 Oracle 网络服务通过基于 TCP/IP 的标准网络（如局域网 (LAN)、城域网 (MAN)、广域网 (WAN)）连接到主数据库。对备用数据库所处位置没有限制，只要它们能够接收来自主数据库的重做数据即可。不过，对于灾难恢复，建议将备用数据库装载在地理上与主站点相分离的站点上。Data Guard 一般要求主系统和备用系统上的操作系统平台一致，例如，如果主数据库是运行在 Intel 架构上的 Linux 操作系统，则其所有备用数据库也应该运行在 Intel 架构上的 Linux 操作系统。此外，Data Guard 环境中的主数据库和所有备用数据库必须运行在相同版本的 Oracle DBMS 软件企业版上。

1.4　数据变更与备用方式

备用数据库最初是从主数据库的一个备份副本创建的。一旦创建了备用数据库，Data Guard 自动将主数据库重做数据传输给备用系统，然后将重做数据应用到备用数据库中，从而使备用数据库保持为与主数据库在事务上一致的副本。Data Guard 提供了两种方法将这些重做数据应用到备用数据库中，这些方法与 Data Guard 支持的如下两种类型的备用数据库对应。

※ Redo Log 应用：用于物理备用数据库。

※ SQL 应用：用于逻辑备用数据库。

从数据变更传播的方式看，无论采用哪一种应用方式，在主数据库方面两者没有区别。一旦重做数据到达备用服务器端，不同的应用方式决定了备用数据库的数据更新方式。

1.4.1　物理备用数据库

通过使用 Oracle 介质恢复应用从主数据库接收到的重做数据，物理备用数据库与主数据库保持同步。它在物理上与主数据库逐块相同，因而数据库模式（包括索引）是相同的。

主数据库上的一个日志切换将触发备用数据库上的一个日志切换，从而使备用数据库上的归档器进程将当前的备用重做日志文件归档到备用数据库上的一个归档日志中。随后，Data Guard 重做应用使用一个专用进程（称为管理的恢复进程 MRP）读取归档日志，并将重做数据应用到物理备用数据库中。如果启动了从 10g 开始诞生的 Data Guard 实时应用功能，则 MRP 将在 RFS 进程将日志写进备用重做日志文件的过程中直接从其中读取重做数据并实施重做应用。

物理备用数据库通过 Redo 应用来保持与 Primary 数据库的一致性，所谓 Redo 应用，实质是通过 Oracle 的基于日志的恢复机制，应用归档文件（或 Standby Redologs 文件）中的 Redo 数据。恢复操作属于块对块的应用。如果正在执行 Redo 应用的操作，则 Oracle 数据库就不能被打开。

可以将物理备用数据库以只读模式打开。当备用数据库以只读模式打开时，传送给它的重做数据将在备用站点上累积而不应用。不过，可以随时在物理备用数据库上恢复操作，并自动应用累积的重做数据。这允许物理备用数据库以一个序列运行，这个序列可能包括在恢复中运行一段时间，然后以只读模式打开来运行报表，接着重新运行恢复来应用尚未应用的重做数据。物理备用数据库在以只读模式打开后，可以在 Standby 数据库执行查询或备份等操作（变相减轻 Primary 数据库的压力）。如果需要的话，可以在两种状态之间转换，如先应用 Redo，然后将数据库置为只读状态，需要与 Primary 同步时再次执行 Redo 应用命令，切换回 Redo 应用状态。

在启动了数据库闪回功能的前提下，物理备用数据库可以读 / 写模式打开，那

么 Standby 数据库将暂停从 Primary 数据库接收 Redo 数据，并且暂时失去灾难保护的功能。当然，以读 / 写模式打开也并非一无是处，如在需要临时调试一些数据，但又不方便在正式库中操作时，就可以临时将 Standby 数据库置为读 / 写模式，操作完之后将数据库闪回到操作前的状态（闪回之后，Data Guard 会自动同步，并不需要重建物理备用数据库）。

值得注意的是，从 Oracle 11g 版本开始，Data Guard 显著增强了物理备用数据库的应用功能，物理备用数据库可以在 Open Read Only 模式下继续应用 Redo 数据，这就极大地提升了物理备用数据库的应用场合。

1.4.2 逻辑备用数据库

逻辑备用数据库也是通过 Primary 数据库（或其备份及复制库，如物理备用数据库）创建，因此，在创建之初与物理备用数据库类似。不过由于逻辑备用数据库通过 SQL 应用的方式应用 Redo 数据，因此，逻辑备用数据库的物理文件结构甚至数据的逻辑结构都可以与 Primary 不一致。

与物理备用数据库不同，逻辑备用数据库正常情况下是以读 / 写模式打开的，用户可以在任何时候访问逻辑备用数据库，即逻辑备用数据库是在打开状态执行 SQL 应用的。有利也有弊，由于 SQL 应用自身的特点，逻辑备用数据库对于某些数据类型及一些 DDL/DML 语句会有操作上的限制。可以在 DBA_LOGSTDBY_UNSUPPORTED 视图中查看不支持的数据类型，如果使用了这种数据类型，则不能保证数据库完全一致。

尽管数据的物理组织和结构可能不同，但逻辑备用数据库包含与主数据库相同的逻辑信息。SQL 应用技术将从主数据库接收到的重做数据转换成 SQL 语句，然后在备用数据库上执行 SQL 语句，以使逻辑备用数据库与主数据库保持同步。从而，在将 SQL 应用到逻辑备用数据库上的同时，可以访问逻辑备用数据库来进行查询和报表操作。由于使用 SQL 语句更新逻辑备用数据库，因此，它保持以读 / 写模式打开，而从主数据库中更新的表可以同时用于诸如报表、统计、查询等其他任务，如通过在维护的表上创建额外的索引和物化视图来优化这些任务。

Oracle 可以按模式来更新逻辑备用数据库中的数据，因此，可以选择更新某些模式而保持另外一些模式不受主数据库更新的影响。

SQL 应用使用多个并行的执行服务器和后台进程，它们将来自主数据库的更改应用到逻辑备用数据库中。

默认情况下，Reader 进程从归档日志（如果启动了实时应用，也可以是备用重做日志）中读取重做记录。Preparer 进程将块更改转换成表更改或逻辑更改记录 (LCR)。在这里，LCR 并不代表任何特定的事务。Builer 进程对来自各个 LCR 的已完成事务进行组合。Analyzer 进程检查完成的事务，辨明不同事务之间的相关性。Coordinator 进程（也称为逻辑备用进程，即 LSP）负责将事务分配给应用进程、

监控事务之间的相关性，以及批准将更改提交给逻辑备用数据库。Applier 进程将已指定事务的 LCR 应用到数据库中，并在 Coordinator 指示提交事务时提交。

如图 1-1 所示为 Data Guard SQL 应用的进程结构。

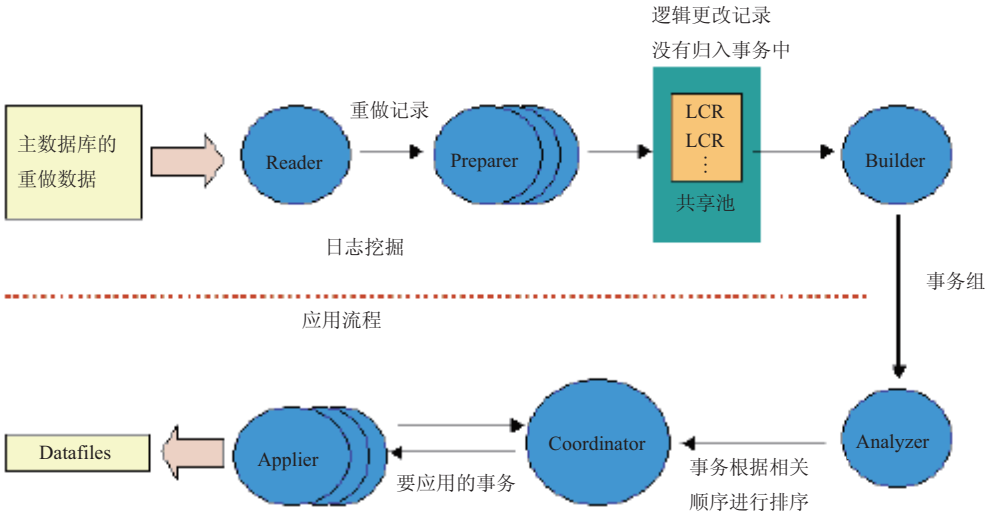


图 1-1　Data Guard SQL 应用的进程结构

1.4.3　两种备用方式的比较

物理备用数据库的特点如下：

（1）灾难恢复及高可用性。物理备用数据库提供了一个健全、高效的灾难恢复，以及高可用性的解决方案。更加易于管理 switchover/failover 角色转换及在更短的计划内或计划外停机时间。

（2）数据保护。使用物理备用数据库，Data Guard 可确保即使在遇到不可预见的灾难时也不会丢失数据。物理备用数据库支持主数据库上的所有数据类型，以及 DDL 和 DML 操作。它还提供了防止数据损坏和用户错误的安全保护。主数据库上存储介质上的物理损坏不会传播到备用数据库上。同样，导致主数据库永久损坏的逻辑损坏或用户错误也能够得到解决。最后，在将重做数据应用到备用数据库时对其进行验证。

（3）分担 Primary 数据库压力。通过将一些备份任务、查询需求等转移到物理备用数据库，可以有效节省 Primary 数据库的 CPU 及 I/O 资源。可以以只读方式打开物理备用数据库，来生成报表和进行查询。此外，利用恢复管理器（RMAN），可以利用物理备用数据库为生产数据库创建备份。

（4）提升性能。物理备用数据库所用的重做应用技术使用底层的基于日志的恢

复机制来应用更改，这种机制绕过了所有 SQL 级代码层，因此，是最有效的应用更改机制。这使得重做应用技术成为一种在数据库之间传播更改的极其有效的机制。

逻辑备用数据库还有下列一些特点：

（1）有效地利用备机的硬件资源。除灾难恢复外，逻辑备用数据库还可用于其他业务需求。例如，通过在 Standby 数据库创建额外的索引、物化视图等提高查询性能并满足特定业务需要；又如，创建新的 SCHEMA（该 SCHEMA 在 Primary 数据库端并不存在），然后在这些 SCHEMA 中执行那些不适于在 Primary 数据库端执行的 DDL 或者 DML 操作等。由于可以将受 Data Guard 保护的逻辑备用表存储在一个与主数据库不同的物理布局中，因此，可以创建额外的索引和物化视图来提高查询性能并满足特定的业务要求。

（2）分担 Primary 数据库压力。逻辑备用数据库可以在保持与 Primary 同步时仍然置于打开状态，这使得逻辑备用数据库能够同时用于数据保护和报表操作，从而将主数据库从报表和查询任务中解脱出来，节约宝贵的 CPU 和 I/O 资源。

（3）平滑升级。可以通过逻辑备用数据库来实现如跨版本升级、安装 DBMS 补丁等操作。

1.5 Active Data Guard 功能

如果把逻辑备用数据库和物理备用数据库的优势结合在一起，既能发挥物理备用数据库同步速度快、耗用资源低，又能够在数据同步的同时，进行数据查询，从而充分发挥备用系统的资源利用率。这就是从 Oracle 11g 开始诞生的 Active Data Guard 的功能。

在 Oracle 11g 以前的 Data Guard 物理备用数据库，可以以只读的方式打开数据库，但是这时 Media Recovery 利用日志进行数据同步的过程就停止了，如果物理备用数据库处于恢复的过程中，数据库就不能打开查询，也就是说日志应用和只读打开两个状态是互相排斥的，而 Active Data Guard 功能解决了这个矛盾，在利用日志恢复数据的同时可以以只读的方式打开数据库，用户可以在备用数据库中进行查询、报表等操作，这类似逻辑备用数据库的功能，但是，数据同步的效率更高、对硬件的资源要求更低。这样可以更大程度地发挥物理备用数据库的硬件资源的效能（如对于实时要求比较高的报表服务）。

Active Data Guard 除了可以运行在只读打开和日志同步应用的场景下，还可以切换到 Snapshot Standby 状态下运行。运行在 Snapshot Standby 状态下的物理备用数据库可以以读 / 写模式打开数据库，但是同时没有破坏它作为物理备用数据库的功能，这个特性可以用来在物理备用数据库上执行某些测试，测试完成后，把数据库再切换到 Physical Standby 状态下，又可以自动利用日志实现数据同步了。当然在 Snapshot Standby 以读 / 写方式打开时，它只能接收生产

数据库传过来的日志，但是不能应用这些日志进行数据同步。

例如，把数据库从数据同步和只读打开状态切换到快照备用状态，只需要在备用数据库上执行 Alter database convert to snapshot standby 语句；而把数据库从快照备用切换回只读同步状态，只需要执行 Alter database convert to physical standby 语句即可。

1.6　角色转换与故障切换

Data Guard 提供了两种易于使用的方法来处理生产站点的计划内停机和计划外中断，这些方法称为角色转换和故障切换。DBA 可以使用 Oracle Enterprise Manager GUI 界面、Data Guard Broker 的命令行界面或直接通过 SQL 来轻松地启动这些方法。故障切换是在主数据库上出现计划外的灾难性故障，且不可能及时恢复主数据库时将一个联机的备用数据库用作新的主数据库的操作。

故障切换操作在将承担主数据库角色的备用数据库上启动。如果在故障恢复之前在原来的主数据库上启动了闪回数据库功能，并且打算将原来的主数据库恢复为 Data Guard 配置中的一个新的备用数据库，则可以在故障恢复之后使用此功能。如果在故障切换之前没有启动闪回数据库，而需要在故障切换之后将原来的主数据库恢复为一个备用数据库，则必须从新主数据库的一个备份副本重新创建该备用数据库。

角色转换是主数据库和备用数据库之间计划的角色转换，用于处理主数据库上的计划维护。转换操作和故障切换操作之间的主要差异是，转换是在主数据库仍然可用时进行，且不需要闪回或重新安装原来的主数据库。这允许原来的主数据库几乎立即承担起备用数据库的角色。因而，可以更轻松、更频繁地执行计划维护。例如，当主站点上升级硬件时，转换功能通过将所有数据库客户机转换到备用站点，以在主站点上执行升级。

角色转换操作始终确保在转换期间不会丢失数据。如果 Data Guard 运行在最大保护模式或最高可用性模式下，故障切换操作确保在故障切换时不会丢失数据。应当强调的是，为了避免在出现临时系统故障或网络故障的情况下进行错误的故障恢复 / 转换，Data Guard 转换和故障恢复操作不是自动进行的，而必须由 DBA 明确启动。一旦启动，Data Guard 就自动运行相关的进程。

当配置中包含多个备用数据库时，故障切换和角色转换操作可以无缝地进行。例如，如果在配置了多个备用数据库的情况下主数据库出现故障，则 DBA 可以灵活地选择一个可用的备用数据库，使其成为主数据库。Data Guard 使重定向其他备用数据库来使用新主数据库的过程（包括传输任意丢失或不完整的重做数据）完全自动化。

1.7 Data Guard 的其他特性

1．自动重新同步

如果主备用数据库之间的网络出现中断，Data Guard 可以顺畅地处理将备用（物理或逻辑）数据库与主数据库暂时断开的网络连接问题。当备用数据库变为不可用时（除非这个备用数据库是最大保护模式下的最后一个可用的备用数据库，在这种情况下主数据库将关闭），则在主数据库本地捕获事务。当重新建立与备用数据库的连接时，将自动传输累积的归档日志，并将其应用到备用数据库中，直到备用数据库已经与主数据库重新同步。这个过程不需要任何管理干预。Oracle 建议，如果在主站点附近经常发生网络中断，则网络容量应足够处理这种重新同步。

2．用户错误的防护

当主数据库打开并处于活动状态时，事务处理正在进行，生成重做数据，并将其传输到备用站点上。鉴于人为错误是造成系统停机的主要原因，这种重做数据可能包含重大的逻辑用户错误，如丢失一个重要的表，这可能使主数据库崩溃。

Data Guard 提供了几种易于使用的方式来避免这种用户错误。DBA 可以决定在主数据库和备用数据库上同时使用闪回数据库功能，以快速将数据库恢复到一个较早的时间点上，从而取消这种用户错误。另外，如果 DBA 决定故障切换到一个备用数据库上，但那些用户错误已经应用到了备用数据库中（例如，由于启动了实时应用），则 DBA 可以简单地将备用数据库闪回到一个安全的时间点上（假定在备用数据库上已经启动了闪回功能）。最后，DBA 还有额外的选择，可以不在一个或多个备用数据库上使用实时应用特性，相反使重做数据在那些备用数据库上的应用延迟一段可配置的时间，这提供了一个防止这种用户错误或损坏的保护窗口。不管选择哪个选项，备用数据库上的应用进程将始终重新验证日志记录，以防止将物理重做数据损坏应用到备用数据库上。

3．级联重做日志的传输

在这种情况下，一个备用数据库是从另一个备用数据库而非原始主数据库接收重做数据。因为主数据库仅将重做数据发送给备用数据库的一个子集，所以，这一特性不仅减少了主系统上的负载，还减少了主站点周围的网络流量和对宝贵的网络资源的使用。

4．Data Guard 和 RAC

Data Guard 和 Oracle 真正应用集群（RAC）彼此互补。RAC 解决系统或例程故障。它提供不影响数据的故障（如节点故障或例程崩溃）的快速和自动恢复。它还为应用程序提供增强的可伸缩性。另一方面，Data Guard 通过使用在事务上一致的主数据库和备用数据库——它们既不共享磁盘也不运行在锁步模式下——提

供数据保护。这使得从站点灾难或数据损坏中恢复成为可能。Data Guard 还生来就与 RAC 集成，例如，一些或所有主 / 备用（物理或逻辑）数据库可以是 RAC 数据库，并且可以使用 Enterprise Manager、Broker 的命令行界面或直接使用 SQL 来管理这些数据库。客户应当把 Oracle Data Guard 和真正应用集群结合使用来获得数据级和系统级保护的好处。

1.8　Data Guard 的技术优势

Data Guard 是数据库系统高可用性架构（High Availability Architecture）的核心组成部分，在 Oracle 数据库服务器中实施 Data Guard 容灾方案具有如下几个方面的明显技术优势。

1．灾难恢复和高可用性

Data Guard 提供了一个高效和全面的灾难恢复和高可用性解决方案。易于管理的转换和故障切换功能允许主数据库和备用数据库之间的角色转换，从而使主数据库因计划内和计划外的中断所导致的停机时间减到最少。

2．完善的数据保护

使用备用数据库，Data Guard 可保证即使遇到不可预见的灾难也不会丢失数据。备用数据库提供了防止数据损坏和用户错误的安全保护。主数据库上的存储器级物理损坏不会传播到备用数据库上。同样，导致主数据库永久损坏的逻辑损坏或用户错误也能够得到解决。最后，在将重做数据应用到备用数据库时会对其进行验证。

3．有效利用系统资源

备用数据库表使用从主数据库接收到的重做数据进行更新，并且可用于诸如备份操作、报表、合计和查询等其他任务，从而减少执行这些任务所必需的主数据库工作负载，节省宝贵的 CPU 和 I/O 周期。使用逻辑备用数据库，用户可以在模式中不从主数据库进行更新的表上执行数据处理操作。逻辑备用数据库可以从主数据库中对表进行更新时保持打开，并可同时对表进行只读访问。最后，可以在维护的表上创建额外索引和物化视图，以获得更好的查询性能和适应特定的业务要求。

4．灵活的数据保护功能

可以根据信息系统的需要实施不同的数据保护级别，从而在可用性与性能要求之间取得平衡。Data Guard 提供了最大保护、最高可用性和最高性能等模式，来帮助企业在系统性能要求和数据保护之间取得平衡。

5．自动间隔检测及其解决方案

如果主数据库与一个或更多个备用数据库之间的连接丢失（例如，由于网络问

题），则在主数据库上生成的重做数据将无法发送到那些备用数据库上。一旦重新建立连接，Data Guard 就自动检测丢失的归档日志序列（或间隔），并将必要的归档日志自动传输到备用数据库中。备用数据库将重新与主数据库同步，而无须 DBA 的任何手动干预。

6．集中管理主备用数据库

主备用数据库是一个分布式系统，Data Guard Broker 可以集中管理和监控 Data Guard 环境中的多个数据库。系统 DBA 可以使用 Oracle Enterprise Manager 或 Broker 专用的命令行界面（DGMGRL）来利用这个集成的管理框架。

1.9 和远程镜像方案的比较

远程镜像解决方案从概念上看好像是要提供简单而完善的数据保护。然而，在数据保护方面，Data Guard 在本质上要比远程镜像解决方案更有效、更便宜、更优化。客户不需要购买远程镜像解决方案或将远程镜像解决方案与 Oracle Data Guard 集成。下面是 Data Guard 与远程镜像解决方案的比较说明。

1．更高的网络效率

使用 Oracle Data Guard，只需将重做数据发送到远程站点上。然而，如果使用远程镜像解决方案进行数据保护，则必须为数据库文件、在线日志、归档日志和控制文件创建镜像。这意味着远程镜像将把每次更改至少发送到远程站点上 3 次。此外，数据库写操作将比日志写操作更频繁地发生，这是因为每次日志写操作一般包含许多更改（称为组提交）。这意味着一个基于数据库重做传输的解决方案所需的网络带宽要比一个远程镜像解决方案少得多。更重要的是，这意味着少得多的网络回程。远程镜像对于非数据库文件可能非常有用，但对于数据库数据，更好地保护和更低成本的结合是使用 Data Guard 的令人信服的原因。

2．更低的网络要求

由于存储系统使用的底层通信技术（光纤、ESCON），基于存储系统的远程镜像解决方案通常有距离的限制。可以使用来自第三方供应商的专门设备来扩展这个距离。这些设备将 ESCON/ 光纤转换成相应的 IP、ATM 或 SONET 网络。问题是，每增加一台这种设备，就在系统中引入延迟，从而影响生产数据库的性能，并使得这种配置不适用于零数据丢失功能所必需的同步传输。可以通过在通信路径中引入中间存储盒来减轻这个问题，但这增加了总体成本。

另一种解决方案是使用变化的同步传输，然而，在依靠远程镜像解决方案的情况下，除同步数据传输外，其他任何方法均不能保持数据库所处的所有镜像卷的写入顺序。这意味着这种配置不能保证所有时间上的数据一致性，从而使得它们不适合作为 OLTP 数据的数据保护 / 灾难恢复解决方案。因为 Data Guard 仅将重做数

据传输到备用站点上（使用标准 IP 网络），并保持在所有保护模式下的事务一致性（即无论使用同步或异步传输模式），并且不需要昂贵的中间存储盒，所以它是一种更好的、适用于广域网的数据恢复和数据保护解决方案。

3．更强的数据保护和恢复能力

Data Guard 进程在它们从主数据库读和写信息时识别数据格式。此外，Oracle Data Guard 集成了闪回数据库功能，并且允许延迟应用程序的更改。这些功能防止了许多人为错误和数据损坏的传播与 / 或对备用数据库的影响。远程镜像没有这种优势，任何无意中对一个重要表的删除都将立即传播给数据库文件的远程副本，从而给该数据库文件的远程副本带来不利影响。

总之，Data Guard 是为现代信息系统提供的一个全面的数据保护、灾难恢复和高可用性解决方案。它提供了一个能够解决计划内和计划外服务器中断的灵活、易于管理的框架。物理备用数据库和逻辑备用数据库互相补充，并且可以同时进行维护，从而在减少主数据库上开销的同时提供高品质的数据保护。不同的数据保护模式提供了适用于各种保护、性能和基础架构需求的灵活性。工程应用中，无论是备用数据库置于隔壁机房，还是置于地球的任何一个角落，Data Guard 提供的数据保护和高可用性就会延伸到那里。

第2章

基于事务日志的数据恢复技术

2.1 事务处理与联机日志

事务是数据库执行数据处理的基本单位。在执行一个事务时，如果把该事务处理（Transaction Processing）从开始到结束所涉及的操作信息（包括事务涉及的数据库对象，以及在这个对象上执行的操作类别、操作前后的数据映像、事务的状态等信息）作为一个特殊的数据项以二进制的方式记录下来，这就是事务日志，可以说事务日志是记录数据库系统数据操作的“流水账”，目前绝大多数数据库系统都支持对事务日志的记录。

在这里简要介绍 Oracle 数据库的体系结构。一个 Oracle 数据库在物理上由 Data Files、Redo Log Files 和 Control Files 三部分组成，它们在逻辑上形成一个有机的整体。Data File 是由表空间（Tablespace）所对应的文件，用于实际存储数据库中的数据；Redo Log File 是用于记录事务处理信息的日志文件；Control File 存储整个数据库的结构信息，它控制整个数据库的运行。

Oracle 数据库使用联机日志文件来记录日志，为了处理无限增长的事务日志和有限日志文件大小之间的矛盾，Oracle 利用多个日志文件以循环的方式来记录日志，例如，若某数据库设置 3 个日志文件，首先用第一个文件来记录日志，第一个记满了后，Oralcle 会自动转移到第二个文件记录（该操作称为日志切换，Log Switch），以此类推，当第三个文件记录满后，又会切换到第一个文件记录日志，如此循环往复。

另外，为了提高日志文件的安全性，Oracle 提供了与控制文件类似的在线镜像 / 复用（Multiplexing）机制，上述 3 个日志文件构成了 3 个日志组（Log Group），每个日志组的日志文件又可以被镜像为多个日志文件，称为日志成员（Log Member）。任一时刻只有一个日志组处于“联机写”的状态，称为当前日志组；在同一个组下，不同的日志成员文件记录的日志内容完全一致，并且为了提高日志组的安全性，不同的日志成员被放置于不同的物理磁盘上，这样即使在某个磁盘出现故障的情况下，同一个日志组下的日志成员也不会全部丢失。因此，通过这种方式，可以显著提高日志文件（内容）的安全性。

2.2　归档日志与运行模式

数据库在启动时需要创建 Oracle Instance（指数据库使用的内存结构），包括 SGA 和 Background Processes 两部分。根据对数据库的配置和 Parameter File 中的参数设置，可以将数据库设置在归档方式（Archive Log Mode）下运行，这样每当出现 Log Switch 操作时，Background Processes 中的进程 ARCH 会将刚写满的 Redo Log File 备份转移，形成归档日志（Archive Log Files，即联机日志的脱机复制）。由于受物理文件大小的限制，联机日志只能记录有限时间段的事务处理信息，而联机日志和归档日志的联合可以连续完整地记录数据库在运行过程中的事务处理信息。

在不同的日志操作模式下，数据库备份和恢复的策略具有很大的差异。本节将以数据库日志操作模式的管理为中心，介绍以下内容：

※　数据库日志操作模式。

※　日志操作模式的设置方法。

※　归档模式的分类。

1. 非归档模式（Noarchivelog）

这种操作方式有如下特点：

※　日志文件以循环方式使用，发生日志切换时，原有的日志内容被覆盖，旧的日志文件中的事务日志记录不归档。

※　由于日志文件的内容被不断地覆盖，所以联机日志文件中只能记录最新的有限时间段的日志信息。

※　只能在数据库关闭时做数据库的操作系统级备份，必须备份所有的控制文件、数据文件和联机日志文件。

2. 归档模式（Archivelog）

Archivelog 方式有如下特点：

※　日志文件在检查点发生和被后台进程 ARCH 物理归档之前，不能被覆盖重用。

※　任何时候实例恢复都可使用最新的变更，而且已归档的事务日志被用于介质恢复。

在 Archivelog 方式下可实现如下功能：

※　数据库的事务日志得到完整保存，以防止数据库出现介质失败。

※　数据库能在打开时进行表空间备份。

※ 当非 SYSTEM 表空间由于介质失败而脱机时，数据库的剩余部分仍可使用，因为除脱机表空间之外的数据没有丢失。

※ 要求日志文件保证在它们需要再使用之前都能完成联机归档。

配置数据库使它在 ARCHIVELOG 模式下操作，可利用脱机或联机备份从介质失效中进行完全和不完全恢复。这种模式也是 Oracle 建议的日志操作模式。没有配置为 ARCHIVELOG 模式的数据库，在介质失效情况下虽然可从备份中恢复数据库，但是不能将数据库恢复到失败时的状态。

例如，如果数据库在 NOARCHIVELOG 模式下操作，且每周日晚做一次数据库的脱机完全备份，如果周五发生了介质失效，此时唯一可做到的是进行不完全恢复，即使用周日晚的备份恢复数据库并重启数据库。但是，如果数据库在 ARCHIVELOG 模式下操作，则可以实现完全恢复，即将数据库恢复到周五数据库失败时的状态。

用户可根据如下原则来选择数据库的日志操作模式：

※ 当出现磁盘故障而破坏数据库中的数据时，如果允许丢失一部分数据，可选择 NOARCHIVELOG 方式。

※ 若不允许丢失数据，则应选择 ARCHIVELOG 方式。

※ 当要求数据库全时（即一天 24 小时，一周 7 天）都在运行时，应选用 ARCHIVELOG 方式。

※ 对于分布式数据库，各个节点都应具有相同的归档方式，以保证实现协调分布式数据库恢复。如果各节点操作方式不一致，则全局分布数据库恢复（使所有数据一致）会受在 NOARCHIVELOG 方式下操作的数据库的恢复能力的限制。用户可根据自己的数据库的特点来选择相应的数据库日志操作模式。

2.3 事务日志用于数据恢复的核心机制

也许有人要问，使用数据库系统我们关心的是“数据”，为什么还要去研究事务日志呢？要弄清这个问题，需要我们动态地考察数据和日志之间的内在联系。

事务日志是以事务为单位进行记录的，每一个日志项 (Log Item/Entity) 详细记录了事务处理的足够信息，以确保系统在必要的时候能够重演（Redo）该事务。如果说数据库中的 Data 是直接数据或“一次数据”，那么数据库中的 Log 则是间接数据或“二次数据”，它记录着 Data 的变更信息。从另一个角度看，Data 反映的是事务处理操作的结果，而 Log 记录的则是事务处理操作的过程，有了这样的历史变更过程，在一定条件下就可以依次“重演”这个过程（即重新执行事务处理的操作），以达到恢复数据的目的。从这个意义上说，事务日志是系统为了“数据”的安全而事先投注的“保险”。

假设某运行中的数据库系统在 T1 时刻做了一次完整的备份，由于某种原因在

T2 时刻（T2>T1）出现了故障，导致数据库损坏，如图 2-1 所示。一般意义上的数据恢复只能做到不完全恢复（Incomplete Recovery），即重建数据库，利用 T1 时刻的数据备份将数据库恢复到 T1 时刻的状态，T1 至 T2 时间段的数据则丢失。如果 T1 至 T2 时间段的事务日志存在，则可以实现数据的完全恢复（Complete Recovery），即将数据库恢复到故障前的状态。因为 T1 至 T2 时间段的事务日志是按照时间顺序详细记录了这一时间段的事务处理信息，之所以数据库中的数据由 T1 状态过渡到 T2 时刻的状态，正是由于这些事务处理操作的结果。如果在 T1 状态的基础上，重新依次运行这些事务，就可以将数据库由 T1 状态逐步推演到 T2 时刻的状态，实现数据库的数据恢复，并且这种“推演”可以根据需要在 T1 与 T2 之间某个时间点停留，即将数据库恢复到过去的某个特定的时间点。图 2-1 对整个过程作了演示。

需要注意的是，日志中记录的事务处理信息是与当时的数据库状态相关联的，离开了特定的数据状态，数据操作也就失去了意义。因此，事务日志的重演是有前提的：一是数据库首先必须处于某一时刻点的完整状态（通常由数据库备份来实现）。只有处于这个状态，在这一时间点之后的事务才有重新运行的基础；二是重新运行的事务处理必须严格按照原来的时间顺序进行，只有这样，才能实现逐步推演，因为实际的数据库操作其后一步操作总是依赖于前一步操作的结果。正因为如此，数据库的事务日志必须连续地保存，如果某个时间段的事务日志损坏或丢失，则此时间段之后的日志也就失去了存在的意义。

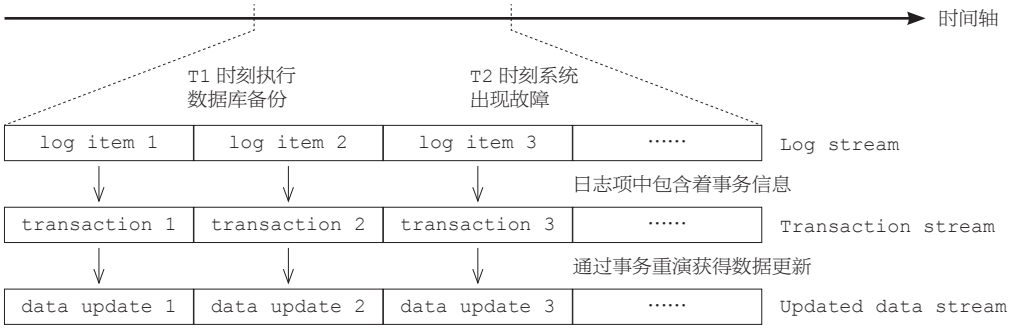


图 2-1　基于事务处理的数据恢复过程

2.4　Oracle 数据库介质恢复框架

无论何种原因导致 Oracle 数据库不能正常启动（即不能进入 Open 状态），此时即需要执行数据库恢复（Recovery）。典型的数据库恢复过程需要如下三个阶段：① Restore：选择某个历史备份作为恢复的起点，即首先将数据恢复至备份时刻的状态；② Roll Forward：利用归档日志和联机日志依次重做自备份时刻以来的事务；③ Roll Back：在故障时刻前附近的一些事务，有些还未来得及提交（Commit），但由于系统内部的 Checkpoint 事件的触发导致已经写入数据文件，

这部分事务需要利用数据映像（Before Image）进行必要的回滚。如果这三个阶段的操作都能够顺利进行，则可以将数据库毫无损失地恢复到损坏前一时刻的状态，即所谓的完全数据库恢复（Complete Recovery），如果这个恢复过程在第二、三阶段由于某种原因中途结束，则数据库只能恢复到过去的某个时间点，即不完全恢复（Incomplete Recovery）。图 2-2 给出了 Oracle 数据库恢复的典型示例。

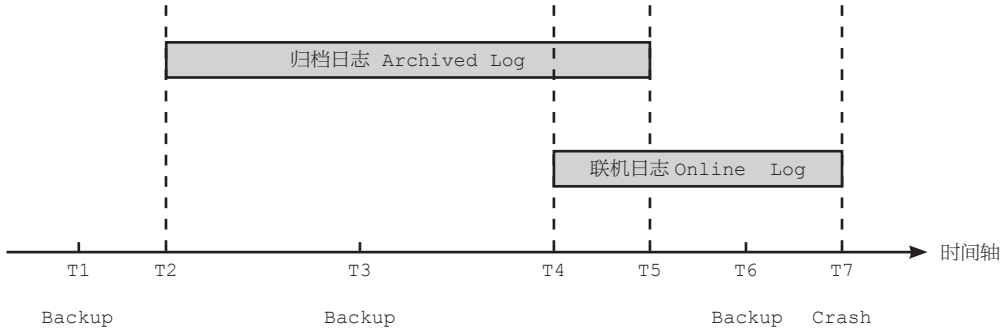


图 2-2 典型 Oracle 数据库恢复示例

设系统在 T7 时刻出现故障导致 Oracle 数据库停止运行，数据库分别在 T1、T3、T6 有三次历史备份，由图示可知，Archive Log 中包含了 T2 至 T5 时间段的事务日志，Online Redo Log 中包含了 T4 至 T7 时间段的事务日志。下面说明恢复过程。

最直接的恢复方法是选择 T6 时刻的备份作为恢复的起点，首先利用备份将数据库 Restore 到 T6 时刻，在此基础上只需利用联机重做日志，依次重新运行 T6 至 T7 时间段所包含的所有事务，即可将数据库恢复到故障时刻，实现 Complete Recovery。如果选择 T3 时刻的备份作为恢复的起点，则需要同时使用两类日志，首先利用归档日志重做 T3 至 T4（或 T5）时间段的事务，接着利用联机日志重做 T4（或 T5）至 T7 时间段的事务，也可将数据库恢复到故障时刻，同样实现 Complete Recovery。

若数据库在 T2 至 T7 时间段没有备份可以利用，则只能选择较远的 T1 时刻的备份作为恢复的起点，此时由于缺乏 T1 至 T2 时间段的事务日志，数据库只能恢复到 T1 时刻，导致 T1 至 T7（故障时刻）时间段的最新数据丢失，即只能实现 Incomplete Recovery；同样在选择 T3 时刻备份作起点的恢复方法中，如果联机重做日志丢失，则数据库只能最大限度地恢复到 T5 时刻，此时同样实现的是 Incomplete Recovery。

需要特别指出的是，尽管在上述恢复过程中没有提到控制文件，但它的作用不可或缺，系统恢复进程正是根据 Control File 中包含的最新的 Log Sequence Number 和 SCN 确定需要重做的事务。由此需要注意，在利用历史备份对数据库进行 Restore 阶段，尽量不要对控制文件进行 Restore 操作，否则可能会导致数据

库的不完全恢复；其次，为确保控制文件的可用性，控制文件最好联机保存多个副本，并分布于不同的物理磁盘上。

2.5　完全数据恢复的黄金法则

俗话说，“养兵千日，用兵一时”，要在故障后能够实现数据的完全恢复，事先就要有所“准备”。我们在大量工程实践的基础上总结出处理数据库日志系统的黄金法则，它和当今的证券投资思想有异曲同工之处，通俗地说就是“不要把所有的鸡蛋放在同一个篮子里”。

1．隔离数据和事务日志的物理存储

数据和事务日志是数据库两个不可分割的部分。实际应用中，在部署一个数据库系统的物理存储时，有必要将数据和事务日志分开，条件许可的情况下，尽可能地将两者置于相互独立的物理磁盘中。这种存储方案可以大大提高数据的可用性：在数据库故障的案例中，存储介质失效占有相当大的比重，两个相互独立的物理磁盘同时出现介质故障的可能性要小得多。如果数据盘出现损坏，而日志盘正常，导致数据丢失而日志存在，这种情况可以通过历史备份和连续存在的事务日志使数据库得到完全恢复。

2．保持事务日志适当冗余（Redundancy）

在前面介质失效的例子中，如果出现另一种情况，即日志盘损坏而数据盘正常，导致日志丢失而“数据”存在，这种情况下存在的“数据”通常也是不可存取的，因为无法定位日志，数据库是不能正常启动的。解决的办法是重建日志并做日志与数据的同步处理或利用备份进行恢复。实践中两种方法都不可避免地要出现部分最新数据的丢失。

比较前后介质失效的两种情况我们可以看出，事务日志在数据恢复过程中显得比数据本身更为重要。因此，在条件许可的情况下，应首先将事务日志进行数据冗余处理，如进行软硬件镜像、采用 RAID 存储、借助第三方软件实现热备等措施，以确保事务日志的安全性。因为数据丢失，我们可以利用历史备份和事务日志来恢复数据，而联机日志丢失则必然要导致数据的不完全恢复。

3．监控事务日志的完整性

数据库的事务日志是随时间一维无限增长的，实际投运的数据库需要保持对事务日志的监控，主要是两个方面，一是维护事务日志的时间连续性和完整性，以确保在数据库在出现故障的情况下数据能够实现完全恢复。这里说的连续性和完整性是指自上一次数据库完整备份以来至数据库运行的当前时刻这一时间段的事务日志。对 Oracle 来说，必须将数据库设置在归档方式（Archive Log Mode）下运行。二是监视事务日志的存储空间，如果对它不作任何处理，事务日志会持续增长，一

旦存储空间用尽，数据库进程将被挂起，此时的数据库将会停止响应对数据的任何操作。

业务数据是任何信息系统的核心，由于实际系统的复杂性（如系统硬件或软件缺陷、存储介质失效、应用软件错误、病毒等导致故障），绝对可靠的系统是不存在的，一旦数据系统出现故障，我们总是希望业务数据能够得到有效的恢复。通过上述讨论和对 Oracle 的典型案例分析，使我们重新认识了数据库日志系统的价值。第一，在规划和部署数据库系统时，重视对日志系统的设计可以显著提高数据系统的可用性；第二，一旦数据库系统出现故障，事务日志是一种强有力的数据再生手段，并且可以实现对数据库故障后的完全恢复（Complete Recovery），保障数据的高可用性，这一特性在一些关键业务系统中具有极高的应用价值。

Dataguard 正是 Oracle 基于事务日志的价值开发出的一类数据保护与容灾方案。

第3章

重做数据与日志挖掘

数据库在运行过程中，用户执行的 DML、DDL 和 DCL 语句带来的任何数据变更（包括用户数据和数据字典的变更）会生成重做数据（Redo Data），数据库将其记入事务日志文件（包括联机日志文件和归档日志文件）。反过来，通过对日志文件内容的分析处理，可以挖掘出用户曾经执行的 SQL 指令，这就是日志挖掘（Log Mining）。本章介绍 Oracle 数据库关于日志挖掘方面的主要内容，包括日志挖掘接口 LogMiner、相关的 PL/SQL 程序包、数据字典视图等。Oracle 的日志挖掘在工程中有多种用途，如 Data Guard 的逻辑备用数据库就是日志挖掘的一项主要应用。

3.1 Oracle 日志挖掘介绍

Oracle 数据库系统提供专门的日志挖掘接口 LogMiner，通过该接口，可以在本地或远程的 Oracle 数据库中执行手工或自动的日志挖掘处理，以实现多种应用目标。

3.1.1 LogMiner 的配置框架

在执行日志挖掘之前，需要对 LogMiner 进行配置，完整的配置涉及 4 个因素：源数据库（Source Database）、挖掘数据库（Mining Database）、日志挖掘字典（LogMiner Dictionary）和需要分析挖掘的重做日志文件列表（Redo Log Files）。

※ 源数据库：产生重做数据的数据库。

※ 挖掘数据库：执行对日志文件进行分析处理的数据库。

※ LogMiner 字典：重做数据中涉及的数据库对象、数据类型都是以内部对象 ID（Internal Object Identifier）形式表示。LogMiner 利用该字典将其转换为有意义的名称（如表名、列名等）。

※ 日志文件列表：这些文件中包含我们需要对其进行分析处理的重做数据。

图 3-1 是我们在 Data Guard 中逻辑备用数据库方案中需要用到的 LogMiner 的典型配置。主数据库中的重做数据被源源不断地通过网络传输到逻辑备用数据库。

图 3-1 Data Guard 中的 LogMiner 配置

在图 3-1 所示的典型配置中，Data Guard 中的主数据库是 LogMiner 的源数据库，逻辑备用数据库则是 LogMiner 的挖掘数据库，两者通过 Oracle Net 网络连接起来，逻辑备用数据库接收来自主数据库的重做数据，并实现自动化的日志挖掘与应用。

当然，LogMiner 源数据库和挖掘数据库可以是同一数据库。当 LogMiner 源数据库和挖掘数据库是分离的两个不同的数据库时，对数据库及其运行平台有如下要求（该要求也是在 Data Guard 中实现逻辑备用数据库方案的要求）：

（1）源数据库和挖掘数据库必须运行在相同的硬件平台上。

（2）挖掘数据库及其 DBMS 的版本必须高于或等于源数据库的版本。

（3）挖掘数据库应使用与源数据库相同的字符集。

（4）挖掘数据库接收的重做数据必须来源于同一 Incarnation 的源数据库（Associated with the same database RESETLOGS SCN）。

（5）建议在源数据库中启动强制日志记录（Force Logging）和补充日志记录（Supplemental Logging）。

LogMiner 日志挖掘的过程可以手工完成或自动完成。典型的情况是当挖掘数据库和源数据库是同一数据库的情况，往往在必要的时候手工执行日志挖掘；当挖掘数据库和源数据库分立于两个数据库时（像 Data Guard 中的逻辑备用数据库方案），往往配置 Oracle 的日志挖掘与应用通过一组相关进程自动完成。

3.1.2 LogMiner 的用户接口

Oracle 提供的 LogMiner 用户接口主要由 PL/SQL 程序包和数据字典视图构成，其主要内容有：包 DBMS_LOGMNR、包 DBMS_LOGMNR_D、视图 V$LOGMNR_

CONTENTS、视图 V$LOGMNR_LOGS 等。

※ 包 DBMS_LOGMNR：用于配置 LogMiner 并启动和停止 LogMiner 的 PL/SQL 包。

※ 包 DBMS_LOGMNR_D：主要用于在源数据库中构建 LogMiner 字典。

※ 视图 V$LOGMNR_LOGS：用于查询当前 LogMiner 日志挖掘的重做日志文件列表。

※ 视图 V$LOGMNR_CONTENTS：用于查询当前 LogMiner 日志挖掘的处理结果，包括事务处理的对象、对应的 SQL 语句等。

典型的 LogMiner 日志挖掘过程如下：

（1）选择或创建用于日志挖掘的 LogMiner 字典。LogMiner 字典可以位于源数据库的数据字典中、也可存在于源数据库的日志流中，还可存在于数据库之外的 OS 文件中。

过程 DBMS_LOGMNR_D.BUILD 专门用于在源数据库中创建 LogMiner 字典。

（2）指定需要分析处理的日志文件列表。这里的日志文件列表用于有目的地选取部分的、连续的重做数据对其执行挖掘处理。

过程 DBMS_LOGMNR.ADD_LOGFILE 用于创建和添加这里的日志文件列表。

（3）启动 LogMiner 日志挖掘。过程 DBMS_LOGMNR.START_LOGMNR 用于手工启动日志挖掘。在自动挖掘处理中，通过在挖掘数据库中启动 SQL Apply 应用服务实现日志挖掘。

（4）查看 LogMiner 日志挖掘的处理结果。日志挖掘的处理结果可通过视图 V$LOGMNR_CONTENTS 查看。注意，只能在启动 LogMiner 日志挖掘会话期间查询该视图，当停止 LogMiner 日志挖掘会话时，该视图被清空。

（5）终止 LogMiner 日志挖掘会话。

3.1.3　补充日志 Supplemental Logging

Oracle 日志文件记录的重做数据是为了满足实例恢复和介质恢复的用途，凡是对这两类恢复需要的重做数据都会被记入日志文件。然而，当需要基于重做数据开展日志挖掘，发掘其中的 DML、DDL 和 DCL 语句时，Oracle 需要在重做数据中记录额外的信息，这就是补充日志（Supplemental Logging）。

默认情况下，Oracle 数据库并没有启动任何补充日志记录。在决定使用 LogMiner 对数据库的日志文件进行日志挖掘并开展应用（如建立逻辑备用数据库）前，必须至少启动最小的补充日志记录（Minimal Supplemental Logging）。

在实例恢复和介质恢复中，通过 ROWID 去更新记录是足够的。正像 Data Guard 中的逻辑备用数据库一样，如果通过日志挖掘出的 SQL 应用到另外一个数据库中去重构数据更新时，就不能仅根据 ROWID 去确定需要更新的行，因为一个数据库中的 ROWID 对另外一个数据库毫无意义。此时日志中的重做数据就要记录更多的字段信息以确保挖掘出的 SQL 语句在另外一个数据库中能够唯一地定位到修改的记录。这就是为什么 LogMiner 日志挖据必须启动补充日志记录的内在原因。

除 ROWID 之外，如果不能标识更新的记录，Oracle 就会通过记录附加的字段值来标识记录，这就是补充日志记录的内容。如果能够通过主键（Primary Key）或唯一性约束（Unique Constraint）来标识更新的记录，这就是 Primary 或 Unique 补充日志。如果没有主键或唯一键约束可以利用，Oracle 就会记录除 LONG 和 LOB 字段外的所有字段值来标识记录，这就是 ALL 补充日志。

1．数据库级的补充日志记录

启动最小补充日志记录：

```
SQL> alter database add supplemental log data;
```

启动主键和唯一键补充日志记录：

```
SQL> alter database add supplemental log data (primary key)
columns;
SQL> alter database add supplemental log data (unique)
columns;
```

启动全补充日志记录：

```
SQL> alter database add supplemental log data (all) columns;
```

取消已经启动的补充日志记录：

```
SQL> alter database drop supplemental log data ... ;
```

2．表级补充日志记录

```
SQL> alter table ... add supplemental log data (primary key)
columns;
SQL> alter table ... add supplemental log data (unique)
columns;
SQL> alter table ... add supplemental log data (all) columns;
```

3.2　确定 LogMiner 字典

在执行具体的日志挖掘之前，确定 LogMiner 字典的存在及其使用选项至关重要，因为它会帮助 LogMiner 给出有意义的处理结果。本质上 LogMiner 字典存在于数据库的数据字典（Database Dictionary）中。

Oracle 为 LogMiner 字典提供如下 3 种使用选项：

※ 使用在线目录（Online Catalog）：即使用数据库字典中的 LogMiner 字典。这是一种 LogMiner 字典最有效的、最方便的使用方式，但前提条件是挖掘数据库必须和源数据库是同一数据库，且在日志挖掘过程中源数据库必须处于 Open 状态。

※ 提取 LogMiner 字典至日志流：将数据库数据字典中的 LogMiner 字典提取至某个或某些归档日志文件中。当挖据数据库与源数据库不是同一数据库时，Oracle 建议使用这一选项。

※ 提取 LogMiner 字典至 OS 文件：将数据库数据字典中的 LogMiner 字典提取至数据库之外的操作系统文件。

比较三种使用方式，以第一种方式使用 LogMiner 字典，该字典内容是动态的，它总是反映数据库最新的对象信息；后两种方式提取的 LogMiner 字典则是静态的，它仅反映提取时的信息。不过当使用提取 LogMiner 字典至日志流的方式时，在启动日志挖掘时使用选项 DDL_DICT_TRACKING 可保持对数据库中 LogMiner 字典信息的跟踪，使日志挖掘总是使用最新的 LogMiner 字典。

1．使用在线目录方式

此方式无须单独创建 LogMiner 字典，只需在启动日志挖掘时设置 START_LOGMNR 过程的 OPTIONS 参数为 DICT_FROM_ONLINE_CATALOG 即可。

```
SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR(-
OPTIONS => DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG);
```

2．使用提取 LogMiner 字典至日志流方式

此方式将 LogMiner 字典提取至归档日志文件中。此方式要求在启动日志挖掘之前必须事先使用过程 DBMS_LOGMNR_D.BUILD 创建 LogMiner 字典。

```
SQL> EXECUTE DBMS_LOGMNR_D.BUILD( -
OPTIONS=> DBMS_LOGMNR_D.STORE_IN_REDO_LOGS);
```

要查看 LogMiner 字典到底存储在哪个或哪些归档日志中，可查询 V$ARCHIVED_LOG 视图，通过 DICTIONARY_BEGIN 字段和 DICTIONARY_END 字段确定 LogMiner 字典存在的归档日志范围。

```
SQL> SELECT NAME fisrst_archived_logfile FROM V$ARCHIVED_LOG
WHERE DICTIONARY_BEGIN='YES';
SQL> SELECT NAME last_archived_logfile FROM V$ARCHIVED_LOG
WHERE DICTIONARY_END='YES';
```

在手工启动日志挖掘操作前，需要建立被挖掘的日志文件列表，使用此种方式需要将上述查询确定的所有归档日志文件包含在日志文件列表中。

3．使用提取 LogMiner 字典至 OS 文件方式

此方式以数据库外部文件的形式存储日志挖掘需要的 LogMiner 字典。使用过程 DBMS_LOGMNR_D.BUILD 提取 LogMiner 字典至外部文件，必须将 OPTIONS 参数设置为 STORE_IN_FLAT_FILE 值。

另外，需要注意，这种方式提取 LogMiner 字典的过程中要确保数据库没有任何 DDL 操作。步骤如下：

（1）设置初始化参数 UTL_FILE_DIR。该参数需要指向 LogMiner 字典文件的存储目录，目录需要事先存在，参数需要生效。例如：UTL_FILE_DIR = /oracle/database

如果在调用 DBMS_LOGMNR_D.BUILD 过程创建 LogMiner 字典文件之前没有设置该参数或参数没有生效，都会遭遇 ora-01308 错误。

```
ora-01308 Initialization parameter UTL_FILE_DIR is not set.
```

（2）调用 DBMS_LOGMNR_D.BUILD 过程。提供如下参数：

※ Directory_file，LogMiner 字典文件名。

※ Directory_location，参数 UTL_FILE_DIR 指向的目录。

※ Options，设置为过程常量 DBMS_LOGMNR_D.STORE_IN_FLAT_FILE。

```
SQL> EXECUTE DBMS_LOGMNR_D.BUILD('logmnr_dict.ora', -
'/oracle/database/', DBMS_LOGMNR_D.STORE_IN_FLAT_FILE);
```

当调用过程 DBMS_LOGMNR_D.BUILD 指定了文件名参数和存储目录参数后，常量值 DBMS_LOGMNR_D.STORE_IN_FLAT_FILE 为 options 参数默认选项。

3.3 建立日志文件列表

每当启动 LogMiner 执行日志挖掘时，Oracle 需要确定哪些归档日志文件需要被分析处理，这就是待挖掘的日志文件列表。

这个日志文件列表的形成有两种方式：动态的和静态的。

1．动态确定日志文件列表

此方式让 Oracle 根据过程 DBMS_LOGMNR.START_LOGMNR 的相关参数（如时间范围、SCN 范围等）在挖掘过程中动态确定需要分析处理的日志文件，此时需要指定过程 DBMS_LOGMNR.START_LOGMNR 的参数 OPTIONS 需要包含 CONTINUOUS_MINE 选项。

下面的例子通过指定具体的时间范围为 2003-01-01 日的上午 8:30 至 8:45，日志文件列表为此时间段产生的日志，LogMiner 字典使用在线目录方式。

```
SQL> alter session set NLS_DATE_FORMAT = 'YYYY-MM-DD
HH24:MI:SS';
SQL> exec DBMS_LOGMNR.START_LOGMNR( -
STARTTIME => '2003-01-01 08:30:00', -
ENDTIME => '2003-01-01 08:45:00', -
OPTIONS => DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG -
+ DBMS_LOGMNR.CONTINUOUS_MINE);
```

注意，此方式要求挖掘过程是在源数据库中完成，并且要求数据库至少处于 mount 状态，因为 Oracle 动态确定日志范围时需要访问控制文件。

2．静态确定日志文件列表

此方式是在启动 LogMiner 日志挖掘之前使用过程 DBMS_LOGMNR.ADD_LOGFILE 手工建立日志文件列表。

使用过程 DBMS_LOGMNR.ADD_LOGFILE 新建一个日志文件列表时，第一个日志文件的加入需要将参数 options 选项设置为 DBMS_LOGMNR.NEW 常量值。当第一个日志文件被加入一个日志文件列表，后续的日志文件都必须来源于同一数据库（并且要求同一 Incarnation）。

下面的例子演示手工建立一个日志文件列表的方法，示例中第一次之后的所有过程调用 ADD_LOGFILE 的参数 options 可选。

```
SQL> exec DBMS_LOGMNR.ADD_LOGFILE( -
LOGFILENAME => '/db_fra/netdb/O1_MF_1_10_8LGQG8PG_.ARC', -
OPTIONS => DBMS_LOGMNR.NEW);
SQL> exec DBMS_LOGMNR.ADD_LOGFILE( -
LOGFILENAME => '/db_fra/netdb/O1_MF_1_11_8LGVQYHN_.ARC', -
OPTIONS => DBMS_LOGMNR.ADDFILE);
...
```

查询视图 V$LOGMNR_LOGS 可以查看在当前 LogMiner 会话中日志文件列表中指定了哪些日志文件。

前面描述的两种建立日志文件列表的方式可以结合起来使用。

（1）使用过程 DBMS_LOGMNR.ADD_LOGFILE 为日志文件列表仅加入第一个日志文件。

（2）之后启动 LogMiner 时指定 CONTINUOUS_MINE 选项。

3.4 启动 LogMiner 日志挖掘

在确定了 Logminer 字典的使用方式、建立了待挖掘的日志文件列表后，我们可以调用过程 DBMS_LOGMNR.START_LOGMNR 来启动日志挖掘处理，该过程的参数如下：

```
STARTSCN                       NUMBER
ENDSCN                         NUMBER
STARTTIME                      DATE
ENDTIME                        DATE
DICTFILENAME                   VARCHAR2
OPTIONS                        BINARY_INTEGER
```

参数 STARTSCN、ENDSCN、STARTTIME、ENDTIME 用来在日志文件列表中过滤需要挖掘的日志范围，或用来动态确定需要挖掘哪些日志文件；不指定这组参数时则挖掘内容为日志文件列表的整个文件；参数 DICTFILENAME 用来指定 LogMiner 字典的 OS 文件，通过过程 DBMS_LOGMNR.START_LOGMNR 的 Options 选项指定 Logminer 字典使用方式为 DICT_FROM_ONLINE_CATALOG 和 DICT_FROM_REDO_LOGS 时无须指定该参数。

参数 OPTIONS 可指定一系列 DBMS_LOGMNR 包常量，通过该参数的设置，可以控制 LogMiner 的行为和输出。这些常量大致可以分为三类，一类是设置 LogMiner 字典的，一类是控制日志挖掘行为的，还有一类是控制挖掘输出结果的。

设置 LogMiner 字典的常量如下：

※ DICT_FROM_ONLINE_CATALOG。

※ DICT_FROM_REDO_LOGS。

※ DDL_DICT_TRACKING。

控制日志挖掘行为的常量如下：

※ CONTINUOUS_MINE。

※ COMMITTED_DATA_ONLY。

※ SKIP_CORRUPTION。

控制挖掘输出结果的常量如下：

※ PRINT_PRETTY_SQL。

※ NO_SQL_DELIMITER。

※ NO_ROWID_IN_STMT。

※ STRING_LITERALS_IN_STMT。

关于这些常量的详细含义请读者参考 Oracle 对 PL/SQL 包 DBMS_LOGMNR 的使用说明，在此不再赘述。

3.5　查看日志挖掘的结果

通过访问视图 V$LOGMNR_CONTENTS 可以获得 LogMiner 日志挖掘的结果，该视图提供被挖掘日志持续期间数据库的活动，包括如下细节：

※ 导致数据库数据变更的操作类型：INSERT、UPDATE、DELETE 或者 DDL（对应 OPERATION 字段）。

※ 数据变更对应的 SCN（对应 SCN 字段）。

※ 事务提交的 SCN（对应 COMMIT_SCN 字段）、事务标识（对应 XIDUSN、XIDSLT、和 XIDSQN 字段）等。

※ 事务操作涉及的数据库对象及其属主（对应 SEG_NAME 和 SEG_OWNER 字段）。

※ 数据变更操作的数据库用户（对应 USERNAME 字段）。

※ 如果数据变更是 DML 操作，则给出对应的等价的 DML SQL；如果操作是 DDL，则给出与用户执行完全相同的 DDL 语句（对应字段 SQL_REDO ）。

※ 如果口令是 SQL 语句的一部分，则 SQL_REDO 字段给出加密的形式。

※ 如果数据变更是 DML 操作，LogMiner 还给出对应的撤销改变的语句（对应 SQL_UNDO 字段）。对于 DDL 操作，SQL_UNDO 字段总是 NULL；对于某些 LogMiner 不支持的数据类型和回滚操作，SQL_UNDO 字段也是给出 NULL 值。

查询实例：假设经过日志挖掘后，DBA 想了解 Scott 用户在 oe.orders 表上执行的删除操作，可以执行如下 SELECT 语句：

```
SYS@NETDB>SELECT OPERATION, SQL_REDO, SQL_UNDO
  2  FROM V$LOGMNR_CONTENTS
  3  WHERE SEG_OWNER = 'OE' AND SEG_NAME = 'ORDERS' AND
  4  OPERATION = 'DELETE' AND USERNAME = 'SCOTT';
```

上面的查询会给出类似如下的结果：

```
    OPERATION SQL_REDO SQL_UNDO
-----------------------------------------------------------------

DELETE      delete from "OE"."ORDERS"       insert into "OE"."ORDERS"
            where "ORDER_ID" = '2413'       ("ORDER_ID","ORDER_MODE",
            and "ORDER_MODE" = 'direct'      "CUSTOMER_ID","ORDER_STATUS",
            and "CUSTOMER_ID" = '101'        "ORDER_TOTAL","SALES_REP_ID",
            and "ORDER_STATUS" = '5'         "PROMOTION_ID")
            and "ORDER_TOTAL" = '48552'      values ('2413','direct','101',
            and "SALES_REP_ID" = '161'       '5','48552','161',NULL);
            and "PROMOTION_ID" IS NULL
            and ROWID = 'AAAHTCAABAAAZAPAAN';

DELETE      delete from "OE"."ORDERS"        insert into "OE"."ORDERS"
            where "ORDER_ID" = '2430'        ("ORDER_ID","ORDER_MODE",
            and "ORDER_MODE" = 'direct'       "CUSTOMER_ID","ORDER_STATUS",
            and "CUSTOMER_ID" = '101'         "ORDER_TOTAL","SALES_REP_ID",
            and "ORDER_STATUS" = '8'          "PROMOTION_ID")
            and "ORDER_TOTAL" = '29669.9'     values('2430','direct','101',
            and "SALES_REP_ID" = '159'        '8','29669.9','159',NULL);
            and "PROMOTION_ID" IS NULL
            and ROWID = 'AAAHTCAABAAAZAPAAe';
```

从上面的查询中可以看出，Scott 用户执行了删除操作，结果两条记录被删除。值得注意的是，这里 LogMiner 给出的 SQL_REDO 挖掘结果与当时 Scott 用户执行的 Delete 语句可能不同，因为一条 Delete 语句可以删除多条记录。Scott 用户原始执行的 Delete 操作也许是 delete from oe.orders where customer_id ='101'; 也许是 delete from oe.orders where promotion_id=NULL; 但达到的效果都是 order_id 为 2413 和 2430 两条记录被删除。这也就是为什么 LogMiner 给出的 SQL_REDO 结果不能和用户当初执行的语句完全相同的原因。

这里要注意，只有在 LogMiner 会话持续期间才能通过该视图查询日志挖掘的结果，否则，会产生 ORA-01306 错误：

```
ORA-01306: dbms_logmnr.start_logmnr() must be invoked before
selecting from v$logmnr_contents.
```

产生 ORA-01306 错误背后的原因是 Oracle 对 V$LOGMNR_CONTENTS 视图查询的响应方式，该视图有别于数据字典的其他视图。该视图实际上是日志文件中的重做数据的关系型表达。当 DBA 执行对 V$LOGMNR_CONTENTS 视图的查询时，挖掘会话有序地读取被挖掘的日志文件中的重做数据，并将其转换为 V$LOGMNR_

CONTENTS 视图中的记录，直到遍历完日志文件列表中的内容。从这个角度看，调用过程 DBMS_LOGMNR.START_LOGMNR 和 DBMS_LOGMNR.END_LOGMNR 只是设置 Oracle 是否允许转换日志中的重做数据的开关而已，实际的挖掘过程是由查询 V$LOGMNR_CONTENTS 视图背后的操作完成的。

Oracle 数据库本身并不存储 V$LOGMNR_CONTENTS 视图中的内容，如果 DBA 需要保存留作进一步的分析利用，应该将其查询结果保存在自己建立的表中。

3.6　控制事务输出

默认情况下，LogMiner 以日志流中记录的数据操作顺序输出挖掘结果（操作记录），包括已经回滚的事务、还未提交的事务（执行过程中），以及数据库内部的操作（如用于数据字典更新的操作）。

在调用 DBMS_LOGMNR.START_LOGMNR 过程时设置 OPTIONS 选项参数指定 COMMITTED_DATA_ONLY 可以过滤掉这些操作记录，并且使得 LogMiner 按已提交事务对挖掘出的记录进行分组，并按照事务提交的先后顺序输出结果。

指定

```
SQL> exec DBMS_LOGMNR.START_LOGMNR(-
OPTIONS => DBMS_LOGMNR.COMMITTED_DATA_ONLY);
```

启动日志挖掘时指定 COMMITTED_DATA_ONLY 选项，LogMiner 会在内存中按事务累积挖掘出所有操作，直到遇到事务提交记录才会产生输出，这会导致 LogMiner 在日志挖掘过程中消耗更多的内存。

假设在没有指定 COMMITTED_DATA_ONLY 选项时，执行对视图 V$LOGMNR_CONTENTS 查询 LogMiner 的输出结果如下：

```
SQL> SELECT (XIDUSN || '.' || XIDSLT || '.' || XIDSQN) AS XID,
USERNAME, SQL_REDO FROM V$LOGMNR_CONTENTS WHERE USERNAME !=
'SYS'
AND SEG_OWNER IS NULL OR SEG_OWNER NOT IN ('SYS', 'SYSTEM');

XID USERNAME SQL_REDO
----------------------------------------------------------------
```

```
1.15.3045   RON       set transaction read write;
1.15.3045   RON       insert into "HR"."JOBS"("JOB_ID","JOB_TITLE",
                      "MIN_SALARY",'MAX_SALARY") values ('9782',
                      'HR_ENTRY',NULL,NULL);
1.18.3046   JANE      set transaction read write;
1.18.3046   JANE      insert into "OE"."CUSTOMERS"("CUSTOMER_ID",
                      "CUST_FIRST_NAME","CUST_LAST_NAME",
                      "CUST_ADDRESS",'PHONE_NUMBERS","NLS_LANGUAGE",
                      "NLS_TERRITORY","CREDIT_LIMIT","CUST_EMAIL",
                      "ACCOUNT_MGR_ID") values ('9839','Edgar',
                      'Cummings',NULL,NULL,NULL,NULL,
                       NULL,NULL,NULL);
1.9.3041    RAJIV      set transaction read write;
1.9.3041    RAJIV      insert into 'OE"."CUSTOMERS'('CUSTOMER_ID",
                       "CUST_FIRST_NAME","CUST_LAST_NAME",'CUST_ADDRESS',
                       "PHONE_NUMBERS","NLS_LANGUAGE","NLS_TERRITORY',
                       "CREDIT_LIMIT","CUST_EMAIL",'ACCOUNT_MGR_ID')
                       values ('9499','Rodney','Emerson',NULL,NULL,NULL,NULL,
                       NULL,NULL,NULL);
1.15.3045    RON       commit;
1.8.3054     RON       set transaction read write;
1.8.3054     RON       insert into "HR"."JOBS'('JOB_ID","JOB_TITLE',
                       "MIN_SALARY",'MAX_SALARY') values ('9566',
                       'FI_ENTRY',NULL,NULL);
1.18.3046    JANE      commit;
1.11.3047    JANE      set transaction read write;
1.11.3047    JANE      insert into "OE"."CUSTOMERS"("CUSTOMER_ID",
                       "CUST_FIRST_NAME","CUST_LAST_NAME",
                       "CUST_ADDRESS","PHONE_NUMBERS","NLS_LANGUAGE",
                       "NLS_TERRITORY","CREDIT_LIMIT","CUST_EMAIL",
                       "ACCOUNT_MGR_ID") values ('8933','Ronald',
                       'Frost',NULL,NULL,NULL,NULL,NULL,NULL,NULL);
1.11.3047    JANE      commit;
1.8.3054     RON       commit;
```

当指定 COMMITTED_DATA_ONLY 选项后，输出结果会变为如下形式：

```
        XID USERNAME SQL_REDO
----------------------------------------------------------------
```

```
1.15.3045   RON       set transaction read write;
1.15.3045   RON       insert into "HR"."JOBS"("JOB_ID","JOB_TITLE",
                      "MIN_SALARY","MAX_SALARY") values ('9782',
                      'HR_ENTRY',NULL,NULL);
1.15.3045    RON       commit;
1.18.3046   JANE      set transaction read write;
1.18.3046   JANE      insert into "OE"."CUSTOMERS"("CUSTOMER_ID",
                      "CUST_FIRST_NAME","CUST_LAST_NAME",
                      "CUST_ADDRESS","PHONE_NUMBERS","NLS_LANGUAGE",
                      "NLS_TERRITORY","CREDIT_LIMIT","CUST_EMAIL",
                      "ACCOUNT_MGR_ID") values ('9839','Edgar',
                      'Cummings',NULL,NULL,NULL,NULL,
                       NULL,NULL,NULL);
1.18.3046    JANE      commit;
1.11.3047    JANE      set transaction read write;
1.11.3047    JANE      insert into "OE"."CUSTOMERS"("CUSTOMER_ID",
                       "CUST_FIRST_NAME","CUST_LAST_NAME",
                       "CUST_ADDRESS","PHONE_NUMBERS","NLS_LANGUAGE",
                       "NLS_TERRITORY","CREDIT_LIMIT","CUST_EMAIL",
                       "ACCOUNT_MGR_ID") values ('8933','Ronald',
                       'Frost',NULL,NULL,NULL,NULL,NULL,NULL,NULL);
1.11.3047    JANE      commit;
1.8.3054     RON       set transaction read write;
1.8.3054     RON       insert into "HR"."JOBS"('JOB_ID","JOB_TITLE",
                       "MIN_SALARY",'MAX_SALARY') values ('9566',
                       'FI_ENTRY',NULL,NULL);
1.8.3054     RON       commit;
```

注意，XIDUSN、XIDSLT 和 XIDSQN 的组合可以唯一地标识一个事务。在上述两次同样的日志挖掘中，前者按照用户的操作顺序输出操作记录，后者则按事务为单位来组织挖掘记录，并按事务的先后提交顺序输出。很显然，后一种日志挖掘的输出更有意义，也更容易理解数据库的数据变更。

3.7　LogMiner 会话及其步骤

运行日志挖掘 LogMiner，启动和停止挖掘会话、建立被挖掘的日志文件列表，都是使用 PL/SQL 包 DBMS_LOGMNR，当需要提取 LogMiner 字典至日志流或 OS 文件时，还需要使用 PL/SQL 包 DBMS_LOGMNR_D（当 LogMiner 字典使用在线目录时，可不使用该包）。

包 DBMS_LOGMNR 和 DBMS_LOGMNR_D 属于 SYS 模式对象，调用它们的用户需要 EXECUTE_CATALOG_ROLE 角色权限。

使用 LogMiner 实施日志挖掘的步骤如下：

（1）在源数据库中启动补充日志。这一步需要在被挖掘日志生成前执行，以便日志中有必要的信息支持 LogMiner 日志挖掘。可以根据需要在数据库级或相关的表级启动补充日志记录。

（2）执行正常的用户访问与操作，生成事务日志。

（3）确定 LogMiner 字典的使用方式。除非准备使用在线目录（Online Catalog）方式，否则需要使用 DBMS_LOGMNR_D 提取 LogMiner 字典至源数据库的日志流中或指定的数据库之外的 OS 文件中。

使用在线目录方式，需要在启动 LogMiner 时指定参数 OPTIONS 包含 DICT_FROM_ONLINE_CATALOG 选项。

当使用提取 LogMiner 字典至日志流方法并手工执行日志挖掘时，在建立被挖掘日志文件列表时，需要额外将包含 LogMiner 字典的日志文件包含其中。

（4）建立日志文件列表。根据需要确定需要挖掘的重做数据范围，从而进一步确定日志文件的详细列表。建立被挖掘的日志文件列表，有多种方式，请参考本章前面的介绍。典型的方法是使用包过程 DBMS_LOGMNR.ADD_LOGFILE 建立文件列表。

下面的演示建立了具有两个归档日志文件的文件列表，其中一个是需要挖掘的归档日志，另一个是包含 LogMiner 字典的归档文件。

```
SYS@NETDB>EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
> LOGFILENAME => '/ARCHIVELOG/O1_MF_1_65_8LR1WWYR_.ARC', -
> OPTIONS => DBMS_LOGMNR.NEW);

PL/SQL procedure successfully completed.

SYS@NETDB>EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
> LOGFILENAME => '/ARCHIVELOG/O1_MF_1_67_8LR2PPTV_.ARC', -
> OPTIONS => DBMS_LOGMNR.ADDFILE);

PL/SQL procedure successfully completed.
```

（5）启动 LogMiner 日志挖掘会话。使用过程 DBMS_LOGMNR.START_LOGMNR 启动日志挖掘会话。下面的示例使用提取 LogMiner 至日志流的方式。

```
SYS@NETDB>exec DBMS_LOGMNR.START_LOGMNR(OPTIONS => -
> DBMS_LOGMNR.DICT_FROM_REDO_LOGS + -
```

```
> DBMS_LOGMNR.COMMITTED_DATA_ONLY);

PL/SQL procedure successfully completed.
```

Oracle 建议指定使用 LogMiner 字典方式的选项，使用在线目录方式，指定参数 OPTIONS 包含 DICT_FROM_ONLINE_CATALOG 选项。当使用“提取 LogMiner 字典至数据库之外的 OS 文件”方式时，则在启动日志挖掘会话中需要为过程 DBMS_LOGMNR.START_LOGMNR 指定 DICTFILENAME 参数。

（6）查看 LogMiner 日志挖掘结果。通过查询视图 V$LOGMNR_CONTENTS 获得 LogMiner 日志挖掘的结果，可以根据其字段设置各种过滤条件。例如：

```
SYS@NETDB>select operation,sql_redo from v$logmnr_contents
  2  where seg_owner='SCOTT';

OPERATION  SQL_REDO
-------------------------------------------------------------
INSERT     insert into "SCOTT"."DEPT"("DEPTNO","DNAME","LOC")
           values ('50','Mining','Beijing');

UPDATE     update "SCOTT"."DEPT" set "LOC" = 'Beijing' where
           "DEPTNO" = '10' and "LOC" = 'NEW YORK' and ROWID =
           'AAAQj1AAEAAACh3AAA';

UPDATE     update "SCOTT"."DEPT" set "LOC" = 'Beijing' where
           "DEPTNO" = '20' and "LOC" = 'DALLAS' and ROWID = '
           AAAQj1AAEAAACh3AAB';
```

（7）结束 LogMiner 日志挖掘会话。当日志挖掘任务完成后，需要及时关闭 LogMiner 日志挖掘会话。使用过程 DBMS_LOGMNR.END_LOGMNR 结束当前日志挖掘会话，该过程不需要参数。

```
SYS@NETDB>exec DBMS_LOGMNR.END_LOGMNR();

PL/SQL procedure successfully completed.
```

及时关闭 LogMiner 日志挖掘会话是很有必要的，它释放挖掘会话所占用的所有资源，并释放打开的归档日志文件和 LogMiner 字典文件。

3.8 日志挖掘典型案例

案例 1 挖掘已提交的事务

在最新的归档日志中挖掘所有的数据修改，并以事务为单位，以提交的先后顺序查看事务处理过程。

（1）确定归档日志：

```
SYS@NETDB> SELECT NAME FROM V$ARCHIVED_LOG
  2  WHERE FIRST_TIME =
  3  (SELECT MAX(FIRST_TIME) FROM V$ARCHIVED_LOG);

NAME
--------------------------------------------------------
/ARCHIVELOG/O1_MF_1_70_8LR43GM1_.ARC
```

（2）建立日志文件列表：

```
SYS@NETDB> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
LOGFILENAME => '/ARCHIVELOG/O1_MF_1_70_8LR43GM1_.ARC', -
OPTIONS => DBMS_LOGMNR.NEW);
```

（3）启动日志挖掘会话：

```
SYS@NETDB> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
OPTIONS => DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG + -
DBMS_LOGMNR.COMMITTED_DATA_ONLY);
```

（4）查询日志挖掘结果：

```
SYS@NETDB> SELECT username,(XIDUSN||'.'||XIDSLT||'.'||XIDSQN)
AS XID, SQL_REDO, SQL_UNDO FROM V$LOGMNR_CONTENTS
WHERE username IN ('HR', 'OE');

USR XID SQL_REDO SQL_UNDO
--------------------------------------------------------
...
```

（5）结束日志挖掘会话：

```
SYS@NETDB> EXECUTE DBMS_LOGMNR.END_LOGMNR();
```

案例 2 限定重做数据的范围

通过限定时间范围来过滤 LogMiner 日志挖掘的输出内容。当然可以在查询

视图 V$LOGMNR_CONTENTS 设置查询条件（类似 timestamp > '2003-01-10 15:59:53'）来过滤 LogMiner 日志挖掘的输出。

如果通过时间范围事先限定被挖掘重做数据的范围，效率会更高。为此事先创建如下过程专门用来通过时间限定被挖掘日志文件的范围。示例如下：

```
SYS@NETDB>CREATE OR REPLACE PROCEDURE my_add_logfiles (
  2  in_start_time IN DATE) AS
  3    CURSOR c_log IS
  4    SELECT NAME FROM V$ARCHIVED_LOG
  5    WHERE FIRST_TIME >= in_start_time;
  6    count pls_integer := 0;
  7    my_option pls_integer := DBMS_LOGMNR.NEW;
  8  BEGIN
  9    FOR c_log_rec IN c_log LOOP
 10      DBMS_LOGMNR.ADD_LOGFILE(
 11        LOGFILENAME => c_log_rec.name,
 12        OPTIONS => my_option);
 13      my_option := DBMS_LOGMNR.ADDFILE;
 14        DBMS_OUTPUT.PUT_LINE('Added logfile '||c_log_rec.
name);
 15    END LOOP;
 16  END;
 17  /

Procedure created.
```

（1）创建被挖掘日志文件列表：

```
SYS@NETDB>alter session set nls_date_format='YYYY-MM-DD
HH24:MI:SS';

Session altered.

SYS@NETDB> set serveroutput on;
SYS@NETDB> EXECUTE my_add_logfiles( -
in_start_time => '2013-02-25 12:00:00');
Added logfile /DB_FRA/ARCHIVELOG/O1_MF_1_65_8LR1WWYR_.ARC
Added logfile /DB_FRA/ARCHIVELOG/O1_MF_1_66_8LR2PJB8_.ARC
Added logfile /DB_FRA/ARCHIVELOG/O1_MF_1_67_8LR2PPTV_.ARC
Added logfile /DB_FRA/ARCHIVELOG/O1_MF_1_68_8LR421YR_.ARC
```

```
Added logfile /DB_FRA/ARCHIVELOG/O1_MF_1_69_8LR427B8_.ARC
Added logfile /DB_FRA/ARCHIVELOG/O1_MF_1_70_8LR43GM1_.ARC

PL/SQL procedure successfully completed.
```

（2）查询待挖掘日志文件列表的内容：

```
SYS@NETDB> SELECT FILENAME name, LOW_TIME start_time, FILESIZE
bytes
FROM V$LOGMNR_LOGS;

NAME                          START_TIME               BYTES
-----------------------------------------------------------
.../O1_MF_1_65_8LR1WWYR_.ARC 2013-02-25 15:30:24   28274176
.../O1_MF_1_66_8LR2PJB8_.ARC 2013-02-26 08:53:48     419328
.../O1_MF_1_67_8LR2PPTV_.ARC 2013-02-26 09:07:28   13504000
.../O1_MF_1_68_8LR421YR_.ARC 2013-02-26 09:07:34   43432960
.../O1_MF_1_69_8LR427B8_.ARC 2013-02-26 09:30:41   48017920
.../O1_MF_1_70_8LR43GM1_.ARC 2013-02-26 09:30:46   40889856
```

（3）启动 LogMiner 日志挖掘会话：在启动日志挖掘会话时，还可以使用 DBMS_LOGMNR.START_LOGMNR 过程的时间参数 STARTTIME 和 ENDTIME 进一步限定被挖掘重做数据的范围。

```
SYS@NETDB>EXECUTE DBMS_LOGMNR.START_LOGMNR(-
> STARTTIME => '2013-02-25 16:00:00', -
> ENDTIME => '2013-02-26 09:00:00', -
> OPTIONS => DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG + -
> DBMS_LOGMNR.COMMITTED_DATA_ONLY + -
> DBMS_LOGMNR.PRINT_PRETTY_SQL);

PL/SQL procedure successfully completed.
```

（4）查询日志挖掘的结果：

```
SYS@NETDB> SELECT TIMESTAMP, (XIDUSN||'.'||
XIDSLT||'.'||XIDSQN) AS XID, SQL_REDO FROM V$LOGMNR_CONTENTS
WHERE SEG_OWNER = 'SCOTT';

TIMESTAMP XID SQL_REDO
-----------------------------------------------------.....
```

（5）结束日志挖掘会话：

```
SYS@NETDB> EXECUTE DBMS_LOGMNR.END_LOGMNR();

PL/SQL procedure successfully completed.
```

案例 3　跟踪特定用户的数据处理

本例展示通过日志挖据掌握特定数据库用户在特定时间范围内的数据库操作。这样的事后检查操作可以脱离源数据库完成，因此，本例选择使用 LogMiner 字典的方式是创建数据库之外的 OS 文件。

（1）创建 LogMiner 字典文件：具体创建 LogMiner 字典文件的方法请参考本章前面的介绍。字典文件要存储在参数 utl_file_dir 设置的目录位置。

（2）建立日志文件列表：注意在使用 DBMS_LOGMNR.ADD_LOGFILE 过程建立待分析日志文件列表，重做数据的时间范围要超过后面在 DBMS_LOGMNR.START_LOGMNR 过程中通过时间参数或 SCN 参数设置的时间范围。

（3）启动日志挖掘会话并设置挖掘重做数据的范围：

```
SYS@NETDB> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
DICTFILENAME => 'logmnr_dict_file.ora', -
STARTTIME => TO_DATE('2013-02-01 08:30:00','YYYY-MM-DD
HH24:MI:SS'),
ENDTIME => TO_DATE('2013-02-01 08:45:00', 'YYYY-MM-DD
HH24:MI:SS'));
```

（4）通过限定用户查看日志挖掘的结果：

```
SYS@NETDB> SELECT SQL_REDO, SQL_UNDO FROM V$LOGMNR_CONTENTS
WHERE USERNAME = 'SCOTT';
...
```

（5）结束 LogMiner 日志会话。

案例 4　统计特定表上的用户访问

本例通过 LogMiner 日志挖掘 DBA 可以了解在特定时间范围内用户表上的 DML 操作及其频繁程度。

在执行指定时间范围内的日志挖掘操作后，DBA 可以对 V$LOGMNR_CONTENTS 视图执行类似如下查询，了解数据库中各模式下的表（排除系统表，特征是表名具有 $ 后缀）的 DML 操作的频次。

```
SELECT SEG_OWNER, SEG_NAME, COUNT(*) AS Operations
```

```
FROM V$LOGMNR_CONTENTS
WHERE SEG_NAME NOT LIKE '%$'
GROUP BY SEG_OWNER, SEG_NAME
ORDER BY Operations DESC;

SEG_OWNER SEG_NAME OPERATIONs
--------- ----------------------------
CUST ACCOUNT 3843
OE ORDERS 3225
SCOTT SALGRADE 2716
SCOTT MEGADONOR 321
HR EMPLOYEES 233
SYSTEM TEST 11
```

3.9 日志挖掘不支持的数据类型

数据库记录事务日志的主要目的是用于实例恢复和介质恢复，通过事务日志可以实现完全的数据库恢复。重做数据的记录机制主要是为满足这一目的而设计的。Oracle 通过对事务日志中的重做数据的日志挖掘，可以从中还原出数据库曾经执行的 DML、DDL、DCL 等数据库操作。从技术上，通过事务日志挖掘出对应的 SQL 语句，LogMiner 还存在一些不支持的数据类型及其存储属性。但从 Oracle 的日志挖掘接口 LogMiner 诞生以来，随着 Oracle 版本的不断升级，不支持的数据类型及其存储属性的范围在持续地缩小，截至 Oracle 11.2 版本，目前这个不支持的范围如下：

※ BFILE datatype。

※ Simple and nested abstract datatypes (ADTs)。

※ Collections (nested tables and VARRAYs)。

※ Object refs。

在日志挖掘的数据处理中，当遭遇不支持的数据类型及其存储属性时，LogMiner 会在 V$LOGMNR_CONTENTS 视图对应记录的 OPERATION 字段填充为 UNSUPPORTED，对应的 SQL_REDO 和 SQL_UNDO 字段保持为 NULL 空值。

第4章

Data Guard 中的进程架构

4.1 DG 环境的进程架构概述

Data Guard 通过数据库事务日志的机制实现对主数据库的保护。受到 Data Guard 保护的主数据库将事务日志记录到本地的同时，也将事务日志传向备用数据库端。当备用数据库接收到从主数据库传输过来的事务日志时，根据处理日志方法的不同，可以把 Data Guard 分为物理备用数据库和逻辑备用数据库两种。物理备用数据库，就是备用数据库处于 Mount 的状态下，直接利用基于事务日志的介质恢复技术，把日志文件中记录的数据变更应用在备用数据库的物理存储上，从而实现与主数据库的数据同步；逻辑备用数据库是处于正常的打开状态的，当它接收到新的日志信息后，备用端对其进行日志挖掘（LogMiner），把日志中记录的变更信息，转换成对应的 SQL 语句，并在逻辑备用数据库上执行这些 SQL 语句，从而实现与主数据库的数据同步，逻辑备用数据库支持在数据同步的同时，进行数据的查询、报表等操作。

将生产用数据库（主数据库）的重做日志传输到远程的目的地，主要有两个目的：一是传输到物理备用或逻辑备用数据库端，以实现数据更新的传播；二是建立一个远程归档日志的存储库 ALR（Archive Log Repository），实现远程异地归档，这样可以确保数据库系统日志的存储安全。此种情况下 ALR 端需要运行一个物理备用数据库实例，并通过一个备用数据库控制文件使实例处于加载状态，ALR 端并不需要任何数据文件和联机日志文件。

对于物理备用数据库和逻辑备用数据库的管理与维护，由于其内部的实现机制不同，两者还是存在较大差异。通过对 V$DATAGUARD_STATUS 视图的观察可以了解主数据库端、物理备用端、逻辑备用端的运行状况。该视图显示那些被自动触发写入 Alert.log 或服务器 Trace 文件的事件。下面的查询可以分别在主备用端执行，获得对应端关于 Dataguard 的运行状况。在此给出的是来自物理备用数据库端的查询示例。

```
SQL> SELECT MESSAGE FROM V$DATAGUARD_STATUS;

MESSAGE
--------------------------------------------------
Media Recovery Waiting for thread 1 sequence 2148
Redo Shipping Client Connected as PUBLIC
-- Connected User is Valid
RFS[1]: Assigned to RFS process 1617
RFS[1]: Identified database type as 'physical standby'
Redo Shipping Client Connected as PUBLIC
-- Connected User is Valid
RFS[2]: Assigned to RFS process 1619
RFS[2]: Identified database type as 'physical standby'
Media Recovery Log /phydb/arch1/1_2148_758642906.dbf
Media Recovery Waiting for thread 1 sequence 149
```

4.2 备用数据库的实现框架

Data Guard 实现备用数据库的主要目标是使得备用数据库成为主数据库的在线副本，从而实现对主数据库的数据保护和高可用性。Data Guard 的实现架构是将主数据库的数据变更及时传播到一个到多个备用数据库，即这种传播可以是一对多的关系。

从备用端应用方式的角度，备用数据库可以划分为物理备用数据库和逻辑备用数据库，但不论是物理备用数据库还是逻辑备用数据库，Data Guard 的核心是基于重做日志的两类服务：重做日志传输服务（Redo Transport Service）和重做日志应用服务（Apply Service），其中对应于应用服务又可根据物理备用和逻辑备用分为重做应用（Redo Apply）和 SQL 应用（SQL Apply）。如图 4-1 所示，Redo 应用由 MRP（Managed Recovery Process）进程完成，SQL 应用由 LSP（Logical Standby Process）进程完成。

与独立的主数据库相对照，Data Guard 架构下的核心进程有如下几个。

※ LNS（Logwriter Network Server）进程：用于在主数据库中将事务日志通过网络向远程目标点传输。

※ RFS（Remote File Server）进程：在备用数据库端用于接收与组织来自主数据库端的事务日志，该进程使传输的日志在备用端落地。

※ MRP（Managed Recovery Process）进程：托管恢复进程，也可理解为 Media Recovery Coordinator，该进程负责协调组织物理备用数据库端的基于事务日志的介质恢复。

※ LSP（Logical Standby Process）进程：该进程负责组织协调逻辑备用数据库端的日志挖掘与 SQL 应用。

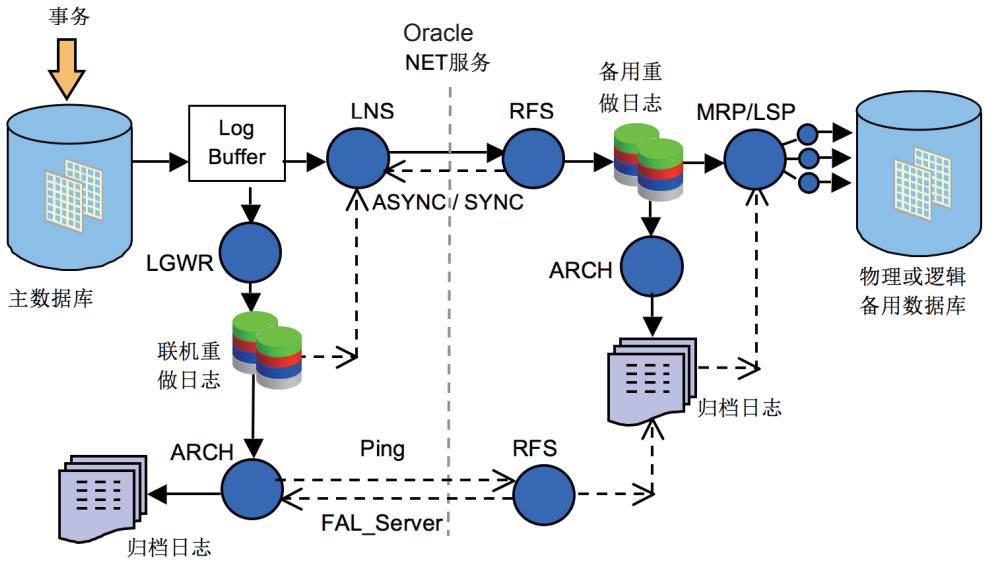


图 4-1　Data Giard 的主要进程结构

通过上述进程架构，Data Guard 在数据保护和高可用性的基础上实现了如下两重分离：一是备用数据库与主数据库的相互分离，两者仅通过单向传播的事务日志联系起来，而且备用端还实现了对接收到的事务日志进行有效性验证，从而防止主数据库的物理损坏（如数据块讹误）传播到备用数据库端。二是实现了日志传输和日志应用的分离，即备用端事务日志的接收和数据变更的应用是两套相互独立的过程，两者不会相互影响，数据变更是否应用并不会影响备用端对事务日志的接收。

另外，Data Guard 可以和 Oracle RAC 的集群环境有机地结合起来，实现更高水平的数据保护、数据访问和高可用性。在 Data Guard 环境中，主数据库和备用数据库都可以是 RAC 数据库，即主备用数据库的单实例和多实例运行可以任意组合。

如果 Data Guard 的主数据库是 RAC 数据库，则集群环境中的每个实例有自己的 LNS 进程，分别负责向备用端传输自己的线程日志。不同线程的事务日志传输到备用端后按照 SCN 的先后顺序合并为统一的日志流，由备用端的应用环节实施一致的数据更新。需要注意的是，如果备用数据库是 RAC 环境，则只能由其中的一个实例实施不同线程的日志流的合并和后续的数据变更的应用。

下面分别详细介绍 Data Guard 的这两类核心服务。

4.3 日志传输服务

Data Guard 的重做日志传输服务负责将主数据库端产生的事务日志的一份复件传输到一个或多个备用数据库端。在主数据库端，LGWR 进程负责将事务日志写到本地的重做联机日志，另一个独立的进程 LNS（Logwriter Network Server）负责将事务日志通过 Oracle Net 服务传输到备用数据库端。

在备用数据库端，由 LNS 进程传输的事务日志由专门的进程 RFS（Remote File Server）负责接收。RFS 进程在备用端接收重做数据，并将其写入备用重做日志（Standby Redo Log）。如果备用端没有创建备用重做日志，RFS 会将接收到的重做数据以文件的形式写入由参数 standby_archive_dest 指定的位置。如果备用端既没有备用重做日志，又没有设置该参数，则 RFS 会将重做数据以文件的形式写入备用端默认的归档日志存储位置（相当于远程传递来的归档日志文件）。

LNS 进程是专门用于远程日志传输的。如果一个主数据库配置了多个备用数据库，则在主数据库中存在多个 LNS 进程对应于多个备用数据库。如果主数据库是 RAC（Real Application Cluster）的多实例环境，则每个数据库实例都有自己的 LNS 进程对应于特定的备用数据库。

LNS 支持两种日志传输模式：异步传输（ASYNC）和同步传输（SYNC）。

4.3.1 异步日志传输（ASYNC）

对于 Data Guard 中的主数据库，就记录事务日志而言，进程 LGWR 用于本地目标的日志记录，进程 LNS 用于远程备用目标的日志传输。对于远程目标，这里的异步日志传输模式（Asynchronous Transport，简写为 ASYNC），指的是进程 LNS 独立地用于远程目的地的日志传输，但能否传输成功或传输是否及时，并不影响进程 LGWR 对于事务日志的记录，因此，这是一种几乎对主数据库没有影响的远程日志传输方式。

在主数据库的运行中，事务处理的日志流（Stream）只有一个，即重做日志缓存（Redo Log Buffer）中的内容只有一份，只是 LGWR 和 LNS 进程将其写向不同的目的地。本地的日志记录不存在网络传输、备用实例是否启动等条件是否具备的问题，远程传输则不同。由于远程目标点采用异步传输，LNS 进程记录日志的进度就很可能落后于 LGWR 的进度。在正常情况下，LGWR 和 LNS 进程都是直接从日志缓存中读取日志内容将其写向不同的目的地。当 LNS 进程的进度落后于 LGWR 的进度，或日志缓存已经被重用，此时 LNS 进程就会从本地的目的地（即联机日志）中去读取日志内容进行远程传输。

4.3.2 同步日志传输（SYNC）

这是一种“零数据丢失”的日志传输模式，但需要的传输条件比异步传输苛刻，并且对主数据库的性能有影响。在主数据库中，当用户执行事务处理的 Commit 指

令时，进程 LGWR 和 LNS 同时会将当前事务的日志写向本地和远程的目标，并且只有当 LNS 返回事务日志已确定写入远程备用站点的确认信息后，LGWR 才返回给用户事务提交成功的消息。此时才允许用户继续执行后续的事务，否则 LGWR 将持续等待 LNS 进程返回远程的确认消息。

之所以这种日志传输模式对主数据库的性能有影响，就是因为这里的 LGWR 和 LNS 进程在写日志时需要保持协同一致，这正是同步的含义。在独立数据库或主数据库异步传输模式的情形中，日志只要通过 LGWR 记入联机日志后当前事务即可结束（Commit），但在同步传输的情形中，日志需要同时写入远程目标和本地目标后，LGWR 才能返回事务成功提交、当前事务结束的信息。

同步传输模式中，日志写入远程备用目标的速度与网络状况、备用站点状况有关，具体地说，涉及如下因素：事务日志量的大小、可用的网络带宽、网络延迟（Network Round-trip Lateency）、备用站点日志记录的磁盘 I/O 性能等。很显然，这些因素都会消耗 LGWR 等待 LNS 回应的时间。如果 LNS 向远程站点传输日志的时间很长或一直没有回应（如网络中断、备用站点关闭等），主数据库如何处理？因此，在设置同步传输模式时，通常需要设置 LGWR 等待时限的参数（即 NET_TIMEOUT）。

4.3.3　日志间隔（Gap）的自动处理

在前面的异步传输模式中，LNS 进程远程传输日志的进度如果远远落后于 LGWR，此时需要传输的日志不仅不在日志缓存中，同时也不在联机日志中，即 LNS 需要传输的日志存在于主数据库的本地归档日志中。这种情形即表示备用数据库的日志和主数据库的日志两者出现了“日志间隔（Gap）”。

存在多种情况导致主备用数据库之间出现日志间隔，主要有两类：一是网络中断或备用站点关闭导致远程日志记录失败；二是 LNS 远程传输的效率过低，以至于 LNS 远程写的进度远远落后于本地的 LGWR 进程。对于后一种情形，需要改进网络传输，否则在实际应用中 Data Guard 的备用数据库方案很难正常运行。

解决日志间隔的途径有两类，一是从主数据库的角度，二是从备用数据库的角度。

从主数据库的角度，在远程连接中断期间，归档进程 ARCH 会不断地 Ping 备用站点以确定备用数据库的状态。一旦主备用站点的网络连接恢复，ARCH 进程会建立与备用数据库 RFS 进程的日志传输通道，并通过访问备用数据库的控制文件获得需要传输的归档日志清单，这样日志间隔中的归档日志就通过 ARCH-RFS 这样的传输方式传向备用数据库。实际上，这是 Data Guard 提供的第三种日志传输模式。与此同时，当主数据库出现日志切换操作时，正常地通过进程 LNS 和 RFS 之间的日志传输通道得以恢复。这样 LNS 进程传输当前最近的事务日志，而 ARCH 在后台处理日志间隔。

从备用数据库的角度，如果发现有日志间隔出现，备用数据库根据设置的 FAL_Server 参数会主动查找可用的 FAL Server（主数据库或其他物理备用站点），

从而建立 ARCH-RFS 的连接通道，实现日志间隔中归档日志的传输。

4.4 事务日志的应用服务

事务日志在备用端的应用分为两类,即重做应用(Redo Apply)和SQL应用(SQL Apply），下面分别叙述。

4.4.1 Redo 应用

物理备用数据库的数据变更采用 Redo 应用方式，实际上是在备用数据库上启动了基于事务日志的托管介质恢复（Managed Media Recovery）。为了获得更好的应用性能，Data Guard 启动并行的介质恢复，即在启动了物理备用数据库的托管介质恢复后，首先在实例中启动了 MRP 进程（它是一个用于介质恢复的协调进程和实施多线程日志合并的进程），并根据需要启动一个到多个具体的并行恢复进程（PRnn）实施并行的介质恢复，进程 PRnn 的数量由主机系统的 CPU 数量决定（在 Oracle 11g 中其数量等于 CPU 数量减 1），即这里的并行执行是 Data Guard 的默认行为。

根据 Oracle 介质恢复的机制，上面的应用过程要求物理数据库始终处于加载（Mount）状态，此时物理备用数据库并没有打开。要访问数据库中的数据，在 11g 之前的版本，物理备用数据库只能暂时以只读的方式打开，从而可以实现数据访问。但是，一旦以只读的方式打开，数据库必须停止对事务日志的应用，即物理备用数据库的 Redo 应用和只读访问是互斥的，这种状况限制了物理备用数据库只读访问的应用价值。

从 11g 开始，Oracle 扩展了物理备用数据库的在线访问功能，即 Active Data Guard，它将物理备用数据库的 Redo 应用和只读访问并行起来，在不需要停止 Redo 应用的同时提供数据的只读访问，这极大地提升了物理备用数据库的应用范围。利用这一特性，在实际应用中，可以从主数据库中卸载只读工作负荷，如将主数据库中的消耗资源的查询功能（如报表系统）卸载到物理备用数据库端执行，这极大地缓解了主数据库的访问压力。

另为，从 11g 开始，Data Guard 还提供了防止主数据库的物理损坏通过日志传播到物理备用数据库的功能，即物理备用数据库在执行 Redo 应用之前，可以对事务日志进行日志验证和数据校验。默认情况下，该功能关闭，可以通过设置参数 db_ultra_safe 打开该功能。

4.4.2 SQL 应用

逻辑备用数据库的数据更新是通过对接收到的事务数据进行日志挖掘，将事务数据最终转换一系列的 SQL，并执行这些 SQL 的过程。与直接的 Redo 应用相比，SQL 应用需要的环节更多，需要的进程结构也更复杂，对应地在逻辑备用数据库端

消耗的资源（如 CPU、内存、磁盘 I/O 等）也更多。SQL 应用的环节与进程结构如图 4-2 所示。

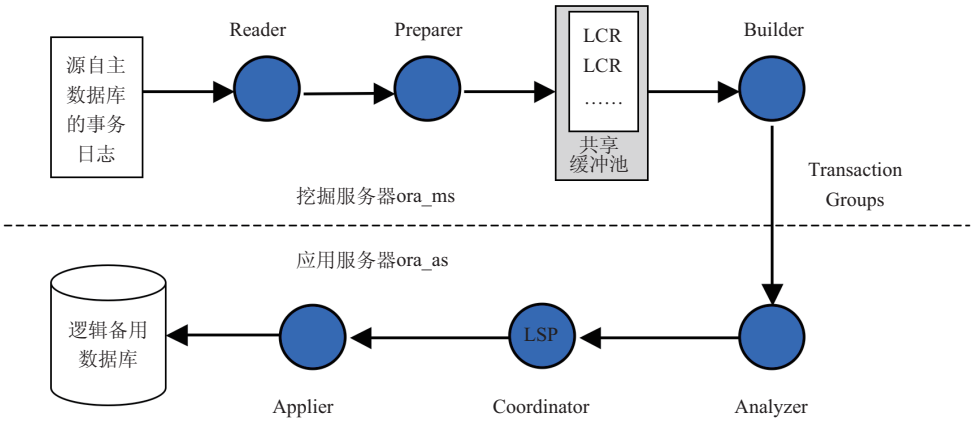


图 4-2　SQL 应用的进程结构

逻辑备用数据库的 SQL 应用由两部分组成：日志挖掘服务器部分（ora_ms）和变更应用服务器部分（ora_as）。

日志挖掘部分包括下列 3 类进程。

※ READER 进程：该进程负责从接收来自主数据库重做数据的日志文件（通常是备用重做日志文件或外部归档文件）中读取重做记录，将其放入共享缓冲池。

※ PERPARER 进程：通常会有多个 PREPARER 进程，该进程负责将共享缓冲池中的重做记录分解为一系列逻辑变更记录 LCR（Logical Change Record），单个事务记录可以转换为多个 LCR。一个 LCR 实际上对应于一个物理数据块的变更向量。

※ BUILDER 进程：该进程将一系列 LCR 分组为事务。

应用服务部分包括下列 3 类进程。

※ ANALYZR 进程：该进程负责分析日志挖掘服务器输出的事务组，确定事务之间的依赖关系，并过滤出逻辑备用不支持的事务。

※ COORDINATOR 进程：即 LSP 进程，事务协调进程，将 ANALYZR 进程的输出以特定的事务顺序提交给 APPLIER 进程。

※ APPLIER 进程：该进程负责实际的事务执行。通常会存在多个 APPLIER 进程，以在备用端支持并发事务的应用。

这里介绍的 SQL 应用涉及的进程架构中，有两方面对 SQL 应用的性能存在显著影响：一方面是 PREPARER 进程和 APPLIER 进程的数量（其他进程只有一个），并且两者在数量上存在一定的比例关系（通常为 1:5 到 1:20）；另一方面是共享缓冲池中的 LCR 缓存的大小。

在逻辑备用数据库中，控制与调整 SQL 应用的性能存在一系列参数（参见 DBA_LOGSTDBY_PARAMETERS），它们都可以通过 PL/SQL 包 DBMS_LOGSTDBY 的 APPLY_SET 过程来设置，其中主要的参数介绍如下。

※ MAX_SGA：设置 LCR 缓存的参数，典型地需要 200MB 的大小。

※ MAX_SERVERS：设置用于 SQL 应用的进程总数（不包含 LSP 进程），其实际数量应该与 CPU 数量有关，典型地可以设置为主机 CPU 数量的 8 倍。

※ PREPARE_SERVERS：设置 PREPARER 进程的数量。

※ APPLY_SERVERS：设置 APPLIER 进程的数量，该进程是 SQL 应用的主力，通常其数量远多于其他进程。MAX_SERVERS=PREPARE_SERVERS+APPLY_SERVERS+3，这里的 3 是考虑到 SQL 应用环境始终存在 READER、BUILDER 和 ANALYZR 这 3 个进程。

第5章 构建物理备用数据库

本章根据一个典型的生产用数据库 ORADB 具体创建一个物理备用数据库，在创建备用数据库的过程中介绍相关的知识与技能。

5.1 主数据库的准备

备用数据库是通过接收并应用主数据库传送的 Redo Data 来实现更新的，这就要求主数据库必须满足几个基本的前提条件。

5.1.1 检查并设置数据库

首先，主数据库必须处于强制日志（Force Logging）模式下运行，这是要防止主数据库中存在直接的数据修改而不记录日志的行为。

检查强制日志模式如下：

```
SYS@ORADB>select dbid, name, force_logging from v$database;

DBID         NAME      FORCE_LOGGING
-------------------------------------------
2551222283   ORADB     NO
```

启动强制日志模式如下：

```
SYS@ORADB>alter database force logging;

Database altered.
```

取消强制日志模式如下：

```
SYS@ORADB> alter database no force logging;
```

```
Database altered.
```

其次，检查主数据库的日志运行模式，Data Guard 要求主数据库必须在归档模式下运行(Archive log Mode),此模式下数据库可以连续完整地保存事务日志。

```
SYS@ORADB>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      117
Next log sequence to archive    118
Current log sequence            118
```

如果数据库在非归档模式下运行，则需要做如下调整：

```
SYS@ORADB>archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     117
Current log sequence           118
SYS@ORADB>shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ORADB>startup mount
ORACLE instance started.

Total System Global Area        167772160 bytes
Fixed Size                      1247876 bytes
Variable Size                   62915964 bytes
Database Buffers                96468992 bytes
Redo Buffers                    7139328 bytes
Database mounted.
SYS@ORADB>alter database archivelog;

Database altered.

SYS@ORADB>alter database open;

Database altered.
```

最后，要检查数据库实例的口令文件。Data Guard 环境下的每个数据库必须强制使用口令文件，并且备用数据库口令文件中 SYS 用户的口令必须和主数据库 SYS 用户的口令完全一致。在 Oracle 11G 中，备用数据库的口令文件不能按相同的 SYS 用户口令直接创建，而是需要将主数据库的口令文件复制为备用数据库的口令文件。若主数据库没有口令文件，此时需要使用下面的方式创建：

```
$orapwd file=$ORACLE_HOME/database/pwdSID.ora
password=admin entries=5
```

5.1.2　设置必要的主数据库参数

我们计划按照如表 5-1 所示的内容来构建 Data Guard 环境。假设一个数据库名称为 ORADB 的主数据库运行在北京，在上海配置一个物理备用数据库。

表 5-1　主备用数据库及其网络服务

DG ROLE	DB_NAME	INSTANCE_NAME	DB_UNIQUE_NAME	TNS NAME
主数据库	ORADB	BJING	BJING	BJING
物理备用	ORADB	SHHAI	SHHAI	SHHAI

为 Data Guard 环境设置主数据库的必要参数，需要设置的参数如下：数据库标识 DB_UNIQUE_NAME，通过 DB_UNIQUE_NAME 设置 Data Guard 环境参数 LOG_ARCHIVE_CONFIG，控制主数据库向远程备用端传输 Redo Data 的参数 LOG_ARCHIVE_DEST_n 等。参考设置如下：

```
DB_UNIQUE_NAME = BJING
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'
LOG_ARCHIVE_DEST_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
LOG_ARCHIVE_DEST_2 =
'SERVICE=SHHAI LGWR ASYNC DB_UNIQUE_NAME= SHHAI'
```

在上述参考设置中，LOG_ARCHIVE_DEST_1 设置向备用数据库传输日志。另外，在大多数 Oracle 数据库中，参数 LOG_ARCHIVE_MAX_PROCESSES 的默认设置为 2，在多数情况下，此值显得不足，特别是在多路归档的情况下，建议适当提高。

此处列出主数据库 ORADB 其他参数的参考设置：

```
######################################################
db_name=ORADB
db_unique_name=BJING
instance_name=BJING

memory_target=256M
```

```
db_block_size=8192

db_create_file_dest='+GRP0'
db_create_online_log_dest_1='+GRP1'
db_recovery_file_dest='+GFRA'
db_recovery_file_dest_size=2G
log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'

control_files='+GRP1/bjing/controlfile/current.256.836859253'

undo_management=auto
undo_tablespace=UNDOTBS

open_cursors=300
processes=150

local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'

remote_login_passwordfile=EXCLUSIVE
compatible=11.2.0.1.0
###############################################################

LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'
LOG_ARCHIVE_DEST_2 = 'SERVICE=SHHAI LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=SHHAI'

#################For Switch to Physical Standby################
FAL_CLIENT = BJING
FAL_SERVER = SHHAI

STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT = '+GRP1/shhai', '+GRP0/bjing'
LOG_FILE_NAME_CONVERT = '+GRP0/shhai', '+GRP1/bjing'

###############################################################
```

5.2　备用数据库控制文件

备用数据库是主数据的副本，但备用数据库的控制文件与主数据库的控制文件不同，Oracle 通过控制文件首要识别数据库在 Data Guard 中的角色，需要通过专门的指令在主数据库中为备用数据库创建控制文件。

备用数据库的数据文件是通过对主数据库数据文件备份（Backup Set 或 Image Copy）的还原（Restore）实现的，在使用 RMAN 对主数据库进行备份时，备份信息的存储库 Repository 存在于控制文件中，因此，在创建备用控制文件之前，最好使用 RMAN 对主数据库执行最新一次的备份。

```
RMAN> backup database plus archivelog;

Starting backup at 15-JAN-14
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=197 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=44 RECID=1 STAMP=836907482
input archived log thread=1 sequence=45 RECID=2 STAMP=836908379
channel ORA_DISK_1: starting piece 1 at 15-JAN-14
channel ORA_DISK_1: finished piece 1 at 15-JAN-14
piece handle=+GFRA/bjing/backupset/2014_01_15/annnf0_tag201401
15t103259_0.260.836908381 tag=TAG20140115T103259 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 15-JAN-14

Starting backup at 15-JAN-14
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=+GRP0/bjing/datafile/
system.256.836859253
input datafile file number=00003 name=+GRP0/bjing/datafile/
undotbs.258.836859509
input datafile file number=00002 name=+GRP0/bjing/datafile/
sysaux.257.836859257
```

```
input datafile file number=00004 name=+GRP0/bjing/datafile/
example.260.836859607
channel ORA_DISK_1: starting piece 1 at 15-JAN-14
channel ORA_DISK_1: finished piece 1 at 15-JAN-14
piece handle=+GFRA/bjing/backupset/2014_01_15/
nnndf0_tag20140115t103301_0.261.836908381
tag=TAG20140115T103301 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:25
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 15-JAN-14
channel ORA_DISK_1: finished piece 1 at 15-JAN-14
piece handle=+GFRA/bjing/backupset/2014_01_15/
ncsnf0_tag20140115t103301_0.262.836908409
tag=TAG20140115T103301 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 15-JAN-14

Starting backup at 15-JAN-14
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=46 RECID=3 STAMP=836908409
channel ORA_DISK_1: starting piece 1 at 15-JAN-14
channel ORA_DISK_1: finished piece 1 at 15-JAN-14
piece handle=+GFRA/bjing/backupset/2014_01_15/
annnf0_tag20140115t103329_0.264.836908409
tag=TAG20140115T103329 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 15-JAN-14
```

接下来在主数据库中创建备用数据库的控制文件。有如下两种方法：

```
SYS@BJING>ALTER DATABASE CREATE STANDBY CONTROLFILE AS '...';
```

```
Database altered.
```

或者

```
RMAN> backup as copy format '+DG_PHY'
2> current controlfile for standby;
```

注意，此处的 RMAN backup 指令使用了 as copy 选项，否则，生成的控制文件备份在备用数据库中不能直接使用，需要还原（Restore）。

备用数据库的控制文件也可以先在主数据库中创建（上面的两种方法都可以），然后在备用端再执行控制文件的还原。

（1）主数据库端创建备用控制文件，代码如下：

```
RMAN> backup current controlfile for standby;

Starting backup at 15-JAN-14
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including standby control file in backup set
channel ORA_DISK_1: starting piece 1 at 15-JAN-14
channel ORA_DISK_1: finished piece 1 at 15-JAN-14
piece handle=+GFRA/bjing/backupset/2014_01_15/
ncnnf0_tag20140115t103627_0.265.836908589
tag=TAG20140115T103627 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 15-JAN-14
```

（2）在备用端还原备用控制文件，代码如下：

```
RMAN> connect target /

connected to target database: ORADB (not mounted)

RMAN> restore standby controlfile
2> from '+GFRA/bjing/backupset/2014_01_15/
ncnnf0_tag20140115t103627_0.265.836908589';

Starting restore at 15-JAN-14
```

```
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=5 device type=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:05
output file name=+GRP0/shhai/controlfile/current.261.836908945
Finished restore at 15-JAN-14
```

（3）修改备用数据库参数 control_files 的指向。

5.3 构造备用数据库运行环境

要启动备用数据库实例，首先需要备用数据库的初始化参数，为此可以在原主数据库初始化参数的基础上进行改造，参数的参考设置如下：

```
#####################################################
db_name=ORADB
db_unique_name=SHHAI
instance_name=SHHAI

memory_target=256M
db_block_size=8192

db_create_file_dest='+GRP1'
db_create_online_log_dest_1='+GRP0'
db_recovery_file_dest='+GFRA'
db_recovery_file_dest_size=2G
log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'

control_files='+GRP0/shhai/controlfile/current.261.836908945'

undo_management=auto
undo_tablespace=UNDOTBS

open_cursors=300
processes=150

local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'
```

```
remote_login_passwordfile=EXCLUSIVE
compatible=11.2.0.1.0
#####################################################################

LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'

FAL_CLIENT = SHHAI
FAL_SERVER = BJING

STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT = '+GRP0/bjing', '+GRP1/shhai'
LOG_FILE_NAME_CONVERT = '+GRP1/bjing', '+GRP0/shhai'

#####################For Switch to Primary#####################
LOG_ARCHIVE_DEST_2 = 'SERVICE=BJING LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=BJING'
```

现在有了参数文件和备用数据库控制文件，就可以将备用数据库实例启动至 mount 状态。

```
SYS@SHHAI>create spfile from pfile='initshhai.ora';

File created.

SYS@SHHAI>startup mount
ORACLE instance started.

Total System Global Area  267227136 bytes
Fixed Size                  2174888 bytes
Variable Size             201326680 bytes
Database Buffers           58720256 bytes
Redo Buffers                5005312 bytes
Database mounted.
SYS@SHHAI>
SYS@SHHAI>select name from v$datafile;

NAME
--------------------------------------------------
```

```
+GRP1/shhai/datafile/system.256.836859253
+GRP1/shhai/datafile/sysaux.257.836859257
+GRP1/shhai/datafile/undotbs.258.836859509
+GRP1/shhai/datafile/example.260.836859607

SYS@SHHAI>select member from v$logfile;

MEMBER
--------------------------------------------------

+GRP0/shhai/onlinelog/group_1.257.836859253
+GRP0/shhai/onlinelog/group_2.258.836859253
+GRP0/shhai/onlinelog/group_3.259.836862919
+GRP0/shhai/onlinelog/group_4.260.836862927
+GRP0/shhai/onlinelog/group_5.261.836862933
```

此处，在加载备用数据库控制文件时，参数 DB_FILE_NAME_CONVERT 已经起了作用，它将控制文件中主数据库数据文件的路径信息已经更改。同样，可以验证参数 LOG_FILE_NAME_CONVERT 也已经起作用。

另外，此时在备用端的数据文件和联机日志文件并不存在，此处查询给出的信息只是控制文件中记录的信息，其中，数据文件的信息正是后面使用 RMAN 在备用数据库中执行数据文件还原的依据。

下面使用 RMAN 还原从主数据库备份的数据文件。

```
RMAN> connect target /

connected to target database: ORADB (DBID=2590124017, not
open)

RMAN> restore database;

Starting restore at 15-JAN-14
Starting implicit crosscheck backup at 15-JAN-14
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=8 device type=DISK
Crosschecked 6 objects
Finished implicit crosscheck backup at 15-JAN-14
```

```
Starting implicit crosscheck copy at 15-JAN-14
using channel ORA_DISK_1
Finished implicit crosscheck copy at 15-JAN-14

searching for all files in the recovery area
cataloging files...
no files cataloged

using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
+GRP1/shhai/datafile/system.256.836859253
channel ORA_DISK_1: restoring datafile 00002 to
+GRP1/shhai/datafile/sysaux.257.836859257
channel ORA_DISK_1: restoring datafile 00003 to
+GRP1/shhai/datafile/undotbs.258.836859509
channel ORA_DISK_1: restoring datafile 00004 to
+GRP1/shhai/datafile/example.260.836859607
channel ORA_DISK_1: reading from backup piece
+GFRA/bjing/backupset/2014_01_15/nnndf0_tag2014011
5t103301_0.261.836908381

channel ORA_DISK_1: piece handle=
+GFRA/bjing/backupset/2014_01_15/nnndf0_tag2014011
5t103301_0.261.836908381 tag=TAG20140
115T103301
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:25
Finished restore at 15-JAN-14
```

到此为止，物理备用数据库已经准备就绪，备用端拥有口令文件（单独创建或从主数据库端复制而来）、初始化参数文件、备用控制文件、数据文件。

需要特别说明的是，在 Oracle 11g 之前备用端的口令文件可以在备用端直接创建（sys 超级用户的口令必须相同），而从 Oracle 11g 开始，备用端的口令文件必须从主数据库端复制而来，不能直接创建。

5.4 启动日志传输和应用服务

5.4.1 主数据库端启动日志传输

在主数据库端设置 Data Guard 的必要参数，启动日志传输服务。

```
SYS@BJING>alter system set LOG_ARCHIVE_CONFIG = 'DG_
CONFIG=(BJING, SHHAI)' scope=both;

System altered.

SYS@BJING>alter system set LOG_ARCHIVE_DEST_2 =
2    'SERVICE=SHHAI LGWR ASYNC DB_UNIQUE_NAME=SHHAI'
scope=both;

System altered.

SYS@ORADB>alter system switch logfile;

System altered.
```

此时在主数据库端切换日志，检查日志是否能够正常传输到备用端。如果主数据库端参数 LOG_ARCHIVE_DEST_2 或备用数据库实例未启动，日志传输都会出现障碍，查询数据字典视图出现类似如下错误：

```
SYS@BJING>select status,error from v$archive_dest where dest_
id=1;

STATUS      ERROR
--------- --------------------------------------------------
ERROR       ORA-12514: TNS:listener does not currently know of
            service requested in connect descriptor
```

排除故障后，再次切换日志，如重做日志能够正常传输，则上述查询中的 ERROR 字段应为空。在没有创建备用重做日志并且没有设置 standby_archive_dest 参数的情况下，传输到备用端的日志默认情况下存储在归档日志的存储位置。

5.4.2 备用端启动 Redo 应用

日志传输问题解决后，即可在备用数据库端启动日志应用服务。

```
SYS@SHHAI>startup mount
ORACLE instance started.
```

```
Total System Global Area  167772160 bytes
Fixed Size                  1247876 bytes
Variable Size              58721660 bytes
Database Buffers          100663296 bytes
Redo Buffers                7139328 bytes
Database mounted.
SYS@SHHAI>alter database recover managed standby database
disconnect;

Database altered.
```

稍等片刻，检查日志备用数据库的最新日志序列号（已经应用的日志序列号），应该已经赶上主数据库的日志序列号。第一次执行此语句时，在启动日志应用服务的同时会生成物理备用数据库的联机日志文件。由于采用 OMF 管理，备用数据库端第一次启动日志应用会出现类似如下的错误，这是正常现象。此后 Oracle 会重新生成新的联机日志文件（注意联机日志的文件名有变化）。

```
ORA-00313: open failed for members of log group 1 of thread 1
ORA-00312: online log 1 thread 1:
 '+DG_PHY/phydb/onlinelog/group_1.257.803599043'
ORA-17503: ksfdopn:2 Failed to open file
+DG_PHY/phydb/onlinelog/group_1.257.803599043
ORA-15012: ASM file '+DG_PHY/phydb/onlinelog/
group_1.257.803599043'
does not exist

Clearing online redo logfile 1
+DG_PHY/phydb/onlinelog/group_1.257.803599043
Clearing online log 1 of thread 1 sequence number 27

SYS@SHHAI>select member from v$logfile;

MEMBER
-----------------------------------------------

+GRP0/shhai/onlinelog/group_1.265.836927579
+GRP0/shhai/onlinelog/group_2.266.836927579
+GRP0/shhai/onlinelog/group_3.262.836910029
+GRP0/shhai/onlinelog/group_5.263.836910029
```

```
+GRP0/shhai/onlinelog/group_4.264.836910029

SYS@SHHAI>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     51
Next log sequence to archive   0
Current log sequence           52

SYS@SHHAI>SELECT SEQUENCE#,APPLIED,FAL FROM V$ARCHIVED_LOG
  2  ORDER BY SEQUENCE#;

 SEQUENCE#      APPLIED        FAL
-----------------------------------------
      46           YES          YES
      47           YES          NO
      48           YES          NO
      49           YES          NO
      50           YES          NO
      51           YES          NO
```

5.4.3 数据更新测试

为了进一步验证物理备用数据库的工作状况，我们在主数据库中更新部分数据，提交并切换日志，之后检查备用数据库的数据更新情况。

```
SYS@ORADB>connect scott/tiger
Connected.
SCOTT@ORADB>select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
      10   ACCOUNTING     NEW YORK
      20   RESEARCH       DALLAS
      30   SALES          CHICAGO
      40   OPERATIONS     BOSTON

SCOTT@ORADB>insert into dept
```

```
  2  select deptno+5,loc,dname from dept;

4 rows created.

SCOTT@ORADB>commit;

Commit complete.

SCOTT@ORADB>conn / as sysdba
Connected.
SYS@ORADB>alter system switch logfile;

System altered.
```

在备用数据库端，停止 Redo 应用服务，然后以只读方式打开数据库，并检查数据更新情况。在第一次以只读方式打开物理备用数据库时，会生成临时表空间的数据文件。至此，物理备用数据库的数据库文件就完全齐全了，包括备用控制文件、数据文件、临时数据文件和联机日志文件。

```
SYS@SHHAI>alter database recover managed standby database
cancel;

Database altered.

SYS@SHHAI>alter database open read only;

Database altered.

SYS@SHHAI>select * from scott.dept;

 DEPTNO          DNAME        LOC
---------- -------------- -------------
   15            NEW YORK     ACCOUNTING
   25            DALLAS       RESEARCH
   35            CHICAGO      SALES
   45            BOSTON       OPERATIONS
   10            ACCOUNTING   NEW YORK
   20            RESEARCH     DALLAS
   30            SALES        CHICAGO
   40            OPERATIONS   BOSTON
```

```
8 rows selected.
```

由此可见，在主数据库中更新的数据已经顺利地传播到备用数据库中。注意：物理备用数据库应该以只读方式打开。至此，构造物理备用数据库的 Data Guard 环境告一段落。

5.5 DG 后续配置的考虑

根据前面介绍的 Data Guard 的配置，当前物理备用数据库为主数据库提供最大性能模式（MAXIMUM PERFORMANCE）的保护，此为默认保护模式，后续可以根据应用环境的需要调整为另外两种保护模式：最大可用（MAXIMUM AVAILABILITY）模式、最大保护模式（MAXIMUM PROTECTION）。

检查主备用数据库的角色与保护模式（级别），当前 Data Guard 将主数据库置于“最大性能”保护状态。

```
SYS@ORADB>SELECT NAME, DATABASE_ROLE, PROTECTION_MODE
  2  FROM V$DATABASE;

NAME      DATABASE_ROLE    PROTECTION_MODE
--------- ---------------- --------------------
ORADB     PRIMARY          MAXIMUM PERFORMANCE

SYS@SHHAI>SELECT NAME, DATABASE_ROLE, PROTECTION_MODE
  2  FROM V$DATABASE;

NAME        DATABASE_ROLE       PROTECTION_MODE
--------- ---------------- --------------------
ORADB       PHYSICAL STANDBY    MAXIMUM PERFORMANCE
```

另外，此时的物理备用数据库的数据库级别的闪回功能并没有启动（不管主数据库是否启动数据库闪回），可以考虑启动备用端的数据库闪回功能，这样物理备用数据库在需要时可以手工方式以读 / 写模式打开，当完成必要的读 / 写测试后，可以再次闪回到过去的物理备用状态，这会给我们灵活运用物理备用数据库提供极大的方便。

对于闪回数据库的功能，在 Data Guard 环境中建议如下：为了最大限度地减少主数据库的磁盘 I/O（特别是在 DML 比较频繁的数据库系统），在主数据库中关闭闪回数据库功能，而在物理备用数据库中启动该功能，这样既可避免闪回日志带来的额外 I/O，又可充分利用闪回数据库带来的便利。

第6章

构建逻辑备用数据库

本章介绍逻辑备用数据库的构建过程。与物理备用数据库不同，逻辑备用数据库并不是主数据库的精确副本，从最终用户的角度来看，逻辑备用数据库中的数据和主数据库中的数据是基本一致的，而两者在数据库的物理存储结构是有可能不同的，通俗地说是容器不同，但里面装的东西是尽可能相同的。正因为如此，逻辑备用数据库的表中的记录可能会有不同的物理存储地址，如果根据行地址 ROWID 去搜索记录，在主数据库中和在备用数据库中往往会得出不同的结果。

6.1 对主数据库的检查

逻辑备用数据库是通过接收主数据库传送的 Redo Data，并从中挖掘出对应的 SQL 语句来实现备用数据库的数据更新。在这个过程中，目前还存在着一些限制条件，在实施逻辑备用数据库的过程中需要特别注意。

6.1.1 检查不支持的模式

逻辑备用数据库是从用户的角度来更新数据的，备用端日志的应用过程与应用系统更新数据库数据的情形非常相似，在这个过程中通常不会涉及 Oracle 的内部模式，如 SYS、System、DBSNMP 模式等，这些模式是数据库系统自身使用的。典型的情况是 SYS 用户下的数据字典信息，这些信息是数据库内部的元数据（Metadata），它们是随数据库管理指令或 DDL 指令而更新的。

执行下面的查询可以查看逻辑备用不支持的模式信息：

```
SYS@BJING>select owner,statement_opt from dba_logstdby_skip
  2  where statement_opt='INTERNAL SCHEMA';

OWNER                          STATEMENT_OPT
-------------------------------------------------------
OUTLN                          INTERNAL SCHEMA
```

```
SYS                              INTERNAL SCHEMA
SYSTEM                           INTERNAL SCHEMA
BI                               INTERNAL SCHEMA
ANONYMOUS                        INTERNAL SCHEMA
MDSYS                            INTERNAL SCHEMA
ORDSYS                           INTERNAL SCHEMA
EXFSYS                           INTERNAL SCHEMA
DBSNMP                           INTERNAL SCHEMA
APPQOSSYS                        INTERNAL SCHEMA
ORDDATA                          INTERNAL SCHEMA
XDB                              INTERNAL SCHEMA
ORDPLUGINS                       INTERNAL SCHEMA
SI_INFORMTN_SCHEMA               INTERNAL SCHEMA
ORACLE_OCM                       INTERNAL SCHEMA
XS$NULL                          INTERNAL SCHEMA
DIP                              INTERNAL SCHEMA

17 rows selected.
```

由此，要通过逻辑备用数据库实现用户数据的同步更新，任何用户数据不能位于这些模式下，因为在备用端的 SQL Apply 过程中，Data Guard 会忽略对这些模式更新的 SQL 指令，这一点需要特别注意。

6.1.2 检查不支持的用户表

逻辑备用数据库不支持某些特定的用户表，通常是因为数据类型的原因，逻辑备用数据库目前还不能支持所有的数据类型，如对象 Object 类型、可变数组 VARRAY 类型、外部文件指针 BFILE 类型等都会被 SQL Apply 忽略，不支持的数据类型与 Oracle 的版本有关。

测试数据库是一个安装了示例模式的 Oracle 11.2 版本的数据库。执行下面的查询可以了解哪些用户下的哪些表在备用数据库端不会被更新：

```
SYS@BJING>select distinct owner, table_name
  2  from dba_logstdby_unsupported order by owner,table_name;

OWNER                            TABLE_NAME
-------------------------------------------------------
IX                               AQ$_ORDERS_QUEUETABLE_G
IX                               AQ$_ORDERS_QUEUETABLE_H
IX                               AQ$_ORDERS_QUEUETABLE_I
```

```
IX                             AQ$_ORDERS_QUEUETABLE_L
IX                             AQ$_ORDERS_QUEUETABLE_S
IX                             AQ$_ORDERS_QUEUETABLE_T
IX                             AQ$_STREAMS_QUEUE_TABLE_C
IX                             AQ$_STREAMS_QUEUE_TABLE_G
IX                             AQ$_STREAMS_QUEUE_TABLE_H
IX                             AQ$_STREAMS_QUEUE_TABLE_I
IX                             AQ$_STREAMS_QUEUE_TABLE_L
IX                             AQ$_STREAMS_QUEUE_TABLE_S
IX                             AQ$_STREAMS_QUEUE_TABLE_T
IX                             ORDERS_QUEUETABLE
IX                             STREAMS_QUEUE_TABLE
OE                             CATEGORIES_TAB
OE                             CUSTOMERS
OE                             PURCHASEORDER
OE                             WAREHOUSES
PM                             ONLINE_MEDIA
PM                             PRINT_MEDIA
SH                             DIMENSION_EXCEPTIONS

22 rows selected.
```

针对某个不支持的表，我们可以继续了解这些不支持的表是由哪些字段不支持的数据类型导致的，为此执行下面的查询（以 OE 模式下的 CUSTOMERS 表为例）：

```
SYS@BJING>select column_name, data_type
  2  from dba_logstdby_unsupported
  3  where table_name='CUSTOMERS' and owner='OE';

COLUMN_NAME                    DATA_TYPE
-----------------------------------------------------
CUST_ADDRESS                   OBJECT
PHONE_NUMBERS                  VARRAY
CUST_GEO_LOCATION              OBJECT
```

从上例查询的结果看，CUSTOMERS 表中有 3 个字段导致该表不被支持，它们的数据类型分别是 Object 类型和 Varray 类型。进一步查看，字段 CUST_GEO_LOCATION 的具体数据类型是 MDSYS.SDO_GEOMETRY:

```
SYS@BJING>desc MDSYS.SDO_GEOMETRY
 Name                          Null?    Type
```

```
-----------------------------------------------------
 SDO_GTYPE                                  NUMBER
 SDO_SRID                                   NUMBER
 SDO_POINT                        MDSYS.SDO_POINT_TYPE
 SDO_ELEM_INFO                    MDSYS.SDO_ELEM_INFO_ARRAY
 SDO_ORDINATES                    MDSYS.SDO_ORDINATE_ARRAY
...
```

继续追踪其中的一个数据类型，如 SDO_ELEM_INFO_ARRAY 类型，发现该类型是可变数组类型，这正是逻辑备用不支持的数据类型之一。

```
SYS@BJING>desc SDO_ELEM_INFO_ARRAY
 SDO_ELEM_INFO_ARRAY VARRAY(1048576) OF NUMBER
```

6.1.3 检查存在唯一性问题的表

存在唯一性问题的表大多数可以得到逻辑备用数据库的支持，只是这些表中的数据更新需要增加重做日志的量（即补充日志 Supplemental Log），并增加逻辑备用数据库 SQL Apply 的负担。原因是：当用户更新表中某一条记录时，重做日志中会创建一个对应的数据项以唯一地标识被改变的记录。最有效的方式是使用行的物理地址 ROWID 来标识行数据的变更，这在主数据库中也是非常有效的，但这样的重做日志信息在逻辑备用数据库中用于重演数据的更新是无效的，因为逻辑备用数据库的表行 ROWID 已经不是主数据库的表行 ROWID 了。在 ROWID 失效的情况下，如果该表中存在主键或唯一性约束（唯一性索引），Oracle 同样可以利用主键或唯一性约束来唯一地标识改变的行，这就是基于主键或唯一性约束的补充日志。如果表中既没有主键也没有唯一性约束，若要在逻辑备用数据库中实现正常更新，Oracle 就需要在日志中使用所有字段的值来标识更新的行，这就是所谓的全补充日志（SUPPLEMENTAL_LOG_DATA_ALL）。但这种情况下如果表中包含 LONG 类型列或 LOB 类型列，即使使用全补充日志，Oracle 也不能保证在逻辑备用数据库中实现正常更新，这就是下面的查询 BAD_COLUMN='Y' 的含义。

查询当前数据库中存在唯一性问题的表，代码如下：

```
SYS@BJING>select * from dba_logstdby_not_unique;

OWNER            TABLE_NAME                      BAD_COLUMN
-----------------------------------------------------------
TSMSYS           SRS$                            Y
OE               INVENTORIES                     N
SH               SUPPLEMENTARY_DEMOGRAPHICS      N
SCOTT            BONUS                           N
```

```
SCOTT            SALGRADE                       N
SH               COSTS                          N
SH               SALES                          N
HR               COUNTRIES                      N

8 rows selected.
```

对于上述查询给出的表，若在其中添加非依赖主键约束（Disabled primary-key RELY Constraint），可以消除该表的唯一性问题。例如：

```
SCOTT@NETDB>ALTER TABLE bonus ADD CONSTRAINT pk_ename
  2  PRIMARY KEY (ename) RELY DISABLE;

Table altered.

SH@BJING>select constraint_name,constraint_type,rely
2  from user_constraints
  3  where table_name='SUPPLEMENTARY_DEMOGRAPHICS';

CONSTRAINT_NAME                C RELY
-------------------------------------------------------
SYS_C005011                    C
SUPP_DEMO_PK                   P

SH@BJING>alter table SUPPLEMENTARY_DEMOGRAPHICS
  2  modify constraint SUPP_DEMO_PK rely disable;

Table altered.

SH@BJING>select constraint_name,constraint_type,rely
2  from user_constraints
  3  where table_name='SUPPLEMENTARY_DEMOGRAPHICS';

CONSTRAINT_NAME                C RELY
-------------------------------------------------------
SYS_C005011                    C
SUPP_DEMO_PK                   P RELY

SH@BJING>conn / as sysdba
Connected.
```

```
SYS@BJING>select * from dba_logstdby_not_unique;

no rows selected
```

非依赖主键约束可以消除唯一性问题，但不会在主数据库中产生真正的约束开销。这里需要特别注意的是，非依赖主键约束的真正含义是：在逻辑备用数据库中，Oracle 使用非依赖主键约束中指定的字段（也许是多个字段）去唯一地标识更新的记录。如果应用软件不能保证非依赖主键约束的字段取值的唯一性，则在逻辑备用数据库中，Oracle 就不能正确地更新数据。

默认情况下，重做日志中记录了 ROWID 信息用于基于日志数据恢复。在逻辑备用数据库中，ROWID 会失效，因此，为了使得逻辑备用数据库能够正常更新数据，在主数据库的重做日志中就必须记录额外的信息用于唯一地标识被更新的记录，这就是补充日志（Supplemental Log）。为了使得 Oracle 在逻辑备用数据库中实现正常的数据更新，DBA 必须在主数据中启动基于主键和唯一性约束的补充日志。

```
SYS@BJING>alter database add supplemental log
  2  data(primary key,unique index) columns;

Database altered.

SYS@BJING>select
  2  supplemental_log_data_min min,
  3  supplemental_log_data_pk pk,
  4  supplemental_log_data_ui ui from v$database;

MIN      PK  UI
--------------------------------------------------
IMPLICIT YES YES
```

至此，为构建逻辑备用数据库，在主数据库中的检查工作完毕。

6.2 准备物理备用数据库

构建逻辑备用数据库是从构建物理备用数据库开始的，即逻辑备用数据库是从物理备用数据库转化而来的，先构造物理备用数据库，然后将其转化为逻辑备用数据库。

我们计划按照如表 6-1 所示的内容构建 Data Guard 环境。数据库名称为 ORADB 的主数据库运行在北京，在上海配置一个物理备用数据库（第 5 章已完成），接下来在济南配置一个逻辑备用数据库。

此处按照前面章节构建物理备用数据库的步骤对 ORADB 再次构建一个物理备用数据库（最后要将其转换为逻辑备用数据库），其实例名称取 JINAN，针对该实例的网络服务名同样取 JINAN。

表 6-1　主备用数据库及其网络服务

DG ROLE	DB_NAME	INSTANCE_NAME	DB_UNIQUE_NAME	TNS NAME
主数据库	ORADB	BJING	BJING	BJING
物理备用	ORADB	SHHAI	SHHAI	SHHAI
逻辑备用	ORADB	JINAN	JINAN	JINAN

备注：逻辑备用数据库是一个独立数据库，其数据库名（DB_NAME）可以与主数据库不同。

6.2.1　主备用数据库参数的准备

下面首先调整主数据库的相关参数，修改部分如下（原有一个物理备用数据库 SHHAI）：在 LOG_ARCHIVE_CONFIG 参数中增加 Data Guard 环境中的 DB_UNIQUE_NAME，增加一个新的网络归档 LOG_ARCHIVE_DEST_2 参数。

```
########################################################
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI, JINAN)'
LOG_ARCHIVE_DEST_2 = 'SERVICE=SHHAI LGWR ASYNC DB_UNIQUE_
NAME=SHHAI'
LOG_ARCHIVE_DEST_3 = 'SERVICE=JINAN LGWR ASYNC DB_UNIQUE_
NAME=JINAN'
########################################################
```

在原有物理备用数据库 SHHAI 初始化参数的基础上准备 JINAN 实例的初始化参数，修改的部分如下（作为测试，将逻辑备用数据库的物理存储统一存放于 ASM 磁盘组 GFRA）：

```
########################################################
db_name=ORADB
db_unique_name=JINAN
instance_name=JINAN
...

db_create_file_dest='+GFRA'
db_create_online_log_dest_1='+GFRA'
db_recovery_file_dest='+GFRA'
db_recovery_file_dest_size=2G
log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
```

```
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI, JINAN)'

FAL_CLIENT = JINAN
FAL_SERVER = BJING, SHHAI

STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT = '+GRP0/bjing', '+GFRA/jinan'
LOG_FILE_NAME_CONVERT = '+GRP1/bjing', '+GFRA/jinan'
#####################################################
```

上述重新设置了参数 instance_name 和 db_unique_name，调整了 OMF 的存储路径（使用磁盘组 GFRA），暂时性地注释了控制文件参数，同样，在 LOG_ARCHIVE_CONFIG 参数中增加 Data Guard 环境中新的 DB_UNIQUE_NAME，更新了 FAL_CLIENT 参数，修改了由主数据库到新的物理备用数据库的路径转换参数 DB_FILE_NAME_CONVERT 和 LOG_FILE_NAME_CONVERT，接下来需要根据新的参数文件启动 JINAN 实例。

6.2.2 物理存储的准备

对主数据库执行一次最新的备份，创建备用控制文件。注意：此处的操作顺序，先执行对主数据库执行物理备份，再创建备用控制文件，这样，备用控制文件中会包含最新的备份信息，以便在备用数据库（加载状态）中利用主数据库中的备份执行还原。

```
SYS@BJING>host rman target /

Recovery Manager: Release 11.2.0.1.0 - Production on Thu Jan
16 10:56:08 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: ORADB (DBID=2590124017)

RMAN> backup database plus archivelog;

Starting backup at 16-JAN-14
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=11 device type=DISK
```

```
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=151 RECID=214
STAMP=836933070
......
input archived log thread=1 sequence=179 RECID=267
STAMP=836996185
channel ORA_DISK_1: starting piece 1 at 16-JAN-14
channel ORA_DISK_1: finished piece 1 at 16-JAN-14
piece handle=+GFRA/bjing/backupset/2014_01_16/annnf0_tag201401
16t105627_0.490.836996189 tag=TAG20140116T105627 comment=N
ONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:25
Finished backup at 16-JAN-14

Starting backup at 16-JAN-14
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=
+GRP0/bjing/datafile/system.256.836859253
input datafile file number=00003 name=
+GRP0/bjing/datafile/undotbs.258.836859509
input datafile file number=00002 name=
+GRP0/bjing/datafile/sysaux.257.836859257
input datafile file number=00004 name=
+GRP0/bjing/datafile/example.260.836859607
channel ORA_DISK_1: starting piece 1 at 16-JAN-14
channel ORA_DISK_1: finished piece 1 at 16-JAN-14
piece handle=
+GFRA/bjing/backupset/2014_01_16/nnndf0_tag2014011
6t105654_0.489.836996215 tag=TAG20140116T105654 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:55
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
```

```
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 16-JAN-14
channel ORA_DISK_1: finished piece 1 at 16-JAN-14
piece handle=
+GFRA/bjing/backupset/2014_01_16/ncsnf0_tag2014011
6t105654_0.486.836996271 tag=TAG20140116T105654 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 16-JAN-14

Starting backup at 16-JAN-14
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=180 RECID=268
STAMP=836996272
channel ORA_DISK_1: starting piece 1 at 16-JAN-14
channel ORA_DISK_1: finished piece 1 at 16-JAN-14
piece handle=
+GFRA/bjing/backupset/2014_01_16/annnf0_tag2014011
6t105752_0.482.836996273 tag=TAG20140116T105752 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 16-JAN-14

RMAN> backup current controlfile for standby;

Starting backup at 16-JAN-14
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including standby control file in backup set
channel ORA_DISK_1: starting piece 1 at 16-JAN-14
channel ORA_DISK_1: finished piece 1 at 16-JAN-14
piece handle=
+GFRA/bjing/backupset/2014_01_16/ncnnf0_tag2014011
6t110317_0.479.836996599 tag=TAG20140116T110317 comment=NONE
```

```
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 16-JAN-14
```

或者在 SQL*Plus 中创建一个中间的备用数据库的控制文件：

```
SYS@BJING>alter database create standby controlfile
  2  as '/database/logdb/control2.ctl';

Database altered.
```

6.2.3　备用实例的启动与还原

用新的物理备用参数文件启动实例，使用 RMAN 还原备用数据库控制文件。修改实例 LOGDB 参数 control_files 指向新还原的控制文件，并将实例转换为 mount 状态。

```
SYS@JINAN>create spfile from pfile='initjinan.ora';

File created.

SYS@JINAN>startup nomount
ORACLE instance started.

Total System Global Area  534462464 bytes
Fixed Size                  2177456 bytes
Variable Size             322963024 bytes
Database Buffers          201326592 bytes
Redo Buffers                7995392 bytes
SYS@JINAN>host rman target /

Recovery Manager: Release 11.2.0.1.0 - Production on Thu Jan
16 11:05:43 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: ORADB (not mounted)

RMAN> restore standby controlfile
from '+GFRA/bjing/backupset/2014_01_16/
ncnnf0_tag20140116t110317_0.479.836996599';
```

```
Starting restore at 16-JAN-14
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=5 device type=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:04
output file name=+GFRA/jinan/controlfile/current.258.836996795
Finished restore at 16-JAN-14
```

注意，此处主数据库控制文件的备份还原生成的备用数据库的控制文件为+GFRA/jinan/controlfile/current.258.836996795，接下来修改实例JINAN的control_files参数指向，并启动实例的mount状态。

```
SYS@JINAN>show parameters control_files

NAME                                 TYPE                 VALUE
------------------------------------------------------------------
control_files                        string
+GFRA/jinan/controlfile/current.258.836996795
SYS@JINAN>alter database mount;

Database altered.
```

使用RMAN连接至使用备用控制文件加载的实例JINAN，还原之前备份的主数据库的数据文件。

```
SYS@JINAN>host rman target /

Recovery Manager: Release 11.2.0.1.0 - Production on Thu Jan
16 11:13:47 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: ORADB (DBID=2590124017, not
open)

RMAN> restore database;

Starting restore at 16-JAN-14
Starting implicit crosscheck backup at 16-JAN-14
```

```
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=9 device type=DISK
Crosschecked 11 objects
Finished implicit crosscheck backup at 16-JAN-14

Starting implicit crosscheck copy at 16-JAN-14
using channel ORA_DISK_1
Finished implicit crosscheck copy at 16-JAN-14

searching for all files in the recovery area
cataloging files...
no files cataloged

using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
+GFRA/jinan/datafile/system.256.836859253
channel ORA_DISK_1: restoring datafile 00002 to
+GFRA/jinan/datafile/sysaux.257.836859257
channel ORA_DISK_1: restoring datafile 00003 to
+GFRA/jinan/datafile/undotbs.258.836859509
channel ORA_DISK_1: restoring datafile 00004 to
+GFRA/jinan/datafile/example.260.836859607
channel ORA_DISK_1: reading from backup piece
+GFRA/bjing/backupset/2014_01_16/nnndf0_tag2014011
6t105654_0.489.836996215
channel ORA_DISK_1: piece handle=
+GFRA/bjing/backupset/2014_01_16/nnndf0_tag2014011
6t105654_0.489.836996215 tag=TAG20140116T105654
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:55
Finished restore at 16-JAN-14
```

另外，物理备用数据库物理存储的产生也可以使用 RMAN 复制（duplicate）和辅助数据库实例（auxiliary）的方法，这里的辅助数据库实例即备用数据库实

例 JINAN。对主数据库进行数据库备份、创建备用控制文件，在备用端启动实例至 nomount 状态后，即可使用 RMAN 的复制方法来创建物理备用数据库。

下面给出一次复制实验的样本输出：

```
RMAN target sys/internal@netdb
RMAN> backup database;
RMAN> backup current controlfile for standby;

RMAN target sys/internal@netdb auxilary /
Recovery Manager: Release 11.2.0.1.0
- Production on Sun Feb 3 14:01:41 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.
All rights reserved.

connected to target database: NETDB (DBID=2992910526)
connected to auxiliary database: NETDB (not mounted)

RMAN> duplicate target database for standby;

Starting Duplicate Db at 03-FEB-13
using target database control file instead of recovery catalog
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=10 device type=DISK

contents of Memory Script:
{
   restore clone standby controlfile;
}
executing Memory Script

Starting restore at 03-FEB-13
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: reading from backup piece
+DG_FRA/NETDB/BACKUPSET/2013_02_02/O1_MF_NCNNF_
TAG20130202T153340_8JSJBODP_.BKP
```

```
channel ORA_AUX_DISK_1: piece handle=
+DG_FRA/NETDB/BACKUPSET/2013_02_02/O1_MF_NCNNF_
TAG20130202T153340_8JSJBODP_.BKP tag=TAG20130202T153340
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time:
00:00:01
output file name=+DG_DATA/LOGDB/CTL1LOGDB.ORA
Finished restore at 03-FEB-13

contents of Memory Script:
{
   sql clone 'alter database mount standby database';
}
executing Memory Script

sql statement: alter database mount standby database

contents of Memory Script:
{
   set newname for tempfile  1 to
 "+DG_DATA/LOGDB/TEMPNETDB.ORA";
   switch clone tempfile all;
   set newname for datafile  1 to
 "+DG_DATA/LOGDB/SYS1NETDB.DBF";
   set newname for datafile  2 to
 "+DG_DATA/LOGDB/AUX1NETDB.DBF";
   set newname for datafile  3 to
 "+DG_DATA/LOGDB/EMP1NETDB.DBF";
   set newname for datafile  4 to
 "+DG_DATA/LOGDB/UNDONETDB.ORA";
   restore
   clone database
   ;
}
executing Memory Script

executing command: SET NEWNAME
```

```
renamed tempfile 1 to +DG_DATA/LOGDB/TEMPNETDB.ORA in control
file

executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME

Starting restore at 03-FEB-13
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
 +DG_DATA/LOGDB/SYS1NETDB.DBF
channel ORA_AUX_DISK_1: restoring datafile 00002 to
 +DG_DATA/LOGDB/AUX1NETDB.DBF
channel ORA_AUX_DISK_1: restoring datafile 00003 to
 +DG_DATA/LOGDB/EMP1NETDB.DBF
channel ORA_AUX_DISK_1: restoring datafile 00004 to
 +DG_DATA/LOGDB/UNDONETDB.ORA
channel ORA_AUX_DISK_1: reading from backup piece
+DG_FRA/NETDB/BACKUPSET/2013_02_02/O1_MF_NNNDF_
TAG20130202T151905_8JSHHB3F_.BKP
channel ORA_AUX_DISK_1: piece handle=
+DG_FRA/NETDB/BACKUPSET/2013_02_02/O1_MF_NNNDF_
TAG20130202T151905_8JSHHB3F_.BKP tag=TAG20130202T151905
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time:
00:00:15
Finished restore at 03-FEB-13

contents of Memory Script:
{
   switch clone datafile all;
}
executing Memory Script
```

```
datafile 1 switched to datafile copy
input datafile copy RECID=1 STAMP=806421732
file name=+DG_DATA/LOGDB/SYS1NETDB.DBF
datafile 2 switched to datafile copy
input datafile copy RECID=2 STAMP=806421732
file name=+DG_DATA/LOGDB/AUX1NETDB.DBF
datafile 3 switched to datafile copy
input datafile copy RECID=3 STAMP=806421732
file name=+DG_DATA/LOGDB/EMP1NETDB.DBF
datafile 4 switched to datafile copy
input datafile copy RECID=4 STAMP=806421732
file name=+DG_DATA/LOGDB/UNDONETDB.ORA
Finished Duplicate Db at 03-FEB-13
```

6.2.4　同步物理备用数据库

在启动主数据库的日志传输和备用端 JINAN 的日志应用之前需要配置好网络服务，此处两端指向主备用数据库的网络服务名分别取为 BJING 和 JINAN，并分别测试成功。

```
SYS@jinan>alter database recover managed standby database
  2  using current logfile disconnect;

Database altered.
```

逻辑备用数据库是由物理备用数据库转换而来的，这里需要等待物理备用数据库经过托管的介质恢复与主数据库完全同步后，取消托管介质恢复，为将其转换为逻辑备用数据库做好准备。

```
SYS@LOGDB>alter database recover managed standby database
cancel;

Database altered.
```

6.3　激活逻辑备用数据库

Data Guard 中物理备用数据库要转换为逻辑备用数据库需要进行如下几个关键的处理过程：

（1）在主数据库的重做日志流中构建 LogMiner 数据字典。

（2）执行由物理备用到逻辑备用的转换，解除物理备用的托管状态。

（3）以 Resetlogs 方式打开，重置日志序列号，产生新的数据库灵魂 Incarnation。

6.3.1 构建 LogMiner 字典

逻辑备用数据库在对重做数据进行日志挖掘 LogMiner 时，需要额外的数据字典信息以便能够将重做数据转换为相应的 SQL 语句。为此，需要在主数据库中执行下面的操作，为 LogMiner 构建必要的元数据：

```
SYS@BJING>exec dbms_logstdby.build;
PL/SQL procedure successfully completed.

SYS@BJING>alter system switch logfile;
System altered.
```

该过程执行完毕后，最好手动切换一次日志，以确保产生的 LogMiner 字典信息传到逻辑备用端。执行该过程 Oracle 主要实现如下工作：

（1）在主数据库上启动补充日志记录，这相当于执行下面的指令。

```
alter database add supplemental log
data(primary key,unique index) columns;
```

（2）构建主数据库元数据的 LogMiner 字典，在日志流中增加此信息，以便逻辑备用端了解如何在重做数据中挖掘出对应的 SQL 语句。

（3）标记物理备用数据库 MRP 进程应用的最后事务（SCN）。

注意，执行 dbms_logstdby.build 过程需要等待当前所有更新事务结束才能返回完成，此时才能确定 MRP 进程需要应用的最后事务。这个构建过程期间启动的任何事务是完成逻辑备用数据库转换后 SQL Apply 需要应用的事务。从此，逻辑数据库端接收到的每个日志文件中都会包含 LogMiner 字典信息，这是由重做数据到 SQL 转换所必需的。

6.3.2 Standby 类型的转换

接下来就可以在备用端将物理备用数据库转换为逻辑备用数据库（注意，最后的标识由参数 DB_NAME 决定）。事实上，在使用 SPFILE 的情况下，该指令最后的数据库名可以重新定义，当指令执行完毕，逻辑数据库的数据库名也一并得到修改（SPFILE 中的 DB_NAME 参数被自动更改为此处设置的值）。

```
SYS@LOGDB>alter database recover to logical standby ORADB;
Database altered.
```

或者，在转换过程中将逻辑备用数据库名一并改为 JINAN：

```
SYS@LOGDB>alter database recover to logical standby JINAN;
Database altered.
```

执行上述指令过程中，告警日志文件中产生类似的输出：

```
alter database recover to logical standby LOGDB
Thu Jan 31 00:27:59 2013
Managed Standby Recovery not using Real Time Apply
Media Recovery Waiting for thread 1 sequence 15 (in transit)
```

该语句等待应用重做数据，直到在日志文件中找到 LogMiner 字典。这一般需要耗费一段时间，具体时间的长短依赖于在主数据库中执行 dbms_logstdby.build 生成的重做多久才能传送到备用数据库，以及需要应用多少重做数据库。如果字典建立没能在主数据库上成功执行，这条命令将永远不会完成。此时，如果出现长时间等待，说明 Oracle 在等待 LogMiner 字典，为了使得该内容及时传递到备用端，可以在主数据库端执行人工的日志切换以完成该过程。

```
SYS@JINAN>alter database recover managed standby database
cancel;

Database altered.

SYS@JINAN>select dbid, database_role from v$database;

      DBID DATABASE_ROLE
---------- ----------------
2590124017 PHYSICAL STANDBY

SYS@JINAN>alter database recover to logical standby ORADB;

Database altered.

SYS@JINAN>shutdown
ORA-01507: database not mounted
ORACLE instance shut down.

SYS@JINAN>startup
ORACLE instance started.

Total System Global Area  534462464 bytes
Fixed Size                  2177456 bytes
```

```
Variable Size             348128848 bytes
Database Buffers          176160768 bytes
Redo Buffers                7995392 bytes
Database mounted.
ORA-01589: must use RESETLOGS or NORESETLOGS option for
database open

SYS@JINAN>alter database open resetlogs;

Database altered.

SYS@JINAN>select dbid, database_role from v$database;

      DBID  DATABASE_ROLE
---------- ----------------
2590233129  LOGICAL STANDBY

SYS@JINAN>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      1
Next log sequence to archive    1
Current log sequence                  1
```

由此可见，逻辑备用数据库和主数据库相比，已经是完全不同的数据库了（DBID 发生了变化，即数据库的 Incarnation 已经不同）。正因为如此，在由物理备用到逻辑备用的转换过程中，可以重新指定数据库名称（DB_NAME）。另外，经过转换后的数据库必须带 Noresetlogs 选项打开，以重新生成新的数据库 Incarnation（灵魂）。

与之相对照，物理备用数据库本质上是和主数据库是同一数据库（DBID 相同），应此物理备用数据库上的物理备份可以用于对主数据库的介质恢复，而逻辑备用数据库的物理备份不能用于对主数据库的介质恢复。

6.3.3 重置新的 Incarnation

至此，物理备用到逻辑备用的转换成功。在物理备用数据库期间，数据库一直处于托管恢复的状态，而逻辑备用数据库需要处于打开状态。正像数据库经过介质恢复后需要带选项 Resetlogs 打开一样，此时的逻辑备用数据库要打开，也

需要经过同样的处理。如果是 RAC 数据库，此时需要以单实例打开（cluster_database=false）。

备用数据库从此结束托管恢复的状态，逻辑备用数据库应用由此开始，我们可以明确地看到日志序列号从 1 开始。实际上，逻辑备用数据库端的针对重做数据的 SQL Apply 过程相当于一个特殊的应用软件持续对数据库进行数据更新。从 Incarnation 的角度，逻辑备用数据库 DBID 发生变化，因此，是一个完全不同的数据库。如果需要，也可将逻辑备用数据库更改为不同的数据库名。

当前逻辑数据库及其实例的名称设置如下：

```
DB_NAME=ORADB
INSTANCE_NAME=JINAN
DB_UNIQUE_NAME=JINAN
```

如果在由物理备用数据库转换为逻辑备用数据库的过程中，没有变更数据库名称（DB_NAME），则可以单独对数据库进行更名。为了避免与主数据库的混淆，可将数据库名由 NETDB 也更改为 LOGDB，这是可选的步骤，示例如下：

```
nid TARGET=sys/internal dbname=LOGDB SETNAME=Y

DBNEWID: Release 10.2.0.1.0 - Production on Mon Nov 5 20:08:49
2012

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to database NETDB (DBID=2551859953)

Connected to server version 10.2.0

Control Files in database:
    +DG_LOG/logdb/controlfile/backup.257.798567197

Change database name of database NETDB to LOGDB? (Y/[N]) => Y

Proceeding with operation
Changing database name from NETDB to LOGDB
Control File +DG_LOG/logdb/controlfile/backup.257.798567197
- modified
Datafile +DG_LOG/logdb/datafile/system.258.798568053
- wrote new name
```

```
Datafile +DG_LOG/logdb/datafile/undotbs.261.798568055
- wrote new name
Datafile +DG_LOG/logdb/datafile/sysaux.259.798568053
- wrote new name
Datafile +DG_LOG/logdb/datafile/example.260.798568055
- wrote new name
Datafile +DG_LOG/logdb/datafile/users.262.798568055
- wrote new name
Datafile +DG_LOG/logdb/tempfile/temp.265.798576901
- wrote new name
Control File +DG_LOG/logdb/controlfile/backup.257.798567197
- wrote new name
    Instance shut down

Database name changed to LOGDB.
Modify parameter file and generate a new password file before
restarting.
Succesfully changed database name.
DBNEWID - Completed succesfully.
```

6.3.4 启动 SQL Apply

这是最后一步，启动针对重做数据流的 SQL 应用，即在逻辑数据库端启动 LSP 进程。

```
SYS@LOGDB>alter database start logical standby apply;
Database altered.
```

执行此指令后，Oracle 将启动逻辑备用数据库特有的进程 LSP，开始挖掘由主数据库端发送过来的重做数据。当前数据库处在最高性能保护模式，当主数据库执行日志切换时，重做数据开始传入逻辑备用端。在测试时，不要期望主数据库中的变更会立即传入逻辑备用数据库，因为从重做数据接收、日志挖掘到 SQL 应用还需要一个过程，即使是实时应用也需要一个较短的时间段。

为了了解执行该指令时 Oracle 的内部活动，特截取由该指令执行后产生的部分告警日志内容，列举如下：

```
alter database start logical standby apply
Wed Nov 28 14:33:04 2012
No optional part
Attempt to start background Logical Standby process
```

```
LSP0 started with pid=21, OS id=2024
Wed Nov 28 14:33:05 2012
Completed: alter database start logical standby apply
Wed Nov 28 14:33:07 2012
LOGSTDBY status: ORA-16111: log mining and apply setting up
Wed Nov 28 14:33:07 2012
LOGMINER: Parameters summary for session# = 1
LOGMINER: Number of processes = 3, Transaction Chunk Size =
201
LOGMINER: Memory Size = 30M, Checkpoint interval = 150M
LOGMINER: session# = 1, reader process P000 started with
pid=22 OS id=1976
LOGMINER: session# = 1, builder process P001 started with
pid=23 OS id=3160
Wed Nov 28 14:33:09 2012
LOGMINER: Begin mining logfile: .../ARC00416_0797957515.001
Wed Nov 28 14:33:09 2012
LOGMINER: Turning ON Log Auto Delete
Wed Nov 28 14:33:09 2012
LOGMINER: End mining logfile: .../ARC00416_0797957515.001
LOGSTDBY Analyzer process P003 started with pid=25 OS id=1116
LOGSTDBY Apply process P006 started with pid=28 OS id=3124
LOGMINER: session# = 1, preparer process P002 started with
pid=24 OS id=3848
LOGSTDBY Apply process P005 started with pid=27 OS id=3860
LOGSTDBY Apply process P007 started with pid=29 OS id=3644
LOGSTDBY Apply process P008 started with pid=30 OS id=2368
LOGSTDBY Apply process P004 started with pid=26 OS id=3332
Wed Nov 28 14:35:01 2012
Redo Shipping Client Connected as PUBLIC
-- Connected User is Valid
RFS[1]: Assigned to RFS process 3792
RFS[1]: Identified database type as 'logical standby'
Wed Nov 28 14:35:01 2012
RFS LogMiner: Client enabled and ready for notification
RFS[1]: Archived Log: '.../ARC00417_0797957515.001'
Wed Nov 28 14:35:03 2012
```

```
RFS LogMiner: Registered logfile [.../ARC00417_0797957515.001]
to LogMiner session id [1]
RFS[1]: Archived Log: '.../ARC00418_0797957515.001'
Wed Nov 28 14:35:03 2012
RFS LogMiner: Registered logfile [.../ARC00418_0797957515.001]
to LogMiner session id [1]
Wed Nov 28 14:35:04 2012
LOGMINER: Begin mining logfile: .../ARC00417_0797957515.001
Wed Nov 28 14:35:04 2012
LOGMINER: End mining logfile: .../ARC00417_0797957515.001
Wed Nov 28 14:35:04 2012
LOGMINER: Begin mining logfile: .../ARC00418_0797957515.001
Wed Nov 28 14:35:04 2012
LOGMINER: End mining logfile: .../ARC00418_0797957515.001
```

要停止逻辑备用数据库端的 SQL Apply，可执行如下指令：

```
SYS@LOGDB>alter database stop logical standby apply;

Database altered.
```

要启动逻辑备用数据库端的实时应用，可执行如下指令：

```
SYS@LOGDB>alter database start logical standby apply
immediate;
alter database start logical standby apply immediate
*
ERROR at line 1:
ORA-16239: IMMEDIATE option not available without standby redo
logs.
```

备用端的实时应用需要创建备用重做日志，在创建备用重做日志组（组数量 N+1）后启动实时应用成功。

```
SYS@LOGDB>alter database add standby logfile group 3 '+GFRA'
size 16M;
Database altered.
SYS@LOGDB>alter database add standby logfile group 4 '+ GFRA '
size 16M;
Database altered.
SYS@LOGDB>alter database add standby logfile group 5 '+ GFRA '
size 16M;
```

```
Database altered.

SYS@LOGDB>alter database start logical standby apply
immediate;
Database altered.
```

第7章

日志传输与应用服务

本章介绍主数据库端的日志传输和备用数据库端的接收与应用过程。

7.1 配置日志传输服务

主备用数据库的日志传输与接收需要 SYS 用户及其口令文件的授权连接及其会话，这是传输权限的要求。Data Guard 环境下的所有备用端，需要与主数据库端完全相同的口令文件。当主数据库端的口令文件更新后，所有备用数据库端的口令文件都需要重新更换。

参数 LOG_ARCHIVE_DEST_n 决定主数据库端重做日志的传输目的地及其传输特性。该参数有一系列属性可以控制重做传输特性，主要的属性如下：

※ SERVICE 属性通过指定 Oracle Net 的网络服务名来决定传输的目的地。

※ DB_UNIQUE_NAME 指定备用端数据库的唯一标识，该属性设置值必须存在于参数 LOG_ARCHIVE_CONFIG 的 DG_CONFIG 属性列表中，如果不匹配，则连接无法建立。

※ SYNC 或 ASYNC 属性用来指定日志的传输模式是同步或异步。

※ AFFIRM 或 NOAFFIRM 属性指定磁盘 I/O 模式，控制重做日志接收端是否向发送端发送确认信息，即是否报告日志在接收端已经写入磁盘文件（备用重做日志）的确认信息，默认是发送端不需要接收端的确认（NOAFFIRM）。

※ NET_TIMEOUT 属性指定 LGWR 进程等待 LNS 进程做出响应的秒数，如果超过此时限 LNS 进程没响应，则表明主数据库视备用端出现故障。该参数默认值 30 秒。

※ REOPEN 属性控制主数据库再次尝试连接故障的备用数据库之前的间隔，默认值是 300 秒，即如果由于备用端故障导致主备用端失去联系，之后如果备用端再次恢复正常，则至少 5 分钟后主备用端的连接才能恢复。

※ VALID_FOR 属性设置有效的归档日志文件类型（ONLINE_LOGFILE、STANDBY_LOGFILE、ALL_LOGFILES）和 Data Guard 数据库角色类型（PRIMARY_ROLE、STANDBY_ROLE、ALL_ROLES），只有当这两项设置的判断都为 True 时，LOG_ARCHIVE_DEST_n 参数设置的传输目的地才有效。

下面是参数设置的示例：

```
DB_UNIQUE_NAME=NETDB
LOG_ARCHIVE_CONFIG='DG_CONFIG=(NETDB,PHYDB,LOGDB)'
LOG_ARCHIVE_DEST_2='SERVICE=PHYDB ASYNC NOAFFIRM
VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)
REOPEN=60 COMPRESSION=ENABLE DB_UNIQUE_NAME=PHYDB'
LOG_ARCHIVE_DEST_STATE_2='ENABLE'
LOG_ARCHIVE_DEST_3='SERVICE=LOGDB SYNC AFFIRM NET_TIMEOUT=30
VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE) REOPEN=60
COMPRESSION=ENABLE DB_UNIQUE_NAME=LOGDB'
LOG_ARCHIVE_DEST_STATE_3='ENABLE'
```

查看视图 V$ARCHIVE_DEST 可以了解当前 LOG_ARCHIVE_DEST_n 参数设置的详细信息，包括一系列属性的默认值。

查看视图 V$ARCHIVE_DEST_STATUS 可以了解当前各归档目标的使用状况。

```
SYS@BJING>SELECT DEST_ID, DESTINATION, STATUS, TYPE, ARCHIVED_
SEQ#, APPLIED_SEQ# FROM V$ARCHIVE_DEST_STATUS -
> WHERE STATUS <> 'DEFERRED' AND STATUS <> 'INACTIVE';

DEST_ID DESTINATION  STATUS  TYPE  ARCHIVED_SEQ# APPLIED_SEQ#
------------------------------------------------------------
1           VALID        LOCAL        194           0
2           SHHAI        VALID        PHYSICAL      194    176
3           JINAN        VALID        LOGICAL       9      8
```

当已经设置某个归档目的地的状态为无效（INVALID）时，可以进一步查看视图 V$ARCHIVE_DEST_STATUS 的 ERROR 字段，了解其中的原因。

7.2　重做数据的接收

备用数据库端接收来自主数据库端的重做数据，该数据有 3 个可能的存储位置：一是备用端归档日志的存储位置（默认位于快速恢复区 FLASH Recovery Area）；二是参数 standby_archive_dest 设置的存储位置；三是备用重做日

志（Standby Redo Log），这是Oracle推荐使用的备用端接收的重做数据的存储位置，在实时应用的情况下必须使用备用重做日志。

备用重做日志是备用数据库端特有的日志文件类型，被设计专门用来接收来自主数据库传输过来的重做数据。当然，如果在主数据库端创建备用重做日志也是可以的，但此时处于未激活状态。

备用数据库使用备用重做日志的方式与主数据库使用联机日志的方式是相似的，都是以轮循方式使用，区别在于备用数据库接收来自主数据库的数据，由备用端的RFS（Remote File Server）进程负责写入。当主数据库日志切换操作时，也会导致备用重做日志的切换并产生归档操作。

```
SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.

SYS@PHYDB>alter database add standby logfile group 3 '+DG_PHY'
size 16M;

Database altered.

SYS@PHYDB>alter database add standby logfile group 4 '+DG_PHY'
size 16M;

Database altered.

SYS@PHYDB>alter database add standby logfile group 5 '+DG_PHY'
size 16M;

Database altered.

SYS@PHYDB>select group#,member,type from v$logfile;
GROUP# MEMBER                                          TYPE
---------- ---------------------------------------- -------
1 +DG_PHY/phydb/onlinelog/group_1.262.798328555     ONLINE
2 +DG_PHY/phydb/onlinelog/group_2.263.798328555     ONLINE
3 +DG_PHY/phydb/onlinelog/group_3.265.798386713     STANDBY
4 +DG_PHY/phydb/onlinelog/group_4.266.798386725     STANDBY
5 +DG_PHY/phydb/onlinelog/group_5.267.798386731     STANDBY
```

```
SYS@PHYDB>alter database recover managed standby database
disconnect;

Database altered.

SYS@PHYDB>select group#,sequence#,used,archived,status
from v$standby_log;

GROUP#      SEQUENCE#    USED   ARCHIVED     STATUS
---------- ---------- ---------- --------- ----------
  3         0            512     NO          UNASSIGNED
  4         122          512     YES         ACTIVE
  5         0            512     YES         UNASSIGNED
```

备用重做日志的状态随主数据库联机日志的切换而切换，其日志序列号也就是主数据库中的联机日志序列号。

创建和使用备用重做日志需要遵循下列原则：

（1）备用重做日志组的数量应该是主数据库联机日志组数量加 1。

（2）备用重做日志文件的大小应该等于主数据库联机日志文件的大小。

（3）RAC 集群环境下，每个实例（线程）的备用重做日志组数量是该实例联机日志组数量加 1。

（4）为了 Data Guard 数据库角色的平滑转换，建议在主数据库中也应该创建备用重做日志。

（5）从重做日志在备用端的接收与应用的效率考虑，一般备用重做日志不需要镜像，即每个备用重做日志组仅需一个成员。

默认情况下，备用重做日志的归档操作与联机日志归档服从同样的规律，但也可以单独配置备用重做日志的归档。参考如下：

```
LOG_ARCHIVE_DEST_2 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST
VALID_FOR=(STANDBY_LOGFILE,STANDBY_ROLE)'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

或者

```
LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive
VALID_FOR=(STANDBY_LOGFILE,STANDBY_ROLE)'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

当备用重做日志不可用，同时也没有设置 standby_archive_dest 参数时，备用端接收到的重做日志会直接写入备用端的归档日志位置。在逻辑备用数据库中，备用日志文件的切换产生的归档文件作为外部归档日志存储在快速恢复区（Flash Recovery Area）。

7.3 备用端的日志应用

Standby 数据库的 MRP 或 LSP 进程读取接收到的 REDO 数据（来自于 Standby Redologs，也可能来自于 Standby 端的归档文件），再将其中的数据变更写入 Standby 数据库。写入的两种方式为：物理 Standby 通过 REDO 应用、逻辑 Standby 通过 SQL 应用。

日志应用服务，就是在 Standby Database 上通过日志重演 Primary Database 的数据变更，从而实现两个数据库的数据同步。根据 Standby Database 重演日志方式的不同，可分为物理 Standby（Physical Standby）和逻辑 Standby（Logical Standby）。

Physical Standby 使用的是 Media Recovery 技术，在数据块级别进行恢复，这种方式没有数据类型的限制，可以保证两个数据库完全一致。Physical Standby 数据库只能在 Mount 状态下进行恢复，也可以打开，但需要以只读方式打开。

Logical Standby 使用的是 Logminer 技术，通过把日志内容还原成 SQL 语句，然后 SQL 引擎执行这些语句，Logminer Standby 不支持所有数据类型，可以在视图 DBA_LOGSTDBY_UNSUPPORTED 中查看不支持的数据类型，如果使用了这种数据类型，则不能保证数据库完全一致。Logical Standby 数据库可以在恢复的同时进行读 / 写操作。

根据 Redo Apply 发生的时间备用端的日志应用可以分成两种。

※ 归档时应用：这种方式在 Primary Database 发生日志切换，触发 Standby Database 归档操作，归档完成后触发恢复。这也是默认的恢复方式。

※ 实时应用（Real-Time Apply），这种方式必须 Standby Redo Log，每当日志被写入 Standby Redo Log 时，就会触发恢复，使用这种方式的好处在与可以减少数据库切换（Switchover 或者 Failover）的时间，因为切换时间主要用在剩余日志的恢复上。

从 Oracle 10g 开始，Data Guard 的实时应用新特性可以在将重做数据一写到备用重做日志（SRL）中时，就将重做数据应用到备用数据库中（无论采用重做应用还是 SQL 应用）。Data Guard 的早期版本要求，先在备用数据库中归档重做数据，才能以归档日志的形式应用重做数据。

实时应用特性允许备用数据库与主数据库保持密切同步，从而启动最新和实时报表（特别用于 Data Guard SQL 应用）。同时，这一特性也可减少转换时间和故障切换时间，进而降低计划内和计划外业务停机时间。灾难的影响通常根据恢复点目标（RPO—即在出现灾难时，业务可以经受得住丢失多少数据）和恢复时间目标（RTO—即在出现灾难时，业务可以经受得住停机多长时间）来衡量。使用 Data Guard，当结合使用最大保护模式（确保即使在出现灾难时也不会丢失数据）和实时应用时，企业在出现灾难时不仅获得了零数据丢失的好处，还获得了最少停机时间的好处，这使得 Oracle Data Guard 成为当今唯一为业务提供最佳 RPO 和 RTO 好处的解决方案。

对于物理备用数据库，启动 Redo 应用服务：

```
SYS@PHYDB>startup mount
ORACLE instance started.

Total System Global Area  167772160 bytes
Fixed Size                  1247876 bytes
Variable Size              58721660 bytes
Database Buffers          100663296 bytes
Redo Buffers                7139328 bytes
Database mounted.
SYS@PHYDB>alter database recover managed standby database
disconnect;

Database altered.

SYS@PHYDB>select process,status,thread#,sequence#,client_pid
2  from v$managed_standby;

PROCESS     STATUS        THREAD#    SEQUENCE#   CLIENT_PID
--------- ------------ ---------- ---------- ----------------
ARCH        CONNECTED     0          0           5492
ARCH        CONNECTED     0          0           5596
MRP0        WAIT_FOR_LOG  1          124         N/A
RFS         IDLE          0          0           3932
```

若要在物理备用数据库中启动 Redo 实时应用服务（需要备用数据库端备用重做日志的配合）：

```
SYS@PHYDB>alter database recover managed standby database
2 using current logfile disconnect;
```

```
Database altered.
```

取消物理备用数据库端的 Redo 应用：

```
SYS@LOGDB> alter database recover managed standby database
cancel;

Database altered.
```

对于逻辑备用数据库，在数据库处于打开状态时启动 SQL 应用：

```
SYS@LOGDB>alter database start logical standby apply;

Database altered.
```

在逻辑备用数据库端启动实时 SQL 应用（同样需要主数据库端日志的同步传输 SYNC 和备用数据库端备用重做日志的配合），使用 immdiate 选项：

```
SYS@LOGDB>alter database start logical standby apply
immediate;

Database altered.
```

取消逻辑备用数据库端的 SQL 应用：

```
SYS@LOGDB>alter database stop logical standby apply;

Database altered.
```

强制取消逻辑备用数据库端的 SQL 应用（正在执行的事务异常终止）：

```
SYS@LOGDB>alter database abort logical standby apply;

Database altered.
```

默认情况下，Log 应用服务会等待单个归档文件全部接收之后再启动应用，如果 Standby 数据库配置了 Standby Redologs，就可以打开实时应用（Real-Time Apply），这样，Data Guard 就不需要再等待接收完归档文件，只要 RFS 进程将 Redo 数据写入 Standby Redologs，即可通过 MRP/LSP 实时写向 Standby 数据库。

7.4 日志间隔的手工处理

当 Primary Database 的某些日志没有成功发送到 Standby Database 时，将发生归档间隙（Archive Gap）。

备用端缺失的这些日志就是间隙（Gap）。Data Guard 能够自动检测，解决归档间隙，不需要 DBA 的介入。这需要配置 FAL_CLIENT 和 FAL_SERVER 参数。FAL（Fetch Archive Log，FAL）的意思是 Standby Database 主动发起的“取”日志的过程，Standby Database 就是 FAL_CLIENT，它从 FAL_SERVER 中取这些 Gap Log，这个 FAL_SERVER 可以是 Primary Database，也可以是其他的 Standby Database。

FAL_CLIENT 和 FAL_SERVER 参数指向的都是 Oracle Net 的网络服务名。FAL_CLIENT 通过网络向 FAL_SERVER 发送请求，FAL_SERVER 通过网络向 FAL_CLIENT 发送缺失的日志。这两个参数都只需要在备用数据库端进行设置。

当然，Data Guard 中出现日志间隙，必要的时候也可以手工处理。

首先，查询 V$ARCHIVE_GAP 视图，获得关于日志间隙的信息，类似如下：

```
SYS@NETDB>select * from v$archive_gap;

THREAD#     LOW_SEQUENCE#       HIGH_SEQUENCE
----------------------------------------------------
1           118                 123
```

然后，查询对应的归档文件：

```
SYS@NETDB>SELECT NAME FROM V$ARCHIVED_LOG WHERE
   2   THREAD#=1 AND DEST_ID=1 AND SEQUENCE# BETWEEN 118 AND
123;
NAME
-------------------------------------------------------------
+DG_FRA/NETDB/ARCHIVELOG/2012_11_03/O1_MF_1_118_899FH1WS_.ARC
+DG_FRA/NETDB/ARCHIVELOG/2012_11_03/O1_MF_1_119_899FND89_.ARC
+DG_FRA/NETDB/ARCHIVELOG/2012_11_03/O1_MF_1_120_899FO36T_.ARC
+DG_FRA/NETDB/ARCHIVELOG/2012_11_03/O1_MF_1_121_899FVFTV_.ARC
+DG_FRA/NETDB/ARCHIVELOG/2012_11_03/O1_MF_1_122_899N4W89_.ARC
+DG_FRA/NETDB/ARCHIVELOG/2012_11_03/O1_MF_1_123_899N4ZWS_.ARC

6 rows selected.
```

对于这些间隙的归档日志文件，在备用端进行日志注册（注意，物理备用和逻辑备用的语法稍有不同）：

```
SYS@PHYDB>ALTER DATABASE REGISTER [PHYSICAL] LOGFILE '...';
SYS@LOGDB>ALTER DATABASE REGISTER LOGICAL LOGFILE '...';
```

注册之后，再次启动 Redo 应用或 SQL 应用，即可消除日志间隙。

7.5 数据保护模式

Data Guard 具有 3 种高水平的数据保护模式来平衡成本、可用性、性能和事务保护。要确定适当的数据保护模式，系统设计人员需要根据用户对系统响应时间的要求来估量它们对数据保护的业务要求。

7.5.1 最大保护模式（Maximum Protection）

最大保护模式为主数据库提供了最高水平的数据保护，从而确保了一个全面的零数据丢失灾难恢复解决方案。当在最大保护模式下运行时，重做记录由日志写入器进程从主数据库同步地传输到备用数据库，并且直到确认事务数据在至少一个备用服务器上的磁盘上可用时，才在主数据库上提交事务。这种模式必须配置至少两个备用数据库，从而提供双重故障保护。当最后参与的备用数据库不可用时，主数据库上的处理将停止。这就确保了当主数据库与其所有备用数据库失去联系时，不会丢失事务。

这种模式能够确保绝无数据丢失。要实现这一步当然是有代价的，它要求所有的事务在提交前其 Redo 不仅被写入到本地的 Online Redologs，还要同时写入到 Standby 数据库的 Standby Redologs，并确认 Redo 数据至少在一个 Standby 数据库中可用（如果有多个的话），然后才会在 Primary 数据库上提交。如果出现了什么故障导致 Standby 数据库不可用的话（比如网络中断），Primary 数据库会被 Shutdown，以防止数据丢失。

使用这种方式要求 Standby Database 必须配置 Standby Redo Log，而 Primary Database 必须使用 LGWR、SYNC、AFFIRM 方式归档到 Standby Database。

由于重做传输的同步特性，这种最大保护模式可能潜在地影响主数据库响应时间。可以通过配置一个低延迟网络，并为它分配足够应付高峰事务负载的带宽来将这种影响减到最小。需要这种最大保护模式的企业有股票交易所、货币交易所、金融机构等。

7.5.2 最高可用性保护（Maximum Availability）

这种模式在不影响 Primary 数据库可用性的前提下，提供最高级别的数据保护策略。其实现方式与最大保护模式类似，也是要求本地事务在提交前必须至少写入一台 Standby 数据库的 Standby Redologs 中，不过与最大保护模式不同的是，如果出现故障导致 Standby 数据库无法访问，Primary 数据库并不会被 Shutdown，而是自动转为最高性能模式，等 Standby 数据库恢复正常之后，Primary 数据库又会自动转换成最高可用性模式。

这种方式虽然会尽量避免数据丢失，但不能绝对保证数据完全一致。这种方式要求 Standby Database 必须配置 Standby Redo Log，而 Primary Database 必须使用 LGWR、SYNC、AFFIRM 方式归档到 Standby Database。

最高可用性模式拥有仅次于最高水平的主数据库数据可用性，并提供零数据丢失和防止单组件故障。与最大保护模式一样，重做数据由日志写入器进程从主数据库同步地传输到备用数据库，直到确认事务数据在备用服务器的磁盘上可用时，事务才在主数据库上完成。不过，在这种模式下（与最大保护模式不同），如果最后参与的备用数据库变为不可用，例如，由于网络连接问题，处理将在主数据库上继续进行。备用数据库与主数据库相比，可能暂时落在后面，但当它再次变为可用时，备用数据库将自动同步，而不会丢失数据。由于同步重做传输，所以这种保护模式可潜在地影响响应时间和吞吐量。

数据库管理人员可以通过配置一个低延迟网络，并为它分配足够应付高峰事务负载的带宽来将这种影响减到最小。最高可用性模式适用于想要在生产站点上出现严重中断时（假定没有其他故障）确保获得零数据丢失保护，但不想让生产数据库受网络 / 备用服务器故障影响的企业。

7.5.3　最高性能保护（Maximum Performance）

缺省模式。 这种模式在不影响 Primary 数据库性能的前提下，提供最高级别的数据保护策略。事务可以随时提交，当前 Primary 数据库的 Redo 数据至少需要写入一个 Standby 数据库，不过这种写入可以是异步的。如果网络条件理想的话，这种模式能够提供类似最高可用性的数据保护，这也是创建 Standby 数据库时，系统的默认保护模式。

这种模式可以使用 LGWR ASYNC 或者 ARCH 进程实现，Standby Database 也不要求使用 Standby Redo Log。

在这种模式下，当主数据库处理事务时，重做数据由日志写入器进程异步传输到备用数据库上。另外，也可以将主数据库上的归档器进程配置为在这种模式下传输重做数据。在任何情况下，均先完成主数据库上的写操作，主数据库的提交操作不等待备用数据库确认接收。如果任意备用目标数据库变为不可用，则处理将在主数据库上继续进行，这对性能只有很小的影响或没有影响。不过在这种情况下，数据库告警日志将记录错误消息，并可以通过 Enterprise Manager 相应地设置告警。在主数据库出现故障的情况下，可能有一些在主数据库上提交了的事务没有传输到备用数据库中，如果网络有足够的吞吐量来跟上重做流量高峰，则丢失的事务将非常少或者为零。

当主数据库上的可用性和性能比丢失少量数据的风险更重要时，应该使用最高性能模式。这种模式还适合于 WAN 上的 Data Guard 部署，在 WAN 中，网络的内在延迟可能限制同步重做传输的适用性。

7.5.4　保护模式的实施与转换

调整主数据库保护模式的基本步骤如下：

（1）关闭数据库，重启到 Mount 状态，如果是 RAC，则需要关闭所有实例，然后只启动一个实例到 mount 状态。

（2）修改模式：

```
ALTER DATABASE SET STANDBY DATABASE TO MAXIMIZE
{PROTECTION | AVAILABILITY | PERFORMANCE};
```

（3）打开数据库：

```
alter database open;
```

（4）确认修改数据保护模式：

```
select dbid, name, protection_mode, protection_level from
v$database;
```

下面将本章示例用的数据库由默认的保护模式调整到最高可用性保护，再到最大保护模式。

表 7-1 对 3 种保护模式的设置条件、要求及其对主数据库的数据保护、性能影响进行了汇总比较，在具体实施时需要参考对照。

表 7-1　3 种保护模式的比较

比较项目	最高性能	最高可用性	最大保护
传输模式	LGWR|ARCH ASYNC	LGWR SYNC	LGWR SYNC
磁盘写确认	NOAFFIRM	AFFIRM	AFFIRM
备用重做日志	可选，但建议	物理备用必需	必需
备用类型	物理 / 逻辑	物理 / 逻辑	物理
实时应用	可选	建议	应该
保护与性能	几乎无性能影响	性能与保护的折中	零数据丢失

当前主数据库的保护模式（默认模式）：

```
SYS@NETDB>select dbid, name, protection_mode, protection_level
from v$database;

      DBID NAME      PROTECTION_MODE      PROTECTION_LEVEL
---------- --------- -------------------- --------------------
2551222283 NETDB     MAXIMUM PERFORMANCE  MAXIMUM PERFORMANCE
```

当前主数据库的传输模式：

```
SYS@NETDB>show parameters log_archive_dest_2
```

```
NAME                          VALUE
------------------ ------------------------------
log_archive_dest_2  SERVICE=PHYDB LGWR ASYNC DB_UNIQUE_NAME=PHYDB
```

当前备用数据库的日志信息：

```
SYS@PHYDB>select group#,type,member from v$logfile;

GROUP#   TYPE   MEMBER
------- ------- ----------------------------------------
   1    ONLINE  +DG_PHY/phydb/onlinelog/group_1.262.798328555
   2    ONLINE  +DG_PHY/phydb/onlinelog/group_2.263.798328555
   3    STANDBY +DG_PHY/phydb/onlinelog/group_3.265.798386713
   4    STANDBY +DG_PHY/phydb/onlinelog/group_4.266.798386725
   5    STANDBY +DG_PHY/phydb/onlinelog/group_5.267.798386731
```

要更改对主数据库的保护模式，首先需要调整传输模式：

```
SYS@NETDB>alter system set
2     log_archive_dest_2='SERVICE=PHYDB LGWR SYNC AFFIRM
3     DB_UNIQUE_NAME=PHYDB' scope=spfile;

System altered.

SYS@NETDB>startup mount
ORACLE instance started.

Total System Global Area  167772160 bytes
Fixed Size                  1247876 bytes
Variable Size              62915964 bytes
Database Buffers           96468992 bytes
Redo Buffers                7139328 bytes
Database mounted.
SYS@NETDB>alter database set standby to maximize availability;

Database altered.

SYS@NETDB>alter database open;

Database altered.
```

此时若备用数据库处于关闭状态，则查看主数据库的保护状态（注意保护级别字段 PROTECTION_LEVEL 的取值）：

```
SYS@NETDB>select dbid, name, protection_mode, protection_level
  2  from v$database;

DBID        NAME   PROTECTION_MODE            PROTECTION_LEVEL
---------- ------- ---------------------- --------------------
2551222283  NETDB  MAXIMUM AVAILABILITY       RESYNCHRONIZATION
```

启动备用数据库后，稍等片刻（需要等待主备用数据库连接 REOPEN），查看主备用数据库获得相同的结果：

```
SYS@NETDB>select dbid, name, protection_mode, protection_level
  2  from v$database;

      DBID NAME      PROTECTION_MODE      PROTECTION_LEVEL
---------- --------- -------------------- --------------------
2551222283 NETDB     MAXIMUM AVAILABILITY MAXIMUM AVAILABILITY
```

接着再将主数据库转换到最大保护状态（Mount 状态下）：

```
SYS@NETDB>alter database set standby to maximize protection;

Database altered.

SYS@NETDB>select dbid, name, protection_mode, protection_level
  2  from v$database;

      DBID NAME      PROTECTION_MODE      PROTECTION_LEVEL
---------- --------- -------------------- --------------------
2551222283 NETDB     MAXIMUM PROTECTION   UNPROTECTED

SYS@NETDB>alter database open;

Database altered.

SYS@NETDB>select dbid, name, protection_mode, protection_level
  2  from v$database;

      DBID NAME      PROTECTION_MODE      PROTECTION_LEVEL
```

```
---------- --------- -------------------- --------------------
2551222283 NETDB     MAXIMUM PROTECTION   MAXIMUM PROTECTION
```

需要注意的是，在最大保护模式下，打开主数据库时，备用数据库必须处于 mount 状态（能够接收重做日志），否则会出现如下错误并导致数据库实例彻底关闭：

```
SYS@NETDB>alter database open;
alter database open
*
ERROR at line 1:
ORA-00449: background process 'DBW0' unexpectedly terminated
with error 16072
ORA-16072: a minimum of one standby database destination is
required.
```

与此同时，在最大保护模式下，主数据库在打开状态，唯一的物理备用数据库是不允许关闭的，如果执行 shutdown 指令，则会返回如下信息：

```
SYS@PHYDB>shutdown
ORA-01154: database busy. Open, close, mount, and dismount not
allowed now.
```

7.6　监控物理备用数据库

7.6.1　进程名称及其状态

视图 V$MANAGED_STANDBY 专门用于物理备用数据库。执行如下查询可以了解物理备用端启动的进程，CLIENT_PROCESS 字段标识日志传输过程中主数据库端对应的进程，有 3 种取值：Archival、ARCH、LGWR，分别表示人工归档、归档进程和联机日志写进程。

```
SYS@PHYDB>SELECT PROCESS, CLIENT_PROCESS, SEQUENCE#, STATUS
  2  FROM V$MANAGED_STANDBY;

PROCESS   CLIENT_P  SEQUENCE#   STATUS
--------- -------- ---------- ------------
ARCH      ARCH            0     CONNECTED
RFS       LGWR           38     IDLE
MRP0      N/A            38     WAIT_FOR_LOG
```

这里的进程主要有 RFS（Remote File Server）、MRP0（Managed Recovery Process）、LNS（Network Server process）、MR(fg)（Foreground recovery session）等，其中，MRP（Managed Recovery

Process）进程是物理备用数据库端所特有的，该进程执行关键的 Redo Apply 操作，其状态有如下几种取值。

※ WAIT_FOR_LOG：正在等待从主数据库端传输过来的待完成的日志。

※ WAIT_FOR_GAP：正在等待完成日志间隔的填补。

※ APPLYING_LOG：正在执行日志应用操作。

7.6.2 物理备用端的应用进展

通过视图 V$ARCHIVED_LOG 查看物理备用数据库端的日志应用情况，其中，FAL 字段指示备用端的归档日志是否是 FAL 请求而产生的结果。

```
SYS@PHYDB>select name, creator, sequence#,applied, completion_
time,
  2  fal from v$archived_log order by sequence#;

NAME                          CREATOR  SEQ   APP COMPLETION_T FAL
----------------------------- -------- ----- --- ---------------
.../ARC00024_0803599033.001   ARCH   24    YES 23-JAN-13    YES
.../ARC00025_0803599033.001   ARCH   25    YES 23-JAN-13    YES
.../ARC00026_0803599033.001   ARCH   26    YES 23-JAN-13    YES
.../ARC00027_0803599033.001   ARCH   27    YES 23-JAN-13    YES
.../ARC00028_0803599033.001   LGWR   28    YES 23-JAN-13    NO
.../ARC00029_0803599033.001   ARCH   29    YES 25-JAN-13    YES
.../ARC00030_0803599033.001   ARCH   30    YES 25-JAN-13    YES

SYS@PHYDB>SELECT THREAD#, MAX(SEQUENCE#) AS "LAST_APPLIED_LOG"
  2  FROM V$LOG_HISTORY GROUP BY THREAD#;

   THREAD# LAST_APPLIED_LOG
---------- ----------------
         1              237
```

正常情况下，物理备用数据库处于一种受托管的介质恢复状态。备用数据库实例一旦启动到 Mount 状态，备用端即可接收来自主数据库的重做日志，此后就可以启动托管的介质恢复。当停止托管的介质恢复后，物理备用数据库可以以只读方式打开。

如果数据库启动了闪回功能，则物理备数据库能以读 / 写模式临时打开以用于开发、报表或测试，然后闪回到过去的点以回复到物理备用数据库。当数据库闪回时，Data Guard 自动与主数据库同步备用数据库，而不需要从主数据库的备份复制重建物理备数据库。

7.6.3　重要的相关视图

```
V$DATAGUARD_STATS
V$DATAGUARD_STATUS
V$MANAGED_STANDBY
V$STANDBY_EVENT_HISTOGRAM
V$STANDBY_LOG
V$RECOVERY_PROGRESS
```

7.7　重新创建物理备用控制文件

物理备用数据库在运行过程中由于某种原因导致物理备用控制文件损坏或丢失，可以通过主数据库重新创建物理备用控制文件，然后在物理备用端执行恢复即可完成。

（1）如果物理备用数据库的托管恢复进程还在运行，则停止该进程。

```
SQL> alter database recover managed standby database cancel;
```

（2）关闭物理备用数据库。

```
SQL> shutdown immediate;
```

（3）连接到主数据库。

```
SQL> connect / as sysdba
```

（4）创建新的备用控制文件。

```
SQL> alter database create standby controlfile as '...';
```

（5）将创建的控制文件移至物理备用端。

（6）在备用端使用新的控制文件加载物理备用数据库。

从 Oracle 10g 版本开始，直接使用 startup mount 指令即可，但在 Oracle 10g 版本之前，需要如下处理：

```
SQL> startup nomount
SQL> alter database mount standby database;
```

（7）检查确认控制文件中数据文件和日志文件的指向。如果备用端的数据库文件存储结构与主数据库不同，则该步骤需要确保控制文件中记录的数据文件和日志文件的位置符合物理备用端的物理存储。如果两者不一致，可使用如下两种方法之一进行调整。

※ 在物理备用数据库端设置 db_file_name_convert 和 log_file_name_convert 参数，重新加载实例。

※ 直接使用如下指令修改：

```
SQL> alter database rename file '...' to '...';
```

（8）检查备用重做日志。如果物理备用数据库配置了备用重做日志，而主数据库中没有预先配置备用重做日志，则此刻需要在物理备用数据库的控制文件中重新登记备用重做日志组。

```
SQL> alter database add standby logfile group ('...') size
reuse;
...
```

（9）重新启动数据库闪回。如果物理备用数据库原来启动了闪回特性，此时需要关闭后重新启动。此步骤同样需更新新的备用控制文件中关于数据库闪回特性的记录。

```
SQL> alter database flashback off;
SQL> alter database flashback on;
```

（10）重新启动物理备用数据库的托管恢复。

```
SQL> alter database recover managed standby database [using
current logfile] disconnect;
```

至此，物理备用数据库控制文件恢复完毕。

第8章

逻辑备用数据库的管理

本质上，物理备用数据库和主数据库是同一数据库（DBID 和 Incarnation）。逻辑备用数据库与物理备用数据库相比，是与主数据库不同的数据库，它仅通过等价的 SQL 与主数据库保持同步。逻辑备用数据库的突出优点是在打开状态下运行，对于逻辑备用维护的数据集可以进行只读操作，对逻辑备用维护之外的数据集，可以方便地进行读 / 写操作，这也提供了额外的好处，如在逻辑备用数据库中添加必要的增加访问效率的相关对象（如索引和物化视图等）。但从管理与维护的角度，逻辑备用数据库要比物理备用数据库来得复杂，这体现在多个方面，主要有：一是逻辑备用需要的支持进程比物理备用多，这就需要消耗更多的计算机资源；二是逻辑备用数据库目前还有一些不支持的数据类型、数据库对象甚至某些 DDL 语句；三是可以有选择地定义逻辑备用维护的数据集。

8.1 逻辑备用的状态

在物理备用数据库中，通过已应用的日志序列号了解物理备用的进展。但在逻辑备用数据库中，逻辑备用的日志序列号已与主数据库完全不同（在创建逻辑备用的过程中必须经历一次 resetlogs 操作）。在备用数据库中，可以查询备用日志的信息了解进展，代码如下：

```
SYS@LOGDB>select group#,sequence#,used,archived,status
  2  from v$standby_log;

    GROUP#  SEQUENCE#       USED       ARC    STATUS
---------- ---------- ---------- --------- ----------
         3          0        512        NO    UNASSIGNED
         4          0        512        NO    UNASSIGNED
         5        197     192512       YES    ACTIVE
```

另外，还可以查询关于 Data Guard 的统计信息，了解传输滞后和应用滞后的

相关信息。传输滞后值表示在逻辑备用数据库中获得的日志在时间上的差异，应用滞后表示逻辑备用数据库中的数据与主数据库中的数据在时间上的差异。

```
SYS@LOGDB>select name, value, unit from v$dataguard_stats;

NAME                 VALUE        UNIT
-----------------------------------------------------
transport lag    +00 00:00:00     day(2) to second(0) interval
apply lag        +00 00:03:38     day(2) to second(0) interval
apply finish time +00 00:00:00.001 day(2) to second(3) interval
estimated startup time         11                   second
```

视图 V$LOGSTDBY_PROGRESS 反映逻辑备用数据库的应用进展，其中，MINING_TIME、MINING_SCN 字段反映日志挖掘阶段的进展，APPLIED_TIME、APPLIED_SCN 字段反映 SQL 应用的进展。

```
SYS@JINAN>select mining_scn,mining_time,applied_scn,applied_
time
  2  from v$logstdby_progress;

MINING_SCN MINING_TIME          APPLIED_SCN APPLIED_TIME
---------- ------------------- ----------- -------------------
831983 2014-01-17 15:12:15      831979 2014-01-17 15:12:13
```

8.2 SQL Apply 的内部设置

逻辑备用数据库 SQL Apply 的主要环节（见图 8-1），就进程处理而言，可划分为两个部分：日志挖掘服务和 SQL 应用服务，两者通过 LCR（Logical Change Records）缓存衔接起来。如果逻辑备用的 SQL Apply 需要优化调整的话，也应着手这三个环节。关于挖掘服务和应用服务启动的进程信息可以从视图 V$LOGSTDBY_PROCESS 中获得。下面的查询进程 READER 正在等待读取 1 号日志线程、日志序列号为 197 的日志内容，HIGH_SCN 字段指示各个进程的工作进展，ORA-16116: no work available 则指示该进程暂时没有工作负荷。

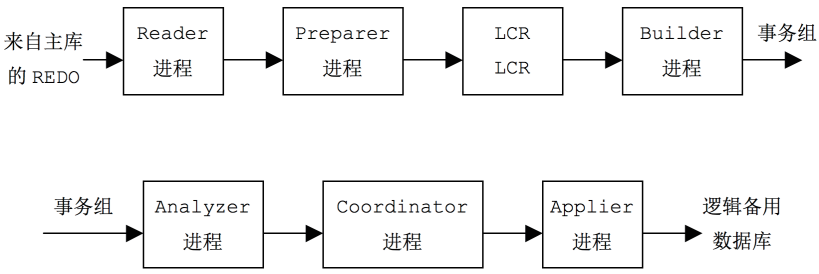


图 8-1　SQL Apply 的主要环节

```
SYS@LOGDB>select type, status from v$logstdby_process;
no rows selected

SYS@LOGDB>alter database start logical standby apply;
Database altered.

SYS@LOGDB>select sid, serial#, spid, type, high_scn
2 from v$logstdby_process;

SID          SERIAL#      SPID         TYPE         HIGH_SCN
------------------------------------------------------------
48           6            11074        COORDINATOR  7178242899
56           56           10858        READER       7178243497
46           1            10860        BUILDER      7178242901
45           1            10862        PREPARER     7178243295
37           1            10864        ANALYZER     7178242900
36           1            10866        APPLIER      7178239467
35           3            10868        APPLIER      7178239463
34           7            10870        APPLIER      7178239461
33           1            10872        APPLIER      7178239472

SYS@JINAN>select type, status from v$logstdby_process;

TYPE              STATUS
------------------------------------------------------------
COORDINATOR       ORA-16117: processing
READER            ORA-16240: Waiting for log file
                  (thread# 1, sequence# 197)
BUILDER           ORA-16116: no work available
```

```
PREPARER          ORA-16117: processing
ANALYZER          ORA-16120: dependencies being computed for
                  transaction at SCN 0x0001.abdb440a
APPLIER           ORA-16124: transaction 1 13 1427 is waiting
                  on another transaction
APPLIER           ORA-16121: applying transaction with commit
                  SCN 0x0001.abdb4390
APPLIER           ORA-16123: transaction 1 23 1231 is waiting
                  for commit approval
APPLIER           ORA-16116: no work available
```

8.2.1 挖掘服务的调整

逻辑备用数据库中的日志挖掘服务由 READER、PERPARER 和 BUILDER 三类进程组成。整体来讲，日志挖掘服务的目标是将重做记录转换为对应的事务，其中，READER 进程从备用日志文件或归档日志文件将日志记录读入共享缓存，PERPARER 进程将共享缓存中的重做记录转换为一系列的 LCR，BUILDER 进程将 LCR 分组或重组为事务。这里 READER 和 BUILDER 进程分别只有一个，而 PERPARER 进程通常会有多个。如果在逻辑备用数据库运行期间大部分时间（特别是在主数据库事务高峰期间）几乎所有的 PERPARER 进程都处于繁忙状态，则可以考虑增加 PERPARER 进程的数量。下列代码将调整进程数量的先后关系。

```
SYS@LOGDB>select name,value from dba_logstdby_parameters;

NAME                                    VALUE
------------------------------------------------------------
MAX_SGA                                 30
MAX_SERVERS                             9
PREPARE_SERVERS                         1
APPLY_SERVERS                           5
MAX_EVENTS_RECORDED                     10000
RECORD_SKIP_ERRORS                      TRUE
RECORD_SKIP_DDL                         TRUE
RECORD_APPLIED_DDL                      FALSE
RECORD_UNSUPPORTED_OPERATIONS           FALSE
EVENT_LOG_DEST                          DEST_EVENTS_TABLE
LOG_AUTO_DELETE                         TRUE
LOG_AUTO_DEL_RETENTION_TARGET           1440
PRESERVE_COMMIT_ORDER                   TRUE
ALLOW_TRANSFORMATION                    FALSE
```

```
14 rows selected.
SYS@LOGDB>exec dbms_logstdby.apply_set('PREPARE_SERVERS',2);
BEGIN dbms_logstdby.apply_set('PREPARE_SERVERS',2); END;

ERROR at line 1:
ORA-16212: number of processes specified for SQL Apply is too
great
ORA-06512: at "SYS.DBMS_LOGSTDBY", line 81
ORA-06512: at line 1

SYS@LOGDB>exec dbms_logstdby.apply_set('MAX_SERVERS',20);

PL/SQL procedure successfully completed.

SYS@LOGDB>exec dbms_logstdby.apply_set('PREPARE_SERVERS',2);
BEGIN dbms_logstdby.apply_set('PREPARE_SERVERS',2); END;
*
ERROR at line 1:
ORA-16212: number of processes specified for SQL Apply is too
great
ORA-06512: at "SYS.DBMS_LOGSTDBY", line 81
ORA-06512: at line 1

SYS@LOGDB>exec dbms_logstdby.apply_set('APPLY_SERVERS',10);

PL/SQL procedure successfully completed.

SYS@LOGDB>exec dbms_logstdby.apply_set('PREPARE_SERVERS',2);

PL/SQL procedure successfully completed.

SYS@LOGDB>select name,value from dba_logstdby_parameters;

NAME                                    VALUE
------------------------------------------------------
MAX_SGA                                 30
MAX_SERVERS                             20
```

```
PREPARE_SERVERS                          2
APPLY_SERVERS                            10
...
```

8.2.2 应用服务的调整

逻辑备用数据库 SQL Apply 的应用服务由 ANALYZER、COORDINATOR（即 LSP）、APPLIER 三类进程构成。从整体来说，应用服务的目标是将挖掘服务输出的事务重新应用到逻辑备用数据库中，其中，ANALYZER 进程分析事务之间的依赖关系及其先后顺序，COORDINATOR 进程在多个 APPLIER 进程之间协调分配需要应用的事务，APPLIER 进程负责具体的事务应用。这里的 ANALYZER、COORDINATOR 进程分别只有一个，通常 APPLIER 进程会有多个，以提高事务应用的吞吐量。如果在逻辑备用数据库运行期间大部分时间（特别是在主数据库事务高峰期间）几乎所有的 APPLIER 进程状态都处于繁忙状态，则可以考虑增加 APPLIER 进程的数量。调整的方法参照上面调整 PREPARR 进程数量的方法。需要注意的是，一般来说，日志挖掘的速度要远高于事务应用的速度，为了 SQL Apply 的协调运行，PREPARR 进程和 APPLIER 进程在数量上通常存在一定的比例关系，一个 PREPARR 进程需要多个 APPLIER 进程配合运行。

8.2.3 LCR 缓存的调整

位于共享缓冲池中的 LCR 缓存是日志挖掘服务和应用服务之间的通道，该缓存的大小影响日志挖掘服务和应用服务的效率，以及整个逻辑备用数据库的吞吐量。

```
SYS@LOGDB>select used_memory_size from v$logmnr_session
  2   where session_id=(select value from v$logstdby_stats
where name='logminer session id');

USED_MEMORY_SIZE
----------------
        33708560
```

当上述查询给出的 LCR 缓存的使用峰值接近甚至超过 LCR 缓存的设置值时，应该考虑增加 LCR 缓存的大小。下面的操作将 LCR 缓存的大小设置为 40MB。典型的生产用逻辑备用数据库的 LCR 缓存大小应该在 200MB 左右。

```
SYS@LOGDB>exec dbms_logstdby.apply_set('MAX_SGA',40);

PL/SQL procedure successfully completed.

SYS@LOGDB>select name,value from dba_logstdby_parameters;
```

```
NAME                                              VALUE
--------------------------------------------------
MAX_SGA                                           40
MAX_SERVERS                                       20
PREPARE_SERVERS                                   2
...
```

下面是一个由于 LCR 缓存不足导致的逻辑备用数据库出现故障的案例。该错误信息来自数据库告警日志,也可以监控DBA_LOGSTDBY_EVENTS 视图获得类似信息。

```
ORA-04031: unable to allocate 50012 bytes of shared memory
("shared pool","unknown object","apply shared t","commbuf_
knasctx[0]")
Mon Feb 11 09:43:31 2013
LOGMINER: session#=1, reader MS00 pid=30 OS id=5832 sid=19
stopped
Mon Feb 11 09:43:31 2013
LOGMINER: session#=1, builder MS01 pid=32 OS id=4920 sid=20
stopped
Mon Feb 11 09:43:31 2013
LOGMINER: session#=1, preparer MS02 pid=33 OS id=4804 sid=105
stopped
LOGSTDBY status: ORA-16222: automatic Logical Standby retry of
last action
LOGMINER: Parameters summary for session# = 1
LOGMINER: Number of processes = 3, Transaction Chunk Size =
201
LOGMINER: Memory Size = 30M, Checkpoint interval = 150M
LOGMINER: SpillScn 524757, ResetLogScn 293297
LOGMINER: summary for session# = 1
LOGMINER: StartScn: 0 (0x0000.00000000)
LOGMINER: EndScn: 0 (0x0000.00000000)
LOGMINER: HighConsumedScn: 0 (0x0000.00000000)
LOGMINER: session_flag 0x0
```

8.3　控制逻辑备用维护的数据集

8.3.1　设置备用数据库的防护

逻辑备用数据库正常运行时处于打开状态。为了防止逻辑备用数据库中的数据被意外修改（这种修改可能会导致主数据库中的数据无法被复制到备用库），默认

情况下，逻辑备用数据库的更新受主数据库的控制，普通用户（超级用户除外）不能使用 SQL 指令创建、修改、删除任何模式下的数据库表对象，即处于全方位的防护状态（ALL）。

逻辑备用数据库的防护状态可以取 3 种可能值：ALL、STABDBY 和 NONE。如果逻辑备用数据库的防护状态为 STANDBY 值，则用户不能修改任何由 SQL Apply 维护的表（只能通过主数据库的更新而更新），但用户可以自由创建新表，或者通过 DML 和 DDL 修改 SQL Apply 维护的数据集之外的表。

```
SYS@LOGDB>select guard_status from v$database;

GUARD_STATUS
-------------
ALL

SYS@LOGDB>alter database guard standby;

Database altered.

SYS@LOGDB>select guard_status from v$database;

GUARD_STATUS
-------------
STANDBY
```

如果在主数据库中将数据库的防护状态设置为 ALL，则相当于变相地把数据库设置为只读状态（此时不需要重启数据库）。

8.3.2 逻辑备用不支持的数据集

视图 DBA_LOGSTDBY_SKIP 给出逻辑备用数据库中 SQL Apply 并没有维护的数据集。可以通过下列查询查看 SQL Apply 不负责维护的内部数据集。

```
SYS@LOGDB>select owner,statement_opt from dba_logstdby_skip;

OWNER                          STATEMENT_OPT
------------------------------ ------------------------------
OUTLN                           INTERNAL SCHEMA
ORACLE_OCM                      INTERNAL SCHEMA
SYS                             INTERNAL SCHEMA
SYSTEM                          INTERNAL SCHEMA
```

```
APPQOSSYS                      INTERNAL SCHEMA
DIP                            INTERNAL SCHEMA
DBSNMP                         INTERNAL SCHEMA
...
```

由此，需要逻辑备用数据库维护的用户数据不能处于上述内部模式中。在 Oracle 数据库内部，这些内部模式下的数据库对象是通过 DDL 或内部调用特殊的 PL/SQL 过程来维护的，并不能简单地通过用户的事务日志来维护。

上面是 Oracle 逻辑备用数据库不支持的用户模式。除此之外，还有一些由于逻辑备用数据库不支持的数据类型而导致的不能复制的表。下面的查询给出结果：

```
SYS@NETDB>select distinct owner, table_name
from dba_logstdby_unsupported;
```

不支持的原因是由于表中存在某些字段的特殊数据类型，例如：

```
SYS@NETDB>desc dba_logstdby_unsupported;
 Name                      Type
 ------------------------------------------------
 OWNER                     VARCHAR2(30)
 TABLE_NAME                VARCHAR2(30)
 COLUMN_NAME               VARCHAR2(30)
 ATTRIBUTES                VARCHAR2(39)
 DATA_TYPE                 VARCHAR2(64)
```

其中，ATTRIBUTES 字段给出由于数据类型或结构导致的不支持的原因，具体有如下几种情况（IOT 是索引组织表 Index Organized Table 的缩写）。

※ IOT with Overflow（带溢出段的 IOT）。

※ IOT with LOB（有大对象类型的 IOT）。

※ Mapping table for physical rowid of IOT(指示 IOT 物理行地址的映射表)。

※ IOT with row movement（存在行移动的 IOT）。

※ Table Compression（表压缩）。

※ Object Table（对象表）。

※ AQ queue table（AQ 队列表）。

※ Unsupported Virtual Column（不支持的虚拟列）。

※ Encrypted Column（加密列）。

8.3.3 自定义逻辑备用维护的数据集

在实际部署逻辑备用数据库时，上述不被逻辑备用数据库维护的数据集可以根据需要进行扩展，即可以有选择地划定由逻辑备用 SQL Apply 服务所维护的数据集。

这里自定义逻辑备用维护的数据集并不影响主数据库对逻辑备用数据库的日志传输。日志传输的量仍然是 100%，只是 SQL Apply 在日志挖掘和应用服务时根据用户定义跳过某些数据库对象而已。

要自定义逻辑备用维护的数据集，使用 PL/SQL 包 DBMS_LOGSTDBY 的 SKIP 过程指定 SQL Apply 的“跳过”规则。需要注意的是，调用该过程之前必须停止 SQL Apply 服务，即执行下面的语句。

```
SQL> alter database stop logical standby apply;

PROCEDURE DBMS_LOGSTDBY.SKIP
 Argument Name            Type          In/Out Default?
 ------------------------------------------------------------
 STMT                     VARCHAR2      IN
 SCHEMA_NAME              VARCHAR2      IN     DEFAULT
 OBJECT_NAME              VARCHAR2      IN     DEFAULT
 PROC_NAME                VARCHAR2      IN     DEFAULT
 USE_LIKE                 BOOLEAN       IN     DEFAULT
 ESC                      CHAR(1)       IN     DEFAULT
```

逻辑备用数据库 SQL Apply 实际运行过程中的“跳过”行为可由视图 DBA_LOGSTDBY_SKIP 给出，其中，ERROR 字段指示该项“跳过”行为是 SKIP 规则导致（Y）还是由于执行错误导致（N）。

```
SYS@NETDB>desc dba_logstdby_skip;
 Name                                      Type
 ----------------------------------------------------
 ERROR                                     VARCHAR2(1)
 STATEMENT_OPT                             VARCHAR2(30)
 OWNER                                     VARCHAR2(30)
 NAME                                      VARCHAR2(65)
 USE_LIKE                                  VARCHAR2(1)
 ESC                                       VARCHAR2(1)
 PROC                                      VARCHAR2(98)
```

从上面的 SKIP 过程和视图 DBA_LOGSTDBY_SKIP 的结构可以看出，参数和字段是具有对应关系的，意义如下：

※ SCHEMA_NAME 或 OWNER 是指用户模式或对象属主的名称，可以使用百分号（%）通配符。

※ OBJECT_NAME 或 NAME 是指具体的数据库对象名，同样可以使用百分号（%）通配符。

※ STMT 或 STATEMENT_OPT 是指跳过执行的 SQL 语句选项，Oracle 通过一系列关键字指示符合条件的一类 SQL 语句，如 DML、SCHEMA_DDL、NON_SCHEMA_DDL、PL/SQL 等分别指示 DML 操作、模式对象的 DDL 操作、非模式的 DDL 操作（如 Alter Tablespace、Create Role 等）、执行 Oracle 提供的 PL/SQL 程序包（Package）等；还可以使用关键字，如 TABLE、INDEX、VIEW、PROCEDURE 等代表一类对象上的各种操作；还可以使用具体的 SQL 语句，如 ALTER TABLESPACE、CREATE TABLE 等代表一类 SQL 操作。

※ PROC_NAME 或 PROC 是指当满足“跳过”规则时调用的过程名称，默认值为 NULL。

※ USE_LIKE 指定 SQL Apply 是否使用 LIKE 条件进行通配符匹配或者精确匹配，默认值为 Y 或 True。

※ ESC 指定当使用 LIKE 条件进行通配符匹配时，标识一个转义符，默认值为 NULL。

实例 1：设置 SQL Apply 跳过某个表上的 DML 或 DDL 操作。

```
SYS@NLOGDB> EXECUTE DBMS_LOGSTDBY.SKIP(-
      stmt => 'DML', -
      schema_name => 'HR', -
      object_name => 'EMPLOYEES' );
SYS@NLOGDB> EXECUTE DBMS_LOGSTDBY.SKIP(-
      stmt => 'SCHEMA_DDL', -
      schema_name => 'HR', -
      object_name => 'EMPLOYEES' );
```

实例 2：设置 SQL Apply 跳过某一用户模式下所有的 DML 和 DDL 操作。

```
SYS@NLOGDB> EXECUTE DBMS_LOGSTDBY.SKIP(-
stmt => 'SCHEMA_DDL', -
schema_name => 'SCOTT', -
object_name => '%', -
proc_name => null);
SYS@NLOGDB> EXECUTE DBMS_LOGSTDBY.SKIP(-
stmt => 'DML', -
```

```
schema_name => 'SCOTT', -
object_name => '%', -
proc_name => null);
```

实例 3：设置 SKIP 跳过规则时使用存储过程转换主备用数据库端文件系统的存储路径。

（1）创建满足“跳过”规则时调用的存储过程，本例的过程是处理与表空间有关的数据文件的存储路径。

```
SYS@NLOGDB> CREATE OR REPLACE PROCEDURE sys.tbs_ddl_handler (
      old_stmt IN VARCHAR2,
      stmt_typ IN VARCHAR2,
      schema IN VARCHAR2,
      name IN VARCHAR2,
      xidusn IN NUMBER,
      xidslt IN NUMBER,
      xidsqn IN NUMBER,
      action OUT NUMBER,
      new_stmt OUT VARCHAR2 ) AS
BEGIN
-- 主数据库端的存储路径为 /primary/database/netdb
-- 对应的备用备用数据库端的存储路径为 /standby/database/logdb
      new_stmt = replace(old_stmt,
             '/primary/database/netdb',
             '/standby/database/logdb');
      action := DBMS_LOGSTDBY.SKIP_ACTION_REPLACE;
EXCEPTION
WHEN OTHERS THEN
      action := DBMS_LOGSTDBY.SKIP_ACTION_ERROR;
      new_stmt := NULL;
END tbs_ddl_handler;
```

（2）在设置 SKIP 跳过规则时注册该存储过程（满足规则时调用）。

```
SYS@NLOGDB> EXECUTE DBMS_LOGSTDBY.SKIP ( -
stmt => 'TABLESPACE', -
      proc_name => 'SYS.TBS_DDL_HANDLER');
```

8.4 判断逻辑备用不维护的表

在逻辑备用数据库中通过设置“跳过”规则来设置 SQL Apply 维护的数据集，

已经设置的“跳过”规则可以通过视图 DBA_LOGSTDBY_SKIP 查询出来。由于多种规则都可能影响到一个表是否被 SQL Apply 维护，因此，判断一个表是否被逻辑备用数据库维护，并不是一件直观的事，而是需要一系列的逻辑判断。

在此给出一个用于判断某个特定的表是否被逻辑备用维护的函数。

```
create or replace function sys.table_is_skipped (
schema_name in varchar2,
table_name in varchar2 )
return number
as
matched_count number := 0;
begin
select count(*) into matched_count from dba_logstdby_skip S
where statement_opt = 'DML' and error = 'N' and
1 = case
    when use_like = 'Y' then
       case
       when esc = 'Y' then
          case
          when schema_name like S.owner escape esc and
               table_name like S.name escape esc then 1 else 0
          end
       when esc = 'N' or esc is NULL then
          case
          when schema_name like S.owner and
               table_name like S.name then 1 else 0
          end
       end
    when use_like = 'N' then
       case
       when schema_name = S.owner and
            table_name = S.name then 1 else 0
       end
    else 0
    end;
return matched_count;
end table_is_skipped;
```

上面的函数可以判断特定的表是否由逻辑备用数据库的 SQL Apply 维护。如

果要给出整个逻辑备用数据库中所有由于 SKIP 规则而不被 SQL Apply 维护的表清单，需要遍历视图 DBA_ALL_TABLES，下面给出满足这一要求的 PL/SQL 程序。

```
SYS@LOGDB>create type tab_obj as object (
  2  schema_name varchar2(30),
  3  table_name varchar2(30) );
  4  /

Type created.

SYS@LOGDB>create type logstdby_skipped_tab as table of tab_
obj;
  2  /

Type created.

SYS@LOGDB> create or replace function sys.list_all_skiped_tabs
return logstdby_skipped_tab pipelined
as
type ref_cur is ref cursor;
     cur ref_cur;
     out_rec tab_obj :=  tab_obj(null, null);
begin
 open cur for 'select owner,table_name from dba_all_tables';
 loop
   fetch cur into out_rec.schema_name, out_rec.table_name;
   exit when cur%notfound;
   if sys.table_is_skipped(out_rec.schema_name, out_rec.table_
name)
<>0 then
       pipe row(out_rec);
   end if;
 end loop;
close cur;
return;
end list_all_skiped_tabs;
/
Function created.
```

上面的函数返回的是表类型，如果需要了解详细的逻辑备用数据库中未由 SQL Apply 维护的表清单，可以执行如下查询：

```
SYS@LOGDB>select * from table(list_all_skiped_tabs);
```

下面是一个 SKIP 测试。

```
SYS@LOGDB>EXECUTE DBMS_LOGSTDBY.SKIP(-
> stmt => 'DML', -
> schema_name => 'SCOTT', -
> object_name => '%', -
> proc_name => null);

PL/SQL procedure successfully completed.

SYS@LOGDB>select * from table(list_all_skiped_tabs);

SCHEMA_NAME                     TABLE_NAME
--------------------------------------------------
SCOTT                           DEPT
SCOTT                           EMP
SCOTT                           BONUS
SCOTT                           SALGRADE
```

如果要取消已经设置的“跳过”，可使用 DBMS_LOGSTDBY.UNSKIP 过程。

```
PROCEDURE UNSKIP
Argument Name             Type          In/Out       Default?
------------------------------------------------------------
STMT                      VARCHAR2      IN
SCHEMA_NAME               VARCHAR2      IN           DEFAULT
OBJECT_NAME               VARCHAR2      IN           DEFAULT
```

需要注意的是，直接调用该过程可以导致涉及的“跳过”规则被取消，但并不能导致涉及的表随后由逻辑备用数据库的 SQL Apply 自动维护该表。很显然，任何 DML 语句的操作都是在已有数据的基础上完成的。要使得取消的“跳过”规则涉及的表能够被 SQL Apply 维护，需要获得该表的最新快照，这就要调用 DBMS_LOGSTDBY 的如下过程：

```
PROCEDURE DBMS_LOGSTDBY.INSTANTIATE_TABLE
 Argument Name             Type          In/Out Default?
 ------------------------------------------------------------
 SCHEMA_NAME               VARCHAR2      IN
```

```
TABLE_NAME                     VARCHAR2        IN
DBLINK                         VARCHAR2        IN
```

参数 DBLINK 是备用数据库端指向主数据库的数据库链（Database Link），创建它需要指定网络服务名（Net Service Name）和用户身份认证信息。数据库链是在一个数据库会话中访问另外一个数据库中的数据库对象的手段。另需注意：修改“跳过”规则和调用 INSTANTIATE_TABLE 过程之前需要停止逻辑备用的 SQL Apply 服务。

8.5 逻辑备用故障排除实例

为了方便在数据库中查询数据字典视图，我们在主数据库的 SYS 模式下编写了如下过程：

```
SYS@ORADB>create or replace procedure list_dict(
  2  vname varchar2, gflag boolean:=false) is
  3   cursor c_tab is select t_name from (
  4    select table_name t_name from dictionary
  5    where table_name like '%'||upper(vname)||'%'
  6    union
  7    select name t_name from v$fixed_table
  8    where name like '%'||upper(vname)||'%'
  9    union
 10    select tname t_name from tab
 11    where tname like '%'||upper(vname)||'%');
 12  begin
 13   DBMS_OUTPUT.PUT_LINE(chr(10)||CHR(9)||'TABLE_NAME');
 14   DBMS_OUTPUT.PUT_LINE(RPAD('-',30,'-'));
 15   for t in c_tab loop
 16   if substr(t.t_name,1,3)='GV$' and not gflag then
 17    null;
 18   else
 19    DBMS_OUTPUT.PUT_LINE(t.t_name);
 20   end if;
 21   end loop;
 22  end list_dict;
 23  /

Procedure created.
```

为了使普通用户也能使用该过程，我们创建了同义词，并做如下授权：

```
SYS@ORADB>create or replace public synonym list_dict
  2  for sys.list_dict;
Synonym created.

SYS@ORADB>grant execute on list_dict to public;
Grant succeeded.
```

上述操作将导致逻辑备用数据库的 SQL Apply 中止，其告警日志中的信息摘录如下：

```
LOGSTDBY stmt: grant execute on list_dict to public
LOGSTDBY status: ORA-01775: looping chain of synonyms
LOGSTDBY id: XID 0x0006.02e.000001df, hSCN 0x0000.0010a57a,
lSCN 0x0000.0010a57a, Thread 1, RBA 0x01aa.0000045c.19c,
txnCscn 0x0000.0010a57d, PID 4820, P004
LOGSTDBY Apply process P004 pid=28 OS id=4820 stopped
Thu Nov 29 08:39:47 2012
Errors in file
 /oracle/product/10.2.0/db_1/rdbms/trace/logdb_lsp0_5140.trc:
ORA-12801: error signaled in parallel query server P004
ORA-01775: looping chain of synonyms

LOGSTDBY Analyzer process P003 pid=27 OS id=5168 stopped
LOGSTDBY Apply process P006 pid=30 OS id=4876 stopped
LOGSTDBY Apply process P005 pid=29 OS id=920 stopped
LOGSTDBY Apply process P008 pid=32 OS id=1672 stopped
LOGSTDBY Apply process P007 pid=31 OS id=4688 stopped
```

从以上告警信息可以看出，SQL Apply 中止的原因是由于主数据库中的权限管理指令 grant execute on list_dict to public 导致的，该操作对应于内部事务 XID 0x0006.02e.000001df，为此，首先执行如下操作：

```
SYS@LOGDB>alter database stop logical standby apply;
Database altered.

SYS@LOGDB>ALTER DATABASE START LOGICAL STANDBY APPLY
  2  SKIP FAILED TRANSACTION;
Database altered.
```

上述指令可以在逻辑备用数据库端顺利重启 SQL Apply，但很快随着日志的应用，再次出现前面同样的错误，导致 SQL Apply 再次中止。

看来逻辑备用数据库的故障是由于主数据库中的同义词导致的，我们决定取消对同义词的授权并删除该同义词，为此，分别在主备用数据库中执行了如下操作：

```
SYS@ORADB>revoke execute on list_dict from public;
Revoke succeeded.
SYS@ORADB>drop public synonym list_dict;
Synonym dropped.

SYS@LOGDB>alter database stop logical standby apply;
Database altered.
SYS@LOGDB>alter session disable guard;
Session altered.
SYS@LOGDB>drop public synonym list_dict;
Synonym dropped.
SYS@LOGDB>EXECUTE DBMS_LOGSTDBY.SKIP(stmt=>'drop public
synonym', schema_name=>'SYS', object_name=>'%');
PL/SQL procedure successfully completed.
SYS@LOGDB>alter session enable guard;
Session altered.
SYS@LOGDB>ALTER DATABASE START LOGICAL STANDBY APPLY
  2  SKIP FAILED TRANSACTION;
Database altered.
```

逻辑备用数据库端的 SQL Apply 重新启动，告警日志中产生如下输出（摘录），故障得以排除，逻辑备用数据库恢复正常。

```
ALTER DATABASE START LOGICAL STANDBY APPLY SKIP FAILED
TRANSACTION
Thu Nov 29 09:23:30 2012
with optional part
TRY TO SKIP FAILED TRANSACTION
Attempt to start background Logical Standby process
LSP0 started with pid=23, OS id=5016
LOGSTDBY status: ORA-16111: log mining and apply setting up
Thu Nov 29 09:23:30 2012
LOGMINER: Parameters summary for session# = 1
LOGMINER: Number of processes = 3, Transaction Chunk Size =
201
```

```
LOGMINER: Memory Size = 30M, Checkpoint interval = 150M
LOGMINER: session# = 1, reader process P000 started with
pid=24 OS id=3096
LSP2 started with pid=33, OS id=4368
LOGSTDBY Analyzer process P003 started with pid=27 OS id=4632
LOGSTDBY Apply process P007 started with pid=31 OS id=1420
LOGSTDBY Apply process P008 started with pid=32 OS id=4692
LOGSTDBY Apply process P006 started with pid=30 OS id=5144
Thu Nov 29 09:23:30 2012
LOGMINER: Begin mining logfile: .../ARC00427_0797957515.001
Thu Nov 29 09:23:30 2012
LOGMINER: Turning ON Log Auto Delete
Thu Nov 29 09:23:30 2012
LOGMINER: End mining logfile: .../ARC00427_0797957515.001
Thu Nov 29 09:23:30 2012
LOGMINER: Begin mining logfile: .../ARC00428_0797957515.001
Thu Nov 29 09:23:30 2012
LOGMINER: End mining logfile: .../ARC00428_0797957515.001
Thu Nov 29 09:23:30 2012
Completed: ALTER DATABASE START LOGICAL STANDBY APPLY SKIP
FAILED TRANSACTION
LOGMINER: session# = 1, preparer process P002 started with
pid=26 OS id=5232
LOGSTDBY Apply process P005 started with pid=29 OS id=2936
LOGMINER: session# = 1, builder process P001 started with
pid=25 OS id=3864
LOGSTDBY Apply process P004 started with pid=28 OS id=5644
LOGSTDBY: Skip Transaction xid 0x0006.01f.000001e0, chunk[0],
scn: 0x0000.0010a88b
LOGSTDBY: Skip Transaction xid 0x0007.002.00000160, chunk[0],
scn: 0x0000.0010aa71
LOGSTDBY: Skip Transaction xid 0x0009.022.000001c4, chunk[0],
scn: 0x0000.0010aaa3
Thu Nov 29 09:23:35 2012
LOGMINER: Log Auto Delete - deleting: .../ARC00427_0797957515.001
Deleted file .../ARC00427_0797957515.001
```

从上面的故障排除案例中，我们切实地体会到，Oracle 的逻辑备用数据库的确存在有待完善的地方，某些操作（特别是 DDL、DCL 操作）在主数据库中可以正常使用，

但在逻辑备用数据库中却导致 SQL Apply 的异常终止，这对于 DBA 管理和维护备用数据库需要特别引起注意。

如果在主数据库中执行 create tablespace 和 drop tablespace 等语句，操作通过日志传播到逻辑备用数据库端同样会导致 SQL Apply 中止，因为检测通不过。另外，对于逻辑备库，db_file_name_convert 参数对此类指令的操作也是无效的。首先，可以把这两个 DDL 语句给忽略（skip）掉，然后，在备用端直接再创建或删除同名的表空间，最后，再启动 SQL Apply 应用：

```
alter database stop logical standby apply;
EXECUTE DBMS_LOGSTDBY.SKIP(stmt=>'CREATE TABLESPACE',
 schema_name=>'SYS', object_name=>'%');
EXECUTE DBMS_LOGSTDBY.SKIP(stmt=>'DROP TABLESPACE',
 schema_name=>'SYS', object_name=>'%');
alter session disable guard;
create tablespace userdata2 datafile ...;
drop tablespace userdata1 including contents and datafiles
cascade constraints;
alter session enable guard;
alter database start logical standby apply immediate skip
failed transaction;
```

第9章

物理备用数据库的新特性

Data Guard 方案中的备用数据库特性在 Oracle 的版本升级过程中不断地得到完善和扩充，如在 9i 中首次引入保护模式的概念、逻辑备用数据库的概念等，在 10g 中增加了物理备用和逻辑备用的实时应用概念，在 11g 中则引入了针对物理备用数据库的 Active Data Guard 特性。

从备用数据库数据更新的角度来看，物理备用比逻辑备用更完善，因为它没有数据类型的限制。但物理备用数据库在正常运行的情况下处于加载（Mount）状态，因此，它的数据访问受到限制，在最新的 Data Guard 方案中，这一限制得到了突破。本章主要介绍针对物理备用数据库最新的几个重要变化。

9.1 物理备用的只读模式

物理备用数据库正常情况下处于托管恢复模式，实例处于 Mount 状态，因此，用户是不能直接访问物理备用数据库中的数据的。要访问物理备用数据库中的数据，需要停止数据库的托管恢复模式，以只读方式打开。

物理备用库以只读方式打开后，可以继续接收来自主数据库的事务日志，只是停止了 Redo 应用。

要从只读方式的打开状态转换到正常的托管恢复模式，只需直接启动 Redo 应用即可，当然也可以关闭数据库、启动到加载状态，再启动托管恢复。

```
SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.

SYS@PHYDB>alter database open [read only];
```

```
Database altered.

SYS@PHYDB>select open_mode from v$database;

OPEN_MODE
----------
READ ONLY

SYS@PHYDB>select status  from v$instance;

STATUS
------------
OPEN

SYS@PHYDB>alter database recover managed standby database
disconnect;

Database altered.

SYS@PHYDB>select status  from v$instance;

STATUS
------------
MOUNTED
```

9.2 物理备用的读 / 写模式

实际上，物理备用数据库可以暂时性地以读 / 写模式打开，这样，物理备用数据库就暂时脱离了主数据库（停止了 Redo 应用和日志接收），以一个独立的数据库在运行。要保证以读 / 写模式打开后还能够恢复到原先的物理备用状态，这需要一定的条件准备和后续处理。下面以 10g 的环境说明这样的过程。

9.2.1 打开前的条件准备

要暂时性地以读 / 写模式打开物理备用数据库，并且要确保之后能够恢复到物理备用状态，需要如下准备工作。

1．设置闪回恢复区并启动数据库闪回功能

闪回恢复区对于数据库闪回功能是必需的，因为闪回日志只能位于闪回恢复区（又称快速恢复区）。

```
SYS@PHYDB>show parameters db_recovery

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------
db_recovery_file_dest                string      +DG_FRA
db_recovery_file_dest_size           big integer 1G
SYS@PHYDB>select flashback_on from v$database;

FLASHBACK_ON
------------------
NO

SYS@PHYDB>alter database flashback on;
Database altered.

SYS@PHYDB>select flashback_on from v$database;

FLASHBACK_ON
------------------
YES
```

2．停止 Redo 应用并创建保证还原点

这里创建的还原点是要确保物理备库在经历读 / 写模式打开后还能够返回到此处的状态。

```
SYS@PHYDB>alter database recover managed standby database
disconnect;

Database altered.

SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.

SYS@PHYDB>create restore point before_read_write guarantee
flashback database;

Restore point created.
```

```
SYS@PHYDB>select name,scn,guarantee_flashback_database
from v$restore_point;

NAME                          SCN        GUA
----------------------------- ----------  ---
BEFORE_READ_WRITE             757562      YES
```

3．切换日志

在主数据库上切换日志，确保还原点（对应的 SCN）时刻的日志由备用日志归档到归档日志中。因为备用日志是备用数据库专用的，一旦物理备用数据库以读 / 写方式打开，实际上是一个独立的数据库，在将来利用还原点还原数据库时不能使用备用日志。因此，当使用备用日志时，此步骤是不能省略的。

```
SYS@NETDB>alter system switch logfile;

System altered.
```

或者

```
SYS@NETDB>alter system archive log current;

System altered.
```

9.2.2 激活物理备用库

以读 / 写模式打开物理备用数据库本质上是要激活物理备用数据库以一个独立的数据库运行。

```
SYS@PHYDB>select database_role from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY

SYS@PHYDB>alter database activate standby database;

Database altered.

SYS@PHYDB>select database_role from v$database;
```

```
DATABASE_ROLE
----------------
PRIMARY

SYS@PHYDB>startup mount force;
ORACLE instance started.

Total System Global Area  125829120 bytes
Fixed Size                  1247660 bytes
Variable Size              62916180 bytes
Database Buffers           58720256 bytes
Redo Buffers                2945024 bytes
Database mounted.
SYS@PHYDB>alter database set standby database to maximize
performance;

Database altered.

SYS@PHYDB>alter database open;

Database altered.
SYS@PHYDB>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     1
Next log sequence to archive   2
Current log sequence           2
```

至此，原来的物理备用数据库转换为独立的、以读 / 写模式运行的数据库。

以读 / 写模式打开的物理备用数据库完全脱离了主数据库（停止接收主数据库的事务日志），它可以用于对主数据库的某种测试或另外一些独立的数据处理。

9.2.3　返回到物理备用状态

在经历了读 / 写操作后，下面再将数据库转换到物理备用状态，并实现与主数据库的同步。这里需要特别注意如下几点：

（1）一旦由读 / 写模式转换到物理备用状态，读 / 写模式期间的所有数据更改将丢失。

（2）要实现与主数据库的再次同步，主数据库在物理备库处于读 / 写模式期间产生的所有事务日志必须连续存在。

（3）如果物理备用数据库处于读 / 写模式期间过长，因保留策略原因，主数据库中的某些归档日志已经被删除，这种情况下要实现与主数据库的同步，就需要 DBA 额外的手工干预。

```
SYS@PHYDB>startup mount force;
ORACLE instance started.

Total System Global Area  125829120 bytes
Fixed Size                  1247660 bytes
Variable Size              62916180 bytes
Database Buffers           58720256 bytes
Redo Buffers                2945024 bytes
Database mounted.
SYS@PHYDB>flashback database to restore point before_read_
write;

Flashback complete.

SYS@PHYDB>alter database convert to physical standby;

Database altered.

SYS@PHYDB>archive log list;
ORA-01507: database not mounted
SYS@PHYDB>startup mount force;
ORACLE instance started.

Total System Global Area  125829120 bytes
Fixed Size                  1247660 bytes
Variable Size              62916180 bytes
Database Buffers           58720256 bytes
Redo Buffers                2945024 bytes
Database mounted.
SYS@PHYDB>select database_role from v$database;

DATABASE_ROLE
----------------
```

```
PHYSICAL STANDBY

SYS@PHYDB>alter database recover managed standby database
disconnect;

Database altered.

SYS@PHYDB>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     145
Next log sequence to archive   0
Current log sequence           145
```

9.2.4　处理主库的日志中断

上一节说明，如果物理备用数据库的读 / 写模式打开期间较长，主数据库中的归档日志因保留策略原因已经被删除，这种情况将导致主数据库的归档日志从物理备用数据库还原点开始不连续，上面的操作不能使转换到物理备用状态的数据库与主数据库同步。

下面的实验步骤模拟处理这种情形。

（1）取消物理备用数据库的托管恢复。

```
SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.
```

（2）创建保证还原点，并记录下对应的 SCN 号。

```
SYS@PHYDB>create restore point before_read_write2
guarantee flashback database;

Restore point created.

SYS@PHYDB>select name,scn,guarantee_flashback_database
  2  from v$restore_point;
```

```
NAME                    SCN        GUA
----------------------- ---------- -----
BEFORE_READ_WRITE2      904755     YES
```

（3）在主数据库中切换日志，使得物理备用数据库中创建还原点（对应的SCN）的事务日志归档到归档日志中，否则存在于物理备用数据库的备用日志中，这会导致后面的闪回操作无法进行（闪回操作不能利用备用日志）。

```
SYS@NETDB>alter system switch logfile;

System altered.
```

（4）激活物理备用数据库。

```
SYS@PHYDB>alter database activate standby database;

Database altered.

SYS@PHYDB>select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY
SYS@PHYDB>startup mount force;

SYS@PHYDB>alter database open;

Database altered.

SYS@PHYDB>select database_role from v$database;

DATABASE_ROLE
----------------
PRIMARY
```

该数据库可作为独立的数据库操作。

下面是再次还原为物理备用数据库的过程。

（1）在 mount 状态下将数据库还原到之前创建的还原点。

```
SYS@PHYDB>shutdown
Database closed.
Database dismounted.
```

```
ORACLE instance shut down.
SYS@PHYDB>startup mount;
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size              96471928 bytes
Database Buffers           62914560 bytes
Redo Buffers                6627328 bytes
Database mounted.
SYS@PHYDB>flashback database to restore point before_read_
write2;

Flashback complete.
```

（2）将数据库重新转换到物理备用模式下运行。

```
SYS@PHYDB>alter database convert to physical standby;

Database altered.

SYS@PHYDB>select database_role from v$database;
select database_role from v$database
                          *
ERROR at line 1:
ORA-01507: database not mounted


SYS@PHYDB>startup mount force;
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size              96471928 bytes
Database Buffers           62914560 bytes
Redo Buffers                6627328 bytes
Database mounted.
SYS@PHYDB>select database_role from v$database;

DATABASE_ROLE
```

```
----------------
PHYSICAL STANDBY
```

（3）启动物理备用数据库的托管恢复。

```
SYS@PHYDB>alter database recover managed standby database
disconnet;

Database altered.
```

（4）由于主数据库缺乏必要的归档日志，托管恢复被迫停顿。经过有限的托管恢复后，查询此时的 SCN 号。在本例的操作中此时的 SCN 号就是前面创面的保证还原点对应的 SCN。

```
SYS@PHYDB>select database_role, current_scn from v$database;

DATABASE_ROLE    CURRENT_SCN
---------------- -----------
PHYSICAL STANDBY      904755
```

（5）取消物理备用数据库的托管恢复。

```
SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.
```

（6）根据前面物理备用数据库停留的 SCN 号在主数据库中执行增量备份。

```
SYS@NETDB>host rman target /

Recovery Manager: Release 10.2.0.1.0 - Production on Sat Feb
16 22:34:08 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: NETDB (DBID=3949937918)

RMAN> backup incremental from scn 904755 database
2> format '/db_fra/inc_from_scn_%U.rmn';

Starting backup at 16-FEB-13
```

```
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=19 device type=DISK
backup will be obsolete on date 23-FEB-13
archived logs will not be kept or backed up
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=/DATABASE/NETDB/SYSNETDB.
DBF
input datafile file number=00002 name=/DATABASE/NETDB/AUXNETDB.
DBF
input datafile file number=00004 name=/DATABASE/NETDB/EXMNETDB.
DBF
input datafile file number=00003 name=/DATABASE/NETDB/UNDNETDB.
DBF
input datafile file number=00005 name=/DATABASE/NETDB/USRNETDB.
DBF
channel ORA_DISK_1: starting piece 1 at 16-FEB-13
channel ORA_DISK_1: finished piece 1 at 16-FEB-13
piece handle=/DB_FRA/INC_FROM_SCN_0HO25859_1_1.RMN
tag=TAG20130216T223520 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:15

using channel ORA_DISK_1
backup will be obsolete on date 23-FEB-13
archived logs will not be kept or backed up
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
channel ORA_DISK_1: starting piece 1 at 16-FEB-13
channel ORA_DISK_1: finished piece 1 at 16-FEB-13
piece handle=/DB_FRA/INC_FROM_SCN_0IO2585O_1_1.RMN
tag=TAG20130216T223520 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 16-FEB-13
```

（7）在主数据库中创建备用控制文件。

```
RMAN> backup current controlfile for standby
2> format '/DB_FRA/ctl_stadby.ora';

Starting backup at 17-FEB-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=105 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including standby control file in backup set
channel ORA_DISK_1: starting piece 1 at 17-FEB-13
channel ORA_DISK_1: finished piece 1 at 17-FEB-13
piece handle=/DB_FRA/CTL_STADBY.ORA tag=TAG20130217T085326
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 17-FEB-13
```

（8）在物理备用库中还原备用控制文件。

```
SYS@PHYDB>host rman target /

Recovery Manager: Release 10.2.0.1.0 - Production on Sun Feb
17 08:58:00 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: NETDB (not mounted)

RMAN> restore standby controlfile from '/DB_FRA/CTL_STADBY.
ORA';

Starting restore at 17-FEB-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=10 device type=DISK
```

```
channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
output file name=/DATABASE/PHYDB/CTLPHYDB.ORA
Finished restore at 17-FEB-13
```

（9）在物理备用数据库中登记增量备份结果（备份片）。

```
RMAN> catalog backuppiece
 '/DB_FRA/INC_FROM_SCN_0HO25859_1_1.RMN';

cataloged backup piece
backup piece handle=/DB_FRA/INC_FROM_SCN_0HO25859_1_1.RMN
RECID=13 STAMP=807613500

RMAN> catalog backuppiece '/DB_FRA/INC_FROM_SCN_0IO2585O_1_1.
RMN';

cataloged backup piece
backup piece handle=/DB_FRA/INC_FROM_SCN_0IO2585O_1_1.RMN
RECID=14 STAMP=807613521
```

（10）利用最新的增量备份恢复数据库。

```
RMAN> recover database noredo;

Starting recover at 17-FEB-13
using channel ORA_DISK_1
...
Finished recover at 17-FEB-13
```

（11）清除物理备用数据库的联机日志组。

```
SYS@PHYDB>alter database clear logfile group 1;

Database altered.

SYS@PHYDB>alter database clear logfile group 2;

Database altered.
```

（12）在物理备用数据库中重新启动托管恢复。

```
SYS@PHYDB>alter database recover managed standby database
disconnect;

Database altered.

SYS@PHYDB>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      152
Next log sequence to archive    0
Current log sequence                  153
```

9.3 快照备用数据库

通过前面的实验我们可以了解到，在 Oracle 10g 中，物理备用数据库结合闪回数据库功能的还原点，可以临时性地以读 / 写方式打开（此时不能接收来自主数据库的事务日志，因为此时的数据库角色为 Primary），然后可以再转换到物理备用的状态，但这个过程有些烦琐。Oracle 11g 克服了这个烦琐的过程，并有效地扩展了此功能，这就是快照备用数据库（Snapshot Standby Database）。

快照备用数据库是 Oracle 11g 的一个新特性，它是一个完全可读 / 写的物理备用数据库，这一特性极大地扩展了 Data Guard 备用数据库方案的应用价值。快照备用数据库是物理备用数据库的一个特定状态（通常是临时性地用于报表或应用测试等），它只能由物理备用数据库转换而来。

令人欣喜的是，快照备用数据库可以接收和归档来自主数据库的重做数据（但不应用它们）。一旦快照备用数据库被转换回物理备用数据库，在丢弃对快照备用数据库的所有本地更新后，将应用从主数据库接收的重做数据。

```
SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.

SYS@PHYDB>select database_role,open_mode,flashback_on
from v$database;

DATABASE_ROLE    OPEN_MODE            FLASHBACK_ON
---------------- -------------------- ------------------
PHYSICAL STANDBY MOUNTED              NO
```

```
SYS@PHYDB>col name for a20
SYS@PHYDB>select name,scn,guarantee_flashback_database
from v$restore_point;

no rows selected

SYS@PHYDB>alter database convert to snapshot standby;

Database altered.

SYS@PHYDB>select name,scn,guarantee_flashback_database
from v$restore_point;

NAME                                              SCN      GUA
------------------------------------------------ -------- ------
SNAPSHOT_STANDBY_REQUIRED_02/17/2013 11:17:18  922554   YES

SYS@PHYDB>select database_role,open_mode,flashback_on
from v$database;

DATABASE_ROLE    OPEN_MODE            FLASHBACK_ON
---------------- -------------------- ------------------
SNAPSHOT STANDBY MOUNTED              RESTORE POINT ONLY

SYS@PHYDB>alter database open;

Database altered.

SYS@PHYDB>select database_role,open_mode,flashback_on
from v$database;

DATABASE_ROLE    OPEN_MODE            FLASHBACK_ON
---------------- -------------------- ------------------
SNAPSHOT STANDBY READ WRITE           RESTORE POINT ONLY
```

此时，可以对备用数据库执行需要的读 / 写操作。

下面的操作是将处于读 / 写模式的备用数据库再次返回到物理备用数据库的正常状态。

```
SYS@PHYDB>startup mount force;
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size              96471928 bytes
Database Buffers           62914560 bytes
Redo Buffers                6627328 bytes
Database mounted.
SYS@PHYDB>alter database convert to physical standby;

Database altered.

SYS@PHYDB>select database_role,open_mode,flashback_on
from v$database;
select database_role,open_mode,flashback_on from v$database
                                                       *
ERROR at line 1:
ORA-01507: database not mounted

SYS@PHYDB>startup mount force;
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size              96471928 bytes
Database Buffers           62914560 bytes
Redo Buffers                6627328 bytes
Database mounted.
SYS@PHYDB>select database_role,open_mode,flashback_on
from v$database;

DATABASE_ROLE    OPEN_MODE                    FLASHBACK_ON
---------------- -------------------- ------------------
PHYSICAL STANDBY MOUNTED                      NO
```

通过上面的实验过程，关于快照备用数据库我们需要注意如下两个方面：

（1）快照备用数据库的内在机制是通过保证还原点实现的，但 Oracle 并不要

求备用数据库必须启动闪回数据库功能。不过建议最好还是启动闪回数据库功能，这样可以充分利用闪回特性，在快照备用数据库读 / 写期间可以实现更灵活的数据访问。

（2）一旦将快照备用数据库再次转换为物理备用数据库状态，快照备用数据库期间的数据变更就会丢失（因为在其内部隐式地回到快照备用时刻的还原点），重新回到由主数据库的托管恢复状态。

9.4　Active Data Guard

9.4.1　备用数据库的读与写

物理备用数据库典型地处于托管恢复模式，实例处于 mount 状态。物理备用数据库可以根据应用的需要，以只读方式、读 / 写方式、快照备用方式打开，还可以以 Active Data Guard 方式打开，后两种方式都是 11g 的新增功能。

所谓以 Active Data Guard 方式打开备用数据库，指的是物理备用数据库在只读打开的情况下进行托管恢复，即备用数据库实例处于 open 状态、Read Only 方式，并且继续接收来自于主数据库的事务日志，启动 Redo Apply 服务，并且可以是实时的 Redo 应用。Active Data Guard 特性可以使得从物理备用数据库获得与主数据库完全同步的数据。

如图 9-1 所示为物理备用数据库各种状态下的比较。

表 9-1　物理备用数据库各种状态下的比较

物理备用库	实例状态	日志接收	日志应用	读 / 写模式
远程归档	Mount	Received	N/A	N/A
托管恢复	Mount	Received	Applying	N/A
只读模式	Open	Received	Stopped	Read Only
读 / 写模式	Open	Rejected	N/A	Read Write
快照备用	Open	Received	Stopped	Read Write
活动备用	Open	Received	Applying	Read Only

9.4.2　实现活动备用数据库

Oracle 从 11.0.0 版本开始支持物理备用数据库的 Active Data Guard 特性。启动 Active Data Guard 特性的步骤非常简洁：以只读方式打开物理备用数据库并启动 Redo Apply 应用服务。

物理备用数据库可通过如下两种途径启动 Active Data Guard 特性：

（1）从物理备用数据库处于关闭状态（非异常关闭）开始。

（2）物理备用数据库处于正常的托管服务（实例处于 mount 状态）期间。

第一种情况下启动 Active Data Guard 特性的步骤如下：

```
SYS@PHYDB>startup [open read only];
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size              96471928 bytes
Database Buffers           62914560 bytes
Redo Buffers                6627328 bytes
Database mounted.
Database opened.

SYS@PHYDB>alter database recover managed standby database
cancel;
alter database recover managed standby database cancel
*
ERROR at line 1:
ORA-16136: Managed Standby Recovery not active

SYS@PHYDB>alter database recover managed standby database
  2  using current logfile disconnect;

Database altered.
```

注意：物理备用在以只读方式打开后，要转换到活动备用数据库状态，此处的启动托管恢复的指令中必须指定 using current logfile 选项，否则会出现如上所述的 ORA-16136 错误。

```
SYS@PHYDB>select database_role,open_mode from v$database;

DATABASE_ROLE            OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY   READ ONLY WITH APPLY
```

第二种情况下启动 Active Data Guard 特性的步骤如下：

```
SYS@PHYDB>select database_role,open_mode from v$database;

DATABASE_ROLE      OPEN_MODE
```

```
---------------- --------------------
PHYSICAL STANDBY   MOUNTED

SYS@PHYDB>alter database recover managed standby database
cancel;

Database altered.

SYS@PHYDB>alter database open read only;

Database altered.

SYS@PHYDB>alter database recover managed standby database
  2  using current logfile disconnect;

Database altered.

SYS@PHYDB>select database_role,open_mode from v$database;

DATABASE_ROLE             OPEN_MODE
---------------- --------------------
PHYSICAL STANDBY   READ ONLY WITH APPLY
```

9.4.3　活动备用数据库应用

Active Data Guard 特性极大地增强了物理备用数据库的应用价值。与集群数据库系统（如 Oracle 的 Real Application Cluster（RAC）数据库）相对照，它从另一个角度扩展了 Oracle 数据库的分布式数据处理能力。利用活动备用数据库，至少可以实现如下几个方面的应用。

1．分离只读、实时访问操作至活动备用数据库

活动备用数据库中的数据可以实现与主数据库同步更新与访问，这样，在实践中就可以从主数据库中将那些只读操作，如复杂的查询、统计与报表操作移动至活动备用数据库，并且可以获得主数据库的实时数据，大大降低了主数据库的访问负荷。

另外，利用活动备用数据库可以实现任意时段的数据库的物理备份，即将 RMAN 备份移动至活动备用数据库。不仅如此，活动备用数据库还可以启动块改变跟踪（Block Change Tracking）功能，这样可以显著提高通过 RMAN 实现增量备份的效率。通过如下方式启动块改变跟踪功能。

```
SYS@PHYDB>alter database enable block change tracking
  2  using file '/db_fra/db_block_change.trc';
Database altered.
```

2．多备用数据库的分布式只读、实时访问

Data Guard 支持多个备用数据库的配置，将生产数据库的多个不同的物理备用数据库配置成 Active Data Guard 数据库，并且将这些活动备用数据库分布于网络的不同位置，从而轻松实现分布式的只读、实时访问，如图 9-1 所示为 Active Data Guard Reader Farm 架构。

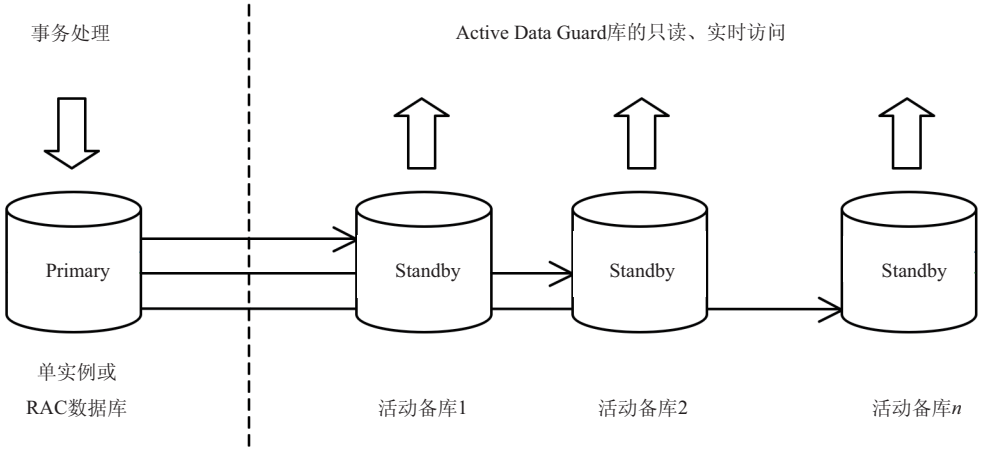


图 9-1 Active Data Guard Reader Farm 架构

这种情形可以看作 RAC 形式备用数据库的扩展。RAC 数据库是一库多实例的形式，而此处的这类应用实现了物理备用数据库的多库多实例形式。很显然，这类应用形式克服了 RAC 数据库的共享物理存储、缓存融合所带来的额外开销。

第10章

角色切换与故障转移

本章介绍Data Guard的角色切换（Switch Over）与故障转移（Fail Over），一个是计划内的系统维护，另一个是计划外的故障处理。Data Guard的核心就是通过备用数据库来实现对主数据库的数据保障，因此，备用数据库要能够在必要的时候代替或替换主数据库运行，成为实施Data Guard的关键。Data Guard中的备用数据库始终是主数据库的候选或替补，角色切换与故障转移正是备用数据库发挥主数据库功能的两种方式。

10.1 角色切换与故障转移技术概要

10.1.1 角色切换（Switch Over）

Data Guard中的角色切换是主数据库和备用数据库角色互换的过程，并且在角色互换过程中不会丢失任何数据，因为在系统允许角色切换前，主数据库必须停止所有的事务处理，停止重做数据的产生。

切换工作从主数据库开始，首先，主数据库的重做生成被终止，并为每个线程（Thread）归档当前事务日志。然后，在每个线程的下一个日志序列的开始放置一个特殊标记EOR（End of Redo），再次归档联机日志，并将最后这个序列的日志发送到备用数据库。

在角色切换过程中，备用数据库收到含有EOR标记的重做数据至关重要，因为在没有收到EOR重做数据的情况下，切换工作将无法完成。不论是物理备用数据库还是逻辑备用数据库，本质上备用端的工作是从收到日志流中的EOR标记开始。

10.1.2 故障转移（Fail Over）

故障转移是在主数据库发生故障的情况下使用某个备用数据库代替主数据库的过程。与计划内的角色切换相比，故障转移是在某些意外情况下的被迫行为，当然DBA应该尽可能实现零数据丢失，但这并不排除产生数据丢失的可能，特别是在最

高性能和最高可用性保护模式下。在最大保护模式下，故障转移不会导致数据丢失。

正常的角色切换是从主数据库开始的，但故障转移不可能从主数据库开始，因为如果主数据库可用，为什么还要实施故障转移呢？因此，这里的故障转移不涉及主数据库。当然，在开始故障转移之前，选择的备用数据库还是要尽可能地应用所有已接收到的、可以利用的重做数据。

故障转移本质上可以看作在主数据库缺位的情况下的模拟角色切换，这时候备用数据库端的 EOR 标记是被人为添加到当前备用日志文件的开始部分，并如同该标记来自于主数据库一样。

10.1.3 角色切换注意事项

不论是角色切换还是故障转移，需要注意如下几点：

（1）在实施角色切换前，备用数据库应该应用所有可以应用的事务日志，包括没有及时传输到备用库、在主数据库中遗留的可用的联机日志或归档日志。

（2）在实施角色切换前，需要将 Data Guard 的保护模式调整为“最大性能”模式。

（3）如果在一个主数据库、多个备用数据库的 Data Guard 环境，不论是角色切换还是故障转移，应该尽可能选择物理备用数据库切换为主数据库，这样，其他逻辑备用数据库还是 Data Guard 环境的一部分。如果选择逻辑备用数据库切换为主数据库，则其他物理备用数据库、原先的主数据库都不能是 Data Guard 环境的组成部分，新的逻辑备用数据库需要重新构建。

（4）如果主数据库或备用数据库是 RAC 数据库，应该关闭其他实例，仅保留一个实例运行。角色切换完毕后再启动其他实例。

（5）为了实现快速和高效地实现角色切换，应该断开所有的其他用户连接。如果不能断开某些用户连接，应该尽可能地限制用户的数据库活动。

（6）选择物理备用数据库切换为主数据库时，需要确保临时表空间有对应的临时数据文件，并且要确保临时数据文件物理地存在。

（7）Data Guard 环境的角色切换从 Primary 数据库开始，主数据库必须处于打开状态，且主数据库的 SWITCHOVER_STATUS（视图 V$DATABASE）必须是 TO STANDBY，否则，不能实施角色切换。

（8）在涉及物理备用数据库的 SWITCHOVER 中，物理备用数据库必须启动 Redo Apply；在涉及逻辑备用数据库的 SWITCHOVER 中，逻辑备用数据库必须启动 SQL Apply。

10.2　故障转移与数据损失

计划内的角色切换不会导致业务数据的损失。故障转移则不同，Data Guard 在最高性能和最高可用性保护模式下，往往会导致或多或少的数据损失，这与多种因素有关，如日志传输模式（ASYNC 或 SYNC）、主数据库故障后其日志系统是否可用、备用端是否使用备用日志（Standby Log）、主数据库是单实例还是 RAC 等。

备用数据库端使用备用重做日志，在主数据库出现故障的情况下可减少备用端的数据损失。原因在于，当备用端不使用备用重做日志的情况下，Data Guard 会把传入的重做数据直接写入备用数据库的归档日志文件。当某个归档日志文件的记录由于主数据库的故障导致中途终止时，部分归档日志往往并不能注册到备用数据库，从而导致备用数据库无法应用。另外，如果应用中途终止的部分归档日志文件，有可能损坏备用数据库，因此，在大多数 Oracle 版本中不会保留这样的部分归档日志。

如果主数据库是 RAC 数据库，则在备用数据库端接收的多线程的日志需要进行线程合并，只有执行线程合并后的日志才能在备用端应用。但由于主数据库 RAC 多实例的线程日志在合并时有一个同步问题，这会导致某些实例传输到备用端的日志有可能不能应用，从而导致部分的数据损失。

这里需要注意的问题有两个：一是多线程的日志合并并不依赖于日志序列号，因为 RAC 数据库的不同实例有自己的日志序列号，不同实例的日志序列号并不连续、也不相互依赖，合并的依据只能是按照数据库内部的 SCN 号；二是如果备用数据库是 RAC 数据库，在运行过程中只有一个实例执行多线程的日志合并和应用。

在 RAC 数据库中不同实例之间存在一个间隔为 6 秒的重做心跳信息（该心跳间隔有可能在未来的版本中发生变化），这样即使在理想的情况下，RAC 主数据库的故障也可能会导致备用数据库 6 秒的数据损失。

10.3　数据库闪回与 Data Guard

实施 Data Guard 并不要求必须启动数据库闪回功能，即使在使用快照备用数据库（Snapshot Standby Database）的情况下也是这样。但启动数据库闪回功能会增加 Data Guard 维护、故障转移等方面的灵活性，建议 Data Guard 环境中的主数据库和备用数据库根据需要启动数据库闪回功能。

在 Data Guard 环境中，当主数据库由于某种原因需要执行数据库闪回操作闪回到过去的某个时间点时，必须停止所有备用数据库的应用进程。在数据库闪回后，一旦使用 Resetlogs 选项打开主数据库，之后必须将所有的备用数据库闪回到主数据库的闪回点之前。当主数据库的重做数据开始重新传输时（日志序列号从 1 开始），备用数据库在应用原先接收的事务日志回到主数据库的闪回点后，然后才能开始处理新的日志流的应用。

在 Data Guard 故障转移的过程中，一旦某个物理备用数据库转换为主数据库后，原先的主数据库就不再是 Data Guard 环境的一部分。此时，如果要完全重建新的备用数据库，往往需要消耗较大的资源和代价。如果出现故障的主数据库的日志系统存在，可以使用数据库闪回技术将其闪回到过去某个时间点，并且将其改造为新的备用数据库，使其重新回到 Data Guard 环境中。下面给出参考步骤。

在故障转移后的主数据库中，执行下面的查询获得备用数据库转换为主数据库时刻对应的 SCN 号。

```
SQL> Select standby_became_primary_scn from v$database;
```

在原来的主数据库中执行如下操作：

（1）启动至加载状态，并执行闪回操作，使其闪回到过去的某个时刻。当然最理想的是上面查询给出的 SCN 点。

```
SQL> startup mount
SQL> flashback database to scn <SCN number>;
```

（2）在原来的主数据库上创建备用控制文件：

```
SQL> alter database create standby controlfile as '/db/...';
```

或者

```
RMAN> backup current controlfile for standby format '/db/...';
```

对应地，还原备用控制文件：

```
RMAN> restore standby controlfile from '/db/...';
```

（3）使用新的备用控制文件启动实例至 mount 状态，这就为从新的主数据库接收远程日志做好准备。

接下来的步骤是在新的主数据库上设置指向原来主数据库（此时为备用数据库）的远程归档参数。

（4）在新的备用数据库（原先的主数据库）中启动托管恢复。

```
SQL> alter database recover managed standby database
disconnect;
```

（5）可选。等新的备用数据库与新的主数据库同步后，如果必要，可以执行角色切换，以便 Data Guard 中的主数据库再次回到原来的主机。

10.4 闪回日志导致的数据库故障

Oracle 数据库一旦启动了数据库闪回功能，数据库会按照一定的间隔记录闪回日志，并且闪回日志只能位于快速恢复区（Flash Recovery Area），DBA 不能

将其备份至其他地方，数据库会根据参数 db_flashback_retention_target 决定闪回日志是否保留。

下面的数据库故障是由于人工删除数据库闪回日志导致数据库不能正常启动，数据库不能打开至 open 状态。为此，我们首先将实例启动至 mount 状态，然后再打开。故障的现象如下：

```
SYS@BJING>alter database open;
alter database open
*
ERROR at line 1:
ORA-38760: This database instance failed to turn on flashback
database

SYS@BJING>alter database flashback off;

Database altered.

SYS@BJING>alter database open;
alter database open
*
ERROR at line 1:
ORA-38760: This database instance failed to turn on flashback
database
```

错误 ORA-38760 提示数据库不能开启闪回功能，先关闭闪回功能，然后再打开，故障依旧。查看数据库跟踪文件，发现如下信息：

```
ORA-38701: Flashback database log 11 seq 58 thread 1:
"+GFRA/BJING/FLASHBACK/O1_MF_8KPK1SJM_.FLB"
ORA-27041: unable to open file
OSD-04002: unable to open file
O/S-Error: (OS 3) 系统找不到指定的路径。
```

此处明确说明数据库在打开时需要读取闪回日志文件 O1_MF_8KPK1SJM_.FLB，但该文件不存在，因为此前已手工删除。从序列号上看，此数据库闪回日志文件是一个较早的闪回日志。可数据库为什么要打开时读取这么一个较早的闪回日志呢？由此联想到数据库可能有保证还原点（Guaranteed Restore Point）的存在。下面的查询也验证了这一点。

```
SYS@BJING>select flashback_on from v$database;
```

```
FLASHBACK_ON
------------------
RESTORE POINT ONLY
```

数据库的当前闪回状况是 RESTORE POINT ONLY，它指示数据库闪回功能虽然没有打开，但启动了闪回到保证还原点。进一步查看还原点的信息。

```
SYS@BJING>select name,guarantee_flashback_database, preserved,
2     to_char(time,'YYYY-MM-DD HH24:MI:SS') time
3     from v$restore_point;

NAME                      GUA PRE TIME
------------------------- --- --- -------------------------------
BEFORE_SWITCH2LOGICAL     YES YES 2013-02-20 14:18:16
```

至此，数据库故障原因查明，由于保证还原点的存在，导致数据库在启动过程中要访问相应的闪回日志。排除故障的方法是删除保证还原点。

```
SYS@BJING>drop restore point before_switch2logical;

Restore point dropped.

SYS@BJING>select flashback_on from v$database;

FLASHBACK_ON
------------------
NO
```

再次启动数据库闪回功能（可选），正常打开数据库，数据库故障被排除。

```
SYS@BJING>alter database flashback on;

Database altered.

SYS@BJING>select flashback_on from v$database;

FLASHBACK_ON
------------------
YES

SYS@BJING>alter database open;
Database altered.
```

10.5　角色切换前的准备工作

10.5.1　检查日志传输

如果能够利用主数据库，可以查询视图 V$ARCHIVE_DEST_STATUS 确定对应于备用数据库的归档目的地的日志传输是否同步。

```
SYS@BJING>select db_unique_name,protection_mode,
  2  synchronization_status, Synchronized
  3  from v$archive_dest_status where dest_id=2;

DB_UNIQUE_NAME   PROTECTION_MODE   SYNCHRONIZATION_STATUS  SYN
-------------------------------------------------------
SHHAI            MAXIMUM AVAILABILITY       OK            YES

SYS@BJING>select db_unique_name,protection_mode,
  2  synchronization_status, Synchronized
  3  from v$archive_dest_status where dest_id=3;

DB_UNIQUE_NAME    PROTECTION_MODE        SYNCHRONIZATION_STATUS
SYN
-------------------------------------------------------------
JINAN                MAXIMUM PERFORMANCE   CHECK CONFIGURATION
NO
```

上述查询可以了解到主数据库的两个归档目的地分别指向物理备用数据库和逻辑备用数据库，物理备用数据库的日志传输已经同步，逻辑备用数据库接收的日志还没有同步。SYNCHRONIZATION_STATUS 列显示 CHECK CONFIGURATION，这是由于测试用主数据库的备用归档目的地的日志传输方式配置为 ASYNC，如果配置为 SYNC，则显示为 OK 状态。如果 SYNCHRONIZED 列显示为非 YES 状态，则可以进一步在备用端检查用于日志传输的进程状态。

```
SYS@SHHAI>select client_process,process,sequence#,status
  2  from v$managed_standby;

CLIENT_P PROCESS    SEQUENCE# STATUS
-------- --------- ---------- ------------
ARCH     ARCH             174   CLOSING
ARCH     ARCH             173   CLOSING
ARCH     ARCH               0   CONNECTED
ARCH     ARCH             171   CLOSING
```

```
N/A       RFS                0    IDLE
LGWR      RFS              175    IDLE
N/A       MRP0             175    WAIT_FOR_LOG
```

上述查询结果的倒数第二行的 CLIENT_PROCESS 列显示为 LGWR，说明主数据库对应的归档目的地配置为使用 LGWR 传输日志（Oracle 已不建议使用 ARCH 传输日志），日志序列号为 175，可以从主数据库中确定 175 是否为当前日志序列号。

```
SYS@BJING>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      174
Next log sequence to archive    175
Current log sequence                  175
```

如果备用数据库没有接收到主数据库的当前联机日志数据，就不能执行角色切换。

10.5.2 检查备用端的日志应用

确定了主数据库的日志已经同步传输到备用端，接下来可以进一步查看备用端日志应用的情况。

物理备用数据库的日志应用 Redo Apply 的状况可以通过查询 V$MANAGED_STANDBY 对应 MRP 进程的状态信息和日志序列号信息获得，如果其状态为 WAIT_FOR_GAP，或对应的日志序列号低于 RFS 对应的日志序列号，则都说明应用进程没有跟上日志传输，应该暂缓执行角色切换。

逻辑备用数据库的日志应用 SQL Apply 的状况不能通过视图 V$MANAGED_STANDBY 获得（但可以通过进程 RFS 及其信息查看日志传输的信息）。要验证逻辑备用端的 SQL Apply 进展可以查看 V$LOGSTDBY_PROCESS 和 V$LOGSTDBY_PROGRESS 视图，前者反映 SQL Apply 的一组进程（READER、PREPARER、BUILDER、COORDINATOR、ANALYZER、APPLIER）的状态信息，后者通过时间 Time 和 SCN 号反映 SQL Apply 的应用进展。

```
SYS@JINAN>select type,status,high_scn from v$logstdby_process;

TYPE                  STATUS                                    HIGH_
SCN
--------------------  --------------------------------
-- ----------
```

```
COORDINATOR              ORA-16116: no work available
978907
ANALYZER                 ORA-16116: no work available
978730
APPLIER                  ORA-16116: no work available
978109
APPLIER                  ORA-16116: no work available
978726
APPLIER                  ORA-16116: no work available
978730
APPLIER                  ORA-16116: no work available
976763
APPLIER                  ORA-16116: no work available
978105
READER                   ORA-16240: Waiting for log file
978907
                         (thread# 1, sequence# 176)
PREPARER                 ORA-16116: no work available
978772

SYS@JINAN>select latest_scn,mining_scn,applied_scn
from v$logstdby_progress;

LATEST_SCN MINING_SCN APPLIED_SCN
---------- ---------- -----------
    981281     978907      978906
```

如果上面的 3 个 SCN 非常接近，且 LATEST_SCN > MINING_SCN > APPLIED_SCN，并且在连续的多次查询中处于不断变化中，这表明逻辑备用的日志应用环节运转正常，并不存在日志间隔（Log Gap）。

10.5.3　检查会话和后台作业

在执行角色切换或故障转移前，应该断开不必要的用户连接，应该取消任何运行在数据库或备用数据库的后台作业（如用户的调度作业、RMAN 备份等），因为它们可能会影响角色切换操作，甚至会导致角色切换操作失败。通俗地说，实施角色切换的数据库应该轻装上阵，避免不必要的干扰。

查询会话信息可通过视图 V$SESSION，查询运行的作业可通过视图 DBA_JOBS_RUNNING，查询 RMAN 的后台备份可通过视图 V$RMAN_STATUS，如果存在不必要的用户会话（如普通用户会话、非 SYS 用户会话、各类应用会话等），以及

正在运行的不必要的后台作业，应该等待它们完成或将它们终止和取消。

强制终止会话可以使用下面的指令：

```
SQL> alter system kill session 'sid,serial#';
```

强制取消正在运行的作业可以使用下面的过程：

```
SQL> exec dbms_scheduler.stop_job(...);

SYS@BJING>select  username,sid,serial#,type,program  from
v$session
  2  where type<>'BACKGROUND';

USERNAME           SID    SERIAL# TYPE       PROGRAM
------------------------------------------------------------
SYSTEM             87     200    USER       java.exe
SYS                105    13           USER       sqlplus.exe

SYS@BJING>select switchover_status from v$database;

SWITCHOVER_STATUS
--------------------
SESSIONS ACTIVE
```

当备用数据库的日志传输没有被同步时：

```
SYS@BJING>select switchover_status from v$database;

SWITCHOVER_STATUS
--------------------
NOT ALLOWED
```

当备用数据库的日志传输同步后：

```
SYS@BJING>select switchover_status from v$database;

SWITCHOVER_STATUS
--------------------
TO STANDBY
```

10.6　物理备用与主数据库的角色切换

10.6.1　网络配置与角色参数

首先，要确保主数据库端和物理备用数据库端都能够通过网络服务名（Net Service Name）访问对方。

其次，要调整与主备用角色有关的参数。下面这组参数与 Data Guard 的配置有关，需要参照数据库、实例及其角色正确设置。

※ DB_UNIQUE_NAME。

※ LOG_ARCHIVE_CONFIG。

※ FAL_CLIENT。

※ FAL_SERVER。

※ STANDBY_FILE_MANAGEMENT。

※ DB_FILE_NAME_CONVERT。

※ LOG_FILE_NAME_CONVERT。

※ LOG_ARCHIVE_DEST_n。

另外，强烈建议在主数据库中也创建备用重做日志，以满足角色切换的需要。

10.6.2　主数据库中的工作

在主数据库中执行下面的指令将数据库切换至备用数据库角色。注意，此时物理备用数据库应该处于托管恢复状态，以便及时应用由主数据库的切换操作产生的重做数据。

为角色切换配置好主备用数据库的初始化参数后，切换工作从主数据库开始。检查主数据库的切换状态（V$DATABASE.SWITCHOVER_STATUS），确保其切换状态的取值为 TO STANDBY，接下来执行下面的指令：

```
SYS@BJING>alter database commit to switchover to physical
standby
  2  with session shutdown;


Database altered.
```

上述切换操作导致数据库实例处于 nomount 状态。详细的切换操作对应的内部机制如下告警文件所述（from alert log）：

```
Wed Feb 20 10:18:16 2013
```

```
alter database commit to switchover to physical standby
with session shutdown
Wed Feb 20 10:18:21 2013
Thread 1 advanced to log sequence 179
Wed Feb 20 10:18:21 2013
  Current log# 1 seq# 179 mem# 0:
+DGDATA/BJING/onlinelog/group_1.258.807869905
  Current log# 1 seq# 179 mem# 1:
+DG_FRA/BJING/onlinelog/group_1.365.807869907
Wed Feb 20 10:18:21 2013
ARCH: Standby redo logfile selected for thread 1 sequence 178
for destination LOG_ARCHIVE_DEST_2
Wed Feb 20 10:18:22 2013
Shutting down archive processes
Current log# 1 seq# 179 mem# 0:
+DGDATA/BJING/onlinelog/group_1.258.807869905
  Current log# 1 seq# 179 mem# 1:
+DG_FRA/BJING/onlinelog/group_1.365.807869907
Wed Feb 20 10:18:23 2013
SMON: disabling tx recovery
Wed Feb 20 10:18:24 2013
Stopping background process QMNC
Wed Feb 20 10:18:27 2013
ARCH shutting down
ARC2: Archival stopped
Wed Feb 20 10:18:27 2013
Thread 1 advanced to log sequence 180
  Current log# 2 seq# 180 mem# 0:
+DGDATA/BJING/onlinelog/group_2.257.807869909
  Current log# 2 seq# 180 mem# 1:
+DG_FRA/BJING/onlinelog/group_2.430.807869911
Wed Feb 20 10:18:27 2013
SMON: disabling cache recovery
Wed Feb 20 10:18:27 2013
LGWR: Waiting for ORLs to be archived...
Wed Feb 20 10:18:27 2013
ARC0: Standby redo logfile selected for thread 1 sequence 179
for destination LOG_ARCHIVE_DEST_2
```

```
Wed Feb 20 10:18:30 2013
LGWR: ORLs successfully archived
Shutting down archive processes
Archiving is disabled
Wed Feb 20 10:18:35 2013
ARCH shutting down
ARC1: Archival stopped
Wed Feb 20 10:18:40 2013
ARCH shutting down
ARC0: Archival stopped
Wed Feb 20 10:18:41 2013
Thread 1 closed at log sequence 180
Successful close of redo thread 1
Wed Feb 20 10:18:41 2013
ARCH: Noswitch archival of thread 1, sequence 180
ARCH: End-Of-Redo Branch archival of thread 1 sequence 180
ARCH: Archiving is disabled due to current logfile archival
Clearing standby activation ID 3950666251 (0xeb7a620b)
The primary database controlfile was created using the
'MAXLOGFILES 32' clause.
There is space for up to 30 standby redo logfiles
Use the following SQL commands on the standby database to
create
standby redo logfiles that match the primary database:
ALTER DATABASE ADD STANDBY LOGFILE 'srl1.f' SIZE 52428800;
ALTER DATABASE ADD STANDBY LOGFILE 'srl2.f' SIZE 52428800;
ALTER DATABASE ADD STANDBY LOGFILE 'srl3.f' SIZE 52428800;
Archivelog for thread 1 sequence 180 required for standby
recovery
MRP0 started with pid=16, OS id=4440
Managed Standby Recovery not using Real Time Apply
Online logfile pre-clearing operation disabled by switchover
Media Recovery Log +DG_FRA/BJING/archivelog/
2013_02_20/thread_1_seq_180.390.807877121
Identified End-Of-Redo for thread 1 sequence 180
Wed Feb 20 10:18:47 2013
Media Recovery End-Of-Redo indicator encountered
Wed Feb 20 10:18:47 2013
```

```
Media Recovery Applied until change 911390
Resetting standby activation ID 3950666251 (0xeb7a620b)
Wed Feb 20 10:18:48 2013
SUCCESS: diskgroup DGDATA was dismounted
SUCCESS: diskgroup DG_FRA was dismounted
Wed Feb 20 10:18:48 2013
Completed: alter database commit to switchover to physical
standby
with session shutdown
```

该指令执行完毕，数据库处于 nomount 状态，重新启动至 mount 状态，查看其数据库角色已经转换为物理备用数据库。

```
SYS@BJING>startup mount force;
ORACLE instance started.

Total System Global Area  125829120 bytes
Fixed Size                  1247660 bytes
Variable Size              71304788 bytes
Database Buffers           50331648 bytes
Redo Buffers                2945024 bytes
Database mounted.
SYS@BJING>select database_role,switchover_status from
v$database;

DATABASE_ROLE            SWITCHOVER_STATUS
---------------- --------------------
PHYSICAL STANDBY         TO PRIMARY
```

上述在主数据库中执行的指令，也会在对应的物理备用数据库中产生一系列的处理。与此同时，在物理备用数据库端的处理过程如下（from alert log）：

```
Wed Feb 20 10:18:17 2013
RFS[1]: Possible network disconnect with primary database
Redo Shipping Client Connected as PUBLIC
-- Connected User is Valid
RFS[3]: Assigned to RFS process 5260
RFS[3]: Identified database type as 'physical standby'
RFS[3]: Successfully opened standby log 3:
'+DG_STD/SHHAI/onlinelog/group_3.275.807274137'
Wed Feb 20 10:18:22 2013
```

```
Media Recovery Log +DG_FRA/SHHAI/archivelog/
2013_02_20/thread_1_seq_178.386.807877101
Media Recovery Waiting for thread 1 sequence 179
Wed Feb 20 10:18:27 2013
Redo Shipping Client Connected as PUBLIC
-- Connected User is Valid
RFS[4]: Assigned to RFS process 4596
RFS[4]: Identified database type as 'physical standby'
RFS[4]: Successfully opened standby log 3:
'+DG_STD/SHHAI/onlinelog/group_3.275.807274137'
Wed Feb 20 10:18:27 2013
Media Recovery Log +DG_FRA/SHHAI/archivelog/
2013_02_20/thread_1_seq_179.405.807877107
Media Recovery Waiting for thread 1 sequence 180
Wed Feb 20 10:18:41 2013
Redo Shipping Client Connected as PUBLIC
-- Connected User is Valid
RFS[5]: Assigned to RFS process 3792
RFS[5]: Identified database type as 'physical standby'
RFS[5]: Archived Log: '+DG_FRA/SHHAI/archivelog/
2013_02_20/thread_1_seq_180.433.807877121'
Wed Feb 20 10:18:43 2013
Media Recovery Log +DG_FRA/SHHAI/archivelog/
2013_02_20/thread_1_seq_180.433.807877121
Identified End-Of-Redo for thread 1 sequence 180
Wed Feb 20 10:18:43 2013
Media Recovery End-Of-Redo indicator encountered
Wed Feb 20 10:18:43 2013
Media Recovery Applied until change 911390
Resetting standby activation ID 3950666251 (0xeb7a620b)

#######################################################
```

此时，物理备用数据库端的 Redo Apply 已经停止，但数据库角色仍然是物理备用数据库，即此时角色转换中的两个数据库都是物理备用数据库。

```
SYS@SHHAI>select client_process,process,sequence#,status
from v$managed_standby;

CLIENT_P PROCESS    SEQUENCE# STATUS
```

```
-------- --------- ---------- ------------
ARCH     ARCH             178 CLOSING
ARCH     ARCH             179 CLOSING
```

10.6.3 物理备用数据库中的工作

接下来需要在物理备用数据库中执行角色切换，将其切换为主数据库。首先，同样需要检查数据库的切换状态（V$DATABASE.SWITCHOVER_STATUS），确保其切换状态的取值为 TO PRIMARY，然后，执行角色切换指令：

```
SYS@SHHAI>select database_role,switchover_status from
v$database;

DATABASE_ROLE            SWITCHOVER_STATUS
---------------- --------------------
PHYSICAL STANDBY   TO PRIMARY

SYS@SHHAI>alter database commit to switchover to primary
  2  with session shutdown;

Database altered.

SYS@SHHAI>select database_role,switchover_status from
v$database;

DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
PRIMARY          NOT ALLOWED

SYS@SHHAI> startup force;

SYS@SHHAI>select database_role,switchover_status from
v$database;

DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
PRIMARY          RESOLVABLE GAP

SYS@SHHAI>select database_role,switchover_status from
v$database;
```

```
DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
PRIMARY          TO STANDBY
```

上述将物理备用数据库切换为主数据库的角色切换过程的内部处理如下（from alert log）：

```
Wed Feb 20 10:34:38 2013
alter database commit to switchover to primary
with session shutdown
Wed Feb 20 10:34:38 2013
If media recovery active, switchover will wait 900 seconds
SwitchOver after complete recovery through change 911390
Online log +DG_STD/SHHAI/onlinelog/group_1.261.807876927:
Thread 1 Group 1 was previously cleared
Online log +DG_FRA/SHHAI/onlinelog/group_1.330.807876931:
Thread 1 Group 1 was previously cleared
Online log +DG_STD/SHHAI/onlinelog/group_2.260.807876933:
Thread 1 Group 2 was previously cleared
Online log +DG_FRA/SHHAI/onlinelog/group_2.317.807876935:
Thread 1 Group 2 was previously cleared
Standby became primary SCN: 911388
Converting standby mount to primary mount.
Completed: alter database commit to switchover to primary
with session shutdown
Wed Feb 20 10:34:39 2013
ARC0: STARTING ARCH PROCESSES
ARC2: Archival started
Wed Feb 20 10:34:39 2013
ARC0: STARTING ARCH PROCESSES COMPLETE
ARC0: Becoming the 'no SRL' ARCH
ARC2 started with pid=19, OS id=4196
```

至此，角色切换完毕，原来的主数据库（实例 BJING）切换为物理备用数据库，原来的物理备用数据库（实例 SHHAI）切换为主数据库。接下来只需在现有的物理备用数据库上启动托管恢复（Redo Apply）服务即可。

```
SYS@BJING>alter database recover managed standby database
disconnect;
```

```
Database altered.
```

对应的内部处理的主要过程如下（from alert log）：

```
Wed Feb 20 10:40:10 2013
alter database recover managed standby database disconnect
MRP0 started with pid=18, OS id=2112
Managed Standby Recovery not using Real Time Apply
Clearing online redo logfile 1
+DGDATA/BJING/onlinelog/group_1.258.807869905
Clearing online log 1 of thread 1 sequence number 179
Deleted Oracle managed file
+DGDATA/BJING/onlinelog/group_1.258.807869905
Deleted Oracle managed file
+DG_FRA/BJING/onlinelog/group_1.365.807869907
Wed Feb 20 10:40:16 2013
Completed: alter database recover managed standby database
disconnect
Wed Feb 20 10:40:17 2013
db_recovery_file_dest_size of 2048 MB is 5.42% used. This is a
user-specified limit on the amount of space that will be used
by this
database for recovery-related files, and does not reflect the
amount of space available in the underlying filesystem or ASM
diskgroup.
Clearing online redo logfile 1 complete
Clearing online redo logfile 2
+DGDATA/BJING/onlinelog/group_2.257.807869909
Clearing online log 2 of thread 1 sequence number 180
Deleted Oracle managed file
+DGDATA/BJING/onlinelog/group_2.257.807869909
Deleted Oracle managed file
+DG_FRA/BJING/onlinelog/group_2.430.807869911
Clearing online redo logfile 2 complete
Media Recovery Waiting for thread 1 sequence 181
```

10.6.4 主数据库与物理备用 SWITCHOVER 小结

（1）确保主备用数据库处于正常工作状态。主数据库处于打开状态，创建备用重做日志组，为 SWITCHOVER 做准备。物理备用处于托管恢复状态。

（2）检查主数据库的 SWITCHOVER STATUS，确保为 TO STANDBY。主数据的角色切换状态应该如下：

```
SQL>select database_role,switchover_status from v$database;

DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
PRIMARY          TO STANDBY
```

（3）将主数据库切换为物理备用角色。

```
SQL>alter database commit to switchover to physical standby
with session shutdown;
```

该操作会导致主数据库内部切换日志，并在下一日志序列的头部放置 EOR 标志。此操作产生的日志传送到物理备用数据库端，被应用后会导致物理备用数据库停止托管恢复。

此时，SWITCHOVER 中的主备用端都处于物理备用的角色。

（4）将物理备用数据库切换为主数据库。主数据库执行完步骤（3）的切换后，物理备用端则允许执行 SWITCHOVER，状态如下：

```
SQL>select database_role,switchover_status from v$database;

DATABASE_ROLE             SWITCHOVER_STATUS
---------------- --------------------
PHYSICAL STANDBY          TO PRIMARY
```

此时，即可执行下面的切换，将物理备用切换为主数据库角色：

```
SQL> alter database commit to switchover to primary
with session shutdown;
```

（5）在新的物理备用数据库上启动托管恢复。SWITCHOVER 完成。

值得注意的是，在上述主数据库和物理备用数据库的角色切换过程中，如果 Data Guard 环境中存在其他的物理备用数据库、逻辑备用数据库，在执行角色切换前，应该将它们的 Redo Apply、SQL Apply 启动，以便它们能正常地处理在角色切换过程中产生的日志流。完成角色切换后，这些其他的物理备用数据库、逻辑备用数据库上 Redo Apply、SQL Apply 有可能会停止，如果是这样，只需正常启动它们即可。

10.7 逻辑备用与主数据库的角色切换

在角色切换中，如果选择逻辑备用数据库与主数据库进行 Switchover，由于逻辑备用数据库并不是主数据库的精确副本，所以，当进行角色对调时，需要更多的内部处理，如主数据库要切换为逻辑备用数据库，必须在接收的日志流里获得 Logminer 字典，否则，不能识别来自新的主数据库的日志流并执行有效的日志挖掘。这个过程就是 Switchover 前的准备（Prepare）阶段。

另外，需要特别注意，如果逻辑备用数据库与主数据库进行角色切换前存在物理备用数据库，则切换后物理备用数据库就不能再配置为 Data Guard 的一部分。其内在原因是：逻辑备用数据库本质上是一个独立的数据库，并不像物理备用数据库是主数据库的精确副本。逻辑备用数据库只是在维护的数据集上与主数据库保持一致。

从 Oracle 11g 开始，在进行逻辑备用数据库与主数据库的 Switchover 过程中，不再需要关闭任何数据库实例，这一特点在工程中还是特别有价值的。

逻辑备用数据库与主数据库之间的角色切换之前，两个数据库实例的参数调整与之前的物理备用数据库的切换相同，在此不再赘述。

10.7.1 准备切换阶段

首先，在主数据库中执行下面的角色切换准备工作，注意前后的 SWITCHOVER_STATUS 状态信息。

```
SYS@BJING>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
PRIMARY            TO STANDBY

SYS@BJING>alter database prepare to switchover to logical
standby;

Database altered.

SYS@BJING>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
```

```
PRIMARY            PREPARING SWITCHOVER
```

这里的角色切换状态 PREPARING SWITCHOVER 的含义是主数据库已经做好准备，准备接收来自逻辑备用数据库的 Logminer 字典，这是当前主数据库切换为逻辑备用数据库所必需的。

然后，切换到当前的逻辑备用数据库中，执行下面的处理，使得该数据库能够在重做日志流中发送 Logminer 字典信息给当前的主数据库（准备切换为逻辑备用数据库）。同样，注意前后的 SWITCHOVER_STATUS 状态信息。

```
SYS@JINAN>select database_role,switchover_status from
v$database;

DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
LOGICAL STANDBY  NOT ALLOWED

SYS@JINAN>alter database prepare to switchover to primary;

Database altered.

SYS@JINAN>select database_role,switchover_status from
v$database;

DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
LOGICAL STANDBY  PREPARING SWITCHOVER
```

这里需要注意的是，要保证这里的 alter database prepare to switchover to primary 指令执行成功，应确保当前逻辑备用数据库指向主数据库的归档路径的设置正确有效，如下面的设置应该将 VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) 部分去掉，否则，在当前的环境下（当前数据库的角色仍然是 STANDBY_ROLE），该归档路径无效，也就无法向主数据库端发送 Logminer 字典信息。选项 VALID_FOR 的默认值是 (ALL_LOGFILES,ALL_ROLES)。

```
LOG_ARCHIVE_DEST_2 =
'SERVICE=BJING LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=BJING'
```

在逻辑备用数据库端的上述 prepare to switchover to primary 的“准

备”工作执行过程中，会向主数据库端发送切换为逻辑备用数据库的 Logminer 字典信息。当主数据库端已经接收到 Logminer 字典信息时，主数据库端的 SWITCHOVER_STATUS 状态就会变为 TO LOGICAL STANDBY，此时说明主数据库切换为逻辑备用数据库的准备工作已经就绪。

```
SYS@BJING>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
PRIMARY            TO LOGICAL STANDBY

SYS@JINAN>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
LOGICAL STANDBY    TO PRIMARY
```

上述主数据库端和逻辑备用数据库端的“准备”工作是可以取消的。如果这个时候不想执行真正的角色切换，前面的工作仅作为一种测试，可以先在主数据库端、后在逻辑备用数据库端执行如下操作，完全取消目前已经在主数据库和逻辑备用数据库上执行的角色切换前的所有准备工作。

```
SYS@BJING>alter database prepare to switchover cancel;

Database altered.

SYS@JINAN>alter database prepare to switchover cancel;

Database altered.
```

10.7.2 执行角色切换

经过上面的准备工作后，下面就可以执行真正的角色切换了。

首先在主数据库执行角色切换，将主数据库切换为逻辑备用数据库。

```
SYS@BJING>alter database commit to switchover to logical
standby;

Database altered.
```

```
SYS@BJING>select database_role,switchover_status from
v$database;

DATABASE_ROLE    SWITCHOVER_STATUS
---------------- --------------------
LOGICAL STANDBY  NOT ALLOWED
```

此时，两个数据库的角色都是逻辑备用数据库。

接下来在原先的逻辑备用数据库上执行角色切换，将逻辑备用数据库切换为主数据库。

```
SYS@JINAN>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
LOGICAL STANDBY    TO PRIMARY
SYS@JINAN>alter database commit to switchover to primary;

Database altered.

SYS@JINAN>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
PRIMARY            LOG SWITCH GAP
```

至此，逻辑备用数据库和主数据库之间的角色切换完毕，原先的主数据库（实例 BJING）切换为逻辑备用数据库，原先的逻辑备用数据库（实例 JINAN）切换为主数据库。这里的主数据库 SWITCHOVER_STATUS 状态为 LOG SWITCH GAP 指示新的逻辑备用数据库和新的主数据库之间存在日志间隔。

处理这个日志间隔只需在新的逻辑备用数据库上启动 SQL Apply，在新的主数据库上执行一次日志切换即可。

```
SYS@BJING>alter database start logical standby apply
immediate;

Database altered.
```

```
SYS@JINAN>alter system switch logfile;

System altered.

SYS@JINAN>select database_role,switchover_status from
v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
PRIMARY            TO STANDBY
```

10.7.3 主数据库与逻辑备用数据库 SWITCHOVER 小结

（1）确保当前主备用数据库处于正常工作状态。主数据库处于打开状态，创建备用重做日志组，为 SWITCHOVER 做准备。逻辑备用处于 SQL Apply 状态。

（2）主备用数据库的“准备”工作。“准备”工作中，当前的逻辑备用数据库要向主数据库端逆向发送日志，因此，要确保相关参数有效。

主数据库的准备工作：

```
SQL>alter database prepare to switchover to logical standby;
SQL>select database_role,switchover_status from v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
PRIMARY            PREPARING SWITCHOVER
```

逻辑备用数据库的准备工作：

```
SQL>alter database prepare to switchover to primary;
SQL>select database_role,switchover_status from v$database;

DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
LOGICAL STANDBY    PREPARING SWITCHOVER
```

注意，当这里的逻辑备用数据库端的“准备”工作完成后，主数据库上的 SWITCHOVER 状态随之切换为：

```
SQL>select database_role,switchover_status from v$database;
```

```
DATABASE_ROLE      SWITCHOVER_STATUS
---------------- --------------------
PRIMARY            TO LOGICAL STANDBY
```

此时，“准备”工作完成，但可以取消。若要取消当前的“准备”工作，可在主数据库端执行如下语句：

```
SQL>alter database prepare to switchover cancel;
```

在逻辑备用数据库端执行如下语句：

```
SQL>alter database prepare to switchover cancel;
```

（3）将主数据库切换为逻辑备用数据库：

```
SQL>alter database commit to switchover to logical standby;
```

此时，SWITCHOVER 中的主备用端都处于逻辑备用的角色。

（4）将逻辑备用数据库切换为主数据库：

```
SQL>alter database commit to switchover to primary;
```

（5）在新的逻辑备用数据库上启动 SQL Apply 服务。SWITCHOVER 完成。

10.8　故障转移至备用数据库

主数据库由于某种原因出现故障甚至是完全瘫痪，这时备用数据库应该替换主数据库，这也正是构建 Data Guard 数据保障方案的主要目的之一。下面的问题是如何选择备用数据库（若存在多个备用数据库的话）、备用数据库在独立切换为主数据库之前应该做怎样的准备工作、当选择的备用数据库切换为主数据库后，原来的主数据库和剩余的其他备用数据库如何利用。这是本节需要探讨的问题。

本节的假设前提是主数据库不能启动到 open 状态，必须将生产系统的数据库转移至某个备用数据库。

10.8.1　备用数据库的选择和处理

如果 Data Guard 环境中有多个备用数据库，建议按照如下思路选择备用数据库。

（1）充分利用重做日志，选择接收并应用最多重做数据的备用数据库。至于如何判断，参见后面的介绍。

（2）备用数据库若存在物理备用和逻辑备用，应优先选择物理备用数据库，本质上物理备用数据库和主数据库是同一数据库，而逻辑备用数据库则是一个独立的数据库。

（3）如果主数据库运行在“最大保护”模式，多个备用数据库中至少有一个备用数据库与主数据库保持完全同步的状态，选择同步的那个数据库。

（4）如果主数据库的归档目标的传输方式存在同步（SYNC）和异步（ASYNC），则选择同步传输的备用数据库。

（5）如果主数据库的重做日志可以利用，也许其中有对于备用数据库有用的部分，在故障转移之前，应尽可能应用到备用数据库。

通过备用重做日志查看备用数据库接收主数据库重做数据的进度。下面的查询在物理备用和逻辑备用数据库中意义一致，给出的示例查询中，接收主数据库的重做数据的日志序列号为230，如果多个备用数据库序列号相同，则比较最后的SCN（这里是1138810），如果last_change#字段内容为空，则可以比较used字段，该列指示接收重做数据的字节数，本例是40960字节。

```
SQL> select group#,sequence#,used,last_change#,status
from v$standby_log;

    GROUP#  SEQUENCE#    USED    LAST_CHANGE# STATUS
---------- ---------- ---------- ------------ ----------
         3         230  40960      1138810     ACTIVE
         4           0    512            0     UNASSIGNED
         5           0    512            0     UNASSIGNED
```

另外，还可以通过视图V$DATAGUARD_STATS查看transport lag、apply lag和apply finish time的信息，也可以比较不同备用数据库接收和应用主数据库重做数据的进度。

```
SQL>select name,nvl(value,0) value,unit,time_computed
from v$dataguard_stats
 where name like '%lag' or name='apply finish time';

NAME                VALUE            UNIT      TIME_COMPUTED
-------------------------------------------------------------------
transport lag 0 day(2) to second(0) interval 02/21/2013 20:56:03
apply lag     0 day(2) to second(0) interval 02/21/2013 20:56:03
apply finish time 0 day(2) to second(3) interval 02/21/2013 20:56:03
```

10.8.2 转移至物理备用数据库

实际的故障转移操作相当于单边的角色切换，仅在备用数据库上即可完成。

在物理备用数据库上查询V$STANDBY_LOG，发现最大的日志序列号为230，

但从主数据库端检查发现存在大于 230 的日志序列号（如果还存在的话），需要手工注册到物理备用数据库。

（1）手工注册从主数据库上获得的可用日志，代码如下：

```
SYS@SHHAI>alter database register or replace physical logfile
  2  '+GFRA/BJING/ARCHIVELOG/O1_MF_1_230_8LD3D92G_.ARC';

Database altered.
SYS@SHHAI>alter database register physical logfile
  2  '+GFRA/BJING/ARCHIVELOG/O1_MF_1_231_8LD82YX9_.ARC';

Database altered.
...
```

（2）启动重做应用，代码如下：

```
SYS@SHHAI>alter database recover managed standby database
disconnect;

Database altered.

SYS@SHHAI>select client_process,process,sequence#,status
from v$managed_standby;

CLIENT_P PROCESS    SEQUENCE# STATUS
-------- --------- ---------- ------------
ARCH     ARCH               0 CONNECTED
ARCH     ARCH               0 CONNECTED
N/A      MRP0             234 WAIT_FOR_LOG
```

由此可见，Redo Apply 进程 MRP 已应用至日志序列号 234（此序列号即是前面注册的最后日志序列）。

（3）取消托管恢复，终止日志应用，代码如下：

```
SYS@SHHAI>alter database recover managed standby database
cancel;

Database altered.
```

```
SYS@SHHAI>alter database recover managed standby database
finish;

Database altered.
```

这里的 finish 关键字（或 finish force）是必需的。指示物理备用数据库应用所有可用的重做数据，并在最后的日志序列 234 中添加 EOR 标记（模拟角色切换），从此停止接收日志。关键字 force 的含义是如果物理备用数据库端还存在 RFS（用于接收远程重做数据的进程），则强制中断该进程，否则，finish 操作不能执行。在 11g 中 force 是 finish 操作的默认选项。

此操作对应的告警文件内容如下：

```
Thu Feb 21 21:50:54 2013
alter database recover managed standby database finish
 started logmerger process
Thu Feb 21 21:50:54 2013
Managed Standby Recovery not using Real Time Apply
Parallel Media Recovery started with 2 slaves
Media Recovery Waiting for thread 1 sequence 234
Begin: Standby Redo Logfile archival
End: Standby Redo Logfile archival
Terminal Recovery timestamp is '02/21/2013 21:50:55'
Terminal Recovery: applying standby redo logs.
Terminal Recovery: thread 1 seq# 234 redo required
Media Recovery Waiting for thread 1 sequence 234
Terminal Recovery: End-Of-Redo log allocation
MRP: Validating standby redo logfile 3
Media Recovery Log /DATABASE/SHHAI/ST3BJING.LOG
Terminal Recovery: log 3 reserved for thread 1 sequence 234
Recovery of Online Redo Log: Thread 1 Group 3 Seq 234 Reading
mem 0
  Mem# 0: /DATABASE/SHHAI/ST3BJING.LOG
Identified End-Of-Redo for thread 1 sequence 234
Incomplete Recovery applied until change 1139434 time
02/21/2013 21:26:04
Terminal Recovery: successful completion
Thu Feb 21 21:50:56 2013
Identified End-Of-Redo for thread 1 sequence 234
Resetting standby activation ID 3949950974 (0xeb6f77fe)
```

```
Completed: alter database recover managed standby database
finish
```

这里存在这样一种情况：如果由于存在日志间隔（Gap）导致 MRP 进程不能应用已接收的所有重做数据，就会导致 finish 操作失败。这时可以直接使用下面的指令直接激活物理备用数据库而不应用任何重做数据。

```
alter database activate physical standby database;
```

一旦这里的 finish 操作完成，不论原先主数据库在什么保护模式下，此时数据库数据库在最高性能模式下。

```
SYS@SHHAI>select database_role,protection_mode,protection_
level
from v$database;

DATABASE_ROLE    PROTECTION_MODE      PROTECTION_LEVEL
---------------- -------------------- --------------------
PHYSICAL STANDBY MAXIMUM PERFORMANCE  MAXIMUM PERFORMANCE
```

（4）角色切换，将物理备用数据库转换为主数据库。下面的操作和正常的角色切换相同，执行当前数据库的角色切换。

```
SYS@SHHAI>alter database commit to switchover to primary
with session shutdown;

Database altered.

SYS@SHHAI>alter database open;

Database altered.

SYS@SHHAI>select database_role,protection_mode,protection_
level
from v$database;

DATABASE_ROLE      PROTECTION_MODE            PROTECTION_LEVEL
---------------- -------------------- --------------------
PRIMARY            MAXIMUM PERFORMANCE        UNPROTECTED
```

至此，基于物理备用数据库的故障转移完毕。注意，此时新的主数据库处于未保护状态。

最后，对因主数据库故障，将系统转移至物理备用数据库的过程进行小结。

（1）如果故障的主数据库能够 mount，可执行下面的指令向备用端传输日志：

```
SQL>alter database flush redo to target_db_unique_name;
```

（2）尽可能地应用已有的（或从主数据库复制注册的）日志：

```
SQL>alter database register physical logfile '...';
SQL>alter database recover managed standby database using
current logfile disconnect;
......
SQL>alter database recover managed standby database cancel;
```

（3）接下来根据下面指令的执行结果分别处理：

```
SQL>alter database recover managed standby database finish;
```

（a）如果没有任何错误，与 SWITCHOVER 一样，将物理备用数据库切换为主数据库：

```
SQL>select switchover_status from v$database; （应该为 TO
PRIMARY）
SQL>alter database commit to switchover to primary with
session shutdown;
```

（b）如果存在任何错误（由缺乏日志或日志间隔导致），则需要激活物理备用数据库：

```
SQL>alter database activate physical standby database;
SQL>alter database open;
```

10.8.3 转移至逻辑备用数据库

首先，确定逻辑备用数据库的应用进展。

逻辑备用数据库是通过 SQL Apply 应用实现数据更新的，有自己的联机日志和归档日志，这里不要和通过 RFS 接收的日志相混淆。下面的查询获得逻辑备用数据库接收和应用来自主数据库的重做数据的信息。

```
SYS@JINAN>select file_name,sequence#,blocks,applied
from DBA_LOGSTDBY_LOG;

FILE_NAME                          SEQUENCE#     BLOCKS   APPLIED
-------------------------------------------------------------------
...
.../O1_MF_1_221_8LD31T5V_.ARC    221           796      YES
```

```
.../O1_MF_1_222_8LD31T7B_.ARC   222          692      YES
.../O1_MF_1_223_8LD31TSD_.ARC   223          3        YES
.../O1_MF_1_224_8LD31XXS_.ARC   224          612      YES
.../O1_MF_1_225_8LD31Z0Z_.ARC   225          63       YES
.../O1_MF_1_226_8LD31ZVC_.ARC   226          4011     CURRENT
.../O1_MF_1_227_8LD31BPY_.ARC   227          181      CURRENT
.../O1_MF_1_228_8LD3190H_.ARC   228          41       CURRENT
```

从上面的查询结果可以知道，逻辑备用数据库收到的最新的主数据库的日志序列为 228，该序列归档文件包含 41 个重做数据块。接下来的查询可以进一步了解逻辑备用数据库的应用进展。

```
SYS@JINAN>select type,status,high_scn from v$logstdby_process;

TYPE              STATUS                              HIGH_SCN
----------------- ----------------------------------- ----------
COORDINATOR       ORA-16116: no work available        1138419
ANALYZER          ORA-16116: no work available        1138381
APPLIER           ORA-16116: no work available
APPLIER           ORA-16116: no work available
APPLIER           ORA-16116: no work available
APPLIER           ORA-16116: no work available
APPLIER           ORA-16116: no work available
READER            ORA-16240: Waiting for log file     1138419
                  (thread# 1, sequence# 229)
BUILDER           ORA-16116: no work available        1138416
PREPARER          ORA-16116: no work available        1138415
```

通过上面的查询我们了解到，当前逻辑备用数据库 SQL Apply 的最新进展到达的 SCN 是 1138419，SQL Apply 正在等待读取序列号为 229 的来自于主数据库的重做数据。我们的实验环境是同一主数据库配置有两个备用数据库（一个物理备用和一个逻辑备用）。与上面的物理备用数据库相比，物理备用数据库接收的最新日志序列为 230，到达的 SCN 为 1138810。两者相比较可以了解到，逻辑备用数据库落后于物理备用数据库。需要说明的是，这里的逻辑备用数据库落后于物理备用数据库，并不是必然出现的情况，而是实验环境有意为之。

特别注意，逻辑备用数据库是在打开状态下通过 SQL Apply 更新数据的，该数据库有自己的日志序列和 SCN，不要和主数据库的日志序列及 SCN 相混淆。

1．手工注册从主数据库中获得的可用日志

这里需要手工注册的是：所有对当前逻辑备用端还没有接收的重做日志（包括在主数据库中可用的归档日志和最新的联机日志）。

```
SYS@JINAN>alter database register or replace logical logfile
  2  '+GFRA/BJING/ARCHIVELOG/O1_MF_1_226_8LD2RD0H_.ARC';

Database altered.

SYS@JINAN>alter database register or replace logical logfile
  2  '+GFRA/BJING/ARCHIVELOG/O1_MF_1_227_8LD30ZD6_.ARC';

Database altered.

SYS@JINAN>alter database register or replace logical logfile
  2  '+GFRA/BJING/ARCHIVELOG/O1_MF_1_228_8LD3136C_.ARC';

Database altered.

SYS@JINAN>alter database register logical logfile
  2  '+GFRA/BJING/ARCHIVELOG/O1_MF_1_229_8LD38Q8S_.ARC';

Database altered.
...
```

2．启动逻辑备用数据库的 SQL Apply 应用

```
SYS@JINAN>alter database start logical standby apply
immediate;

Database altered.

SYS@JINAN>select file_name,sequence#,blocks,applied
  2  from DBA_LOGSTDBY_LOG;
...
.../O1_MF_1_227_8LD30ZD6_.ARC          227    181 YES
.../O1_MF_1_228_8LD3136C_.ARC          228    41 YES
.../O1_MF_1_229_8LD38Q8S_.ARC          229    315 YES
.../O1_MF_1_230_8LD3D92G_.ARC          230    118 YES
```

```
.../O1_MF_1_231_8LD82YX9_.ARC          231    472 YES
.../O1_MF_1_232_8LD8325C_.ARC          232    68 YES
.../O1_MF_1_233_8LD83H2X_.ARC          233    119 CURRENT
.../O1_MF_1_234_8LG2NP1G_.ARC          234    845 CURRENT

SYS@JINAN>select type,status,high_scn from v$logstdby_process;

TYPE               STATUS                              HIGH_SCN
------------------ ------------------------------ ----------
......
READER             ORA-16240: Waiting for log file    1140113
                   (thread# 1, sequence# 235)
...
```

3．激活逻辑备用数据库

等待所有已注册的重做文件全部应用后，将此逻辑备用数据库切换为主数据库。与物理备用数据库切换为主数据库的情况一样，首先，要应用所有未应用的已接收的重做数据，然后，再进行角色切换。逻辑备用数据库运行在 open 状态下，切换为主数据库的步骤相对简单，代码如下：

```
SYS@JINAN>alter database stop logical standby apply;

Database altered.

SYS@JINAN>alter database activate logical standby database
finish apply;

Database altered.

SYS@JINAN>select database_role,protection_mode,protection_
level
from v$database;

DATABASE_ROLE    PROTECTION_MODE       PROTECTION_LEVEL
---------------- -------------------- --------------------
PRIMARY          MAXIMUM PERFORMANCE   MAXIMUM PERFORMANCE
```

注意上面的语句 alter database activate logical standby database finish apply; 此处指定 finish apply 选项的含义是在切换为主

数据库之前尽可能应用已接收的所有重做数据。如果由于某种原因（如存在日志间隔），不能挖掘和应用所有已接收的日志，则可使用 alter database activate logical standby database; 语句直接激活逻辑备用数据库（切换为主数据库）。

下面是上面的角色切换指令的内部处理过程（示例）。

```
Fri Feb 22 10:27:10 2013
alter database activate logical standby database finish apply
LOGSTDBY: Logical ACTIVATE state
LOGSTDBY: terminating active RFS connections for failover
LOGSTDBY: Starting Logical standby for Terminal Apply
Fri Feb 22 10:27:10 2013
LSP0 started with pid=30, OS id=1372
LOGMINER: Parameters summary for session# = 4
LOGMINER: Number of processes = 3, Transaction Chunk Size =
201
LOGMINER: Memory Size = 30M, Checkpoint interval = 150M
LOGMINER: SpillScn 1138121, ResetLogScn 1
LOGMINER: summary for session# = 4
LOGMINER: StartScn: 0 (0x0000.00000000)
LOGMINER: EndScn: 0 (0x0000.00000000)
LOGMINER: HighConsumedScn: 1138129 (0x0000.00115dd1)
LOGMINER: session_flag 0x3
Fri Feb 22 10:27:12 2013
LOGMINER: session#=4, reader MS00 pid=31 OS id=3600 sid=106
started
Fri Feb 22 10:27:12 2013
LOGMINER: session#=4, builder MS01 pid=32 OS id=2556 sid=23
started
Fri Feb 22 10:27:12 2013
LOGMINER: session#=4, preparer MS02 pid=33 OS id=4080 sid=105
started
LOGMINER: Turning ON Log Auto Delete
Fri Feb 22 10:27:13 2013
LOGSTDBY Analyzer process AS00 started with server id=0 pid=34
OS id=2952
Fri Feb 22 10:27:13 2013
LOGSTDBY Apply process AS02 started with server id=2 pid=36 OS
id=3796
Fri Feb 22 10:27:13 2013
```

```
LOGSTDBY Apply process AS01 started with server id=1 pid=35 OS
id=1572
Fri Feb 22 10:27:13 2013
LOGSTDBY Apply process AS05 started with server id=5 pid=39 OS
id=128
Fri Feb 22 10:27:13 2013
LOGSTDBY Apply process AS04 started with server id=4 pid=38 OS
id=1932
LOGMINER: begin Terminal Apply mode
LOGMINER: Begin mining logfile for session 4 thread 1 sequence
226, +GFRA/JINAN/FOREIGN_ARCHIVELOG/BJING/2013_02_21/O1_
MF_1_226_8LD31ZVC_.ARC
LOGMINER: End   mining logfile for session 4 thread 1 sequence
226, +GFRA/JINAN/FOREIGN_ARCHIVELOG/BJING/2013_02_21/O1_
MF_1_226_8LD31ZVC_.ARC
LOGMINER: Begin mining logfile for session 4 thread 1 sequence
227, +GFRA/JINAN/FOREIGN_ARCHIVELOG/BJING/2013_02_21/O1_
MF_1_227_8LD31BPY_.ARC
LOGMINER: End   mining logfile for session 4 thread 1 sequence
227, +GFRA/JINAN/FOREIGN_ARCHIVELOG/BJING/2013_02_21/O1_
MF_1_227_8LD31BPY_.ARC
LOGMINER: Begin mining logfile for session 4 thread 1 sequence
228, +GFRA/JINAN/FOREIGN_ARCHIVELOG/BJING/2013_02_21/O1_
MF_1_228_8LD3190H_.ARC
LOGMINER: End   mining logfile for session 4 thread 1 sequence
228, +GFRA/JINAN/FOREIGN_ARCHIVELOG/BJING/2013_02_21/O1_
MF_1_228_8LD3190H_.ARC
LOGMINER: thread 1 seq 228 not written in SYNC AFFIRM mode
LOGMINER: Terminal Apply complete with thread 1 seq 228 scn
0x0000.00115ef3
LOGMINER: WARNING: failover completed with potential data loss
LOGMINER: thread merge scn 0x0000.00115ef2
Fri Feb 22 10:27:13 2013
LOGSTDBY Apply process AS03 started with server id=3 pid=37 OS
id=2708
LOGSTDBY: STANDBY_BECAME_PRIMARY_SCN established at
[0x0000.00115ef1]
```

```
LOGSTDBY Analyzer process AS00 server id=0 pid=34 OS id=2952
stopped
Fri Feb 22 10:27:14 2013
Logminer Bld: Lockdown Complete.  DB_TXN_SCN is   UnwindToSCN
(LockdownSCN) is 1043438
LOGSTDBY: Starting SCN of new stream from recent lockdown
[0x0000.000febee]
LOGSTDBY: enabling scheduler job queue processes.
JOBQ: re-enabling CJQ0
Completed: alter database activate logical standby database
finish apply
LOGSTDBY Apply process AS04 server id=4 pid=38 OS id=1932
stopped
Fri Feb 22 10:27:14 2013
LSP1 started with pid=40, OS id=2632
LOGSTDBY: LogMiner Dictionary Build Process Created
LOGSTDBY Event: Starting Unwind LogMiner Dictionary Build
Fri Feb 22 10:27:16 2013
Logminer Bld: Build started
LOGSTDBY Apply process AS03 server id=3 pid=37 OS id=2708
stopped
Fri Feb 22 10:27:16 2013
Logminer Bld: Lockdown Complete.  DB_TXN_SCN is 0 1043478
LockdownSCN is 1043478
Fri Feb 22 10:27:16 2013
Thread 1 advanced to log sequence 65 (LGWR switch)
  Current log# 1 seq# 65 mem# 0: /DATABASE/JINAN/RD1BJING.LOG
Fri Feb 22 10:27:16 2013
ARC3: STARTING ARCH PROCESSES
Fri Feb 22 10:27:16 2013
ARC4 started with pid=37, OS id=2532
LOGSTDBY Apply process AS02 server id=2 pid=36 OS id=3796
stopped
ARC4: Archival started
ARC3: STARTING ARCH PROCESSES COMPLETE
LOGSTDBY Apply process AS01 server id=1 pid=35 OS id=1572
stopped
```

```
LOGSTDBY Apply process AS05 server id=5 pid=39 OS id=128
stopped
LOGMINER: session#=4, reader MS00 pid=31 OS id=3600 sid=106
stopped
Archived Log entry 83 added for thread 1 sequence 64 ID
0xeb704c14 dest 1:
LOGMINER: session#=4, builder MS01 pid=32 OS id=2556 sid=23
stopped
LOGMINER: session#=4, preparer MS02 pid=33 OS id=4080 sid=105
stopped
LOGSTDBY status: ORA-16128: User initiated stop apply
successfully completed
Fri Feb 22 10:27:31 2013
Logminer Bld: Done
LOGSTDBY: (LSP1/Rebuild) Archiving standby redo logfiles.
LOGSTDBY: (LSP1/Rebuild) Using surrogate archiving mode
LOGSTDBY: (LSP1/Rebuild) Complete.  No standby redo logfiles
needed archiving.
LOGSTDBY Event: LogMiner Dictionary Build Process Completed.
```

在某些情况下，如果不希望在逻辑备用数据库切换为主数据库前处理与应用已接收的重做数据（如使用数据库闪回技术将逻辑备用数据库闪回到过去某个特定时刻），则可直接使用 `alter database activate logical standby database;` 指令执行由逻辑备用到主数据库的角色切换。

令人欣喜的是，在执行上述数据库角色切换后，数据库依旧处于 open 状态，即逻辑备用数据库在切换为主数据库的过程中数据库实例并不需要关闭，这是在实践中非常有价值的一个特性。

10.8.4　原有主数据库的再利用

物理备用数据库和主数据库本质上是同一数据库。由于实施故障转移，原来的物理备用数据库已经变成当前的主数据库。原来的主数据库虽然遭到损坏，但如果利用数据库闪回技术可以将其闪回到过去的某个时间点，则可以将其切换为新的主数据库的物理备用数据库，这样的处理比根据新的主数据库重新建立物理备用数据库更方便快捷。

实验条件：当前数据库实例 SHHAI 对应的是主数据库，数据库实例 BJING 对应的是原先的主数据库（目前只能启动至 mount 状态）。

（1）确定当前主数据库的 STANDBY_BECAME_PRIMARY_SCN。STANDBY_BECAME_PRIMARY_SCN 代表的是备用数据库切换为当前主数据库时对应的 SCN，此前是物理备用数据库，此后是主数据库。

```
SYS@SHHAI>select database_role,standby_became_primary_scn
from v$database;

DATABASE_ROLE    STANDBY_BECAME_PRIMARY_SCN
---------------- --------------------------
PRIMARY                             1139432
```

（2）尝试将原来的主数据库通过数据库闪回 FLASHBACK 到上一步查询到的 SCN。

```
SYS@BJING>startup mount force
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size             100666232 bytes
Database Buffers           58720256 bytes
Redo Buffers                6627328 bytes
Database mounted.
SYS@BJING>
SYS@BJING>flashback database to scn 1139432;

Flashback complete.
```

对应的内部闪回过程如下：

```
Fri Feb 22 19:40:31 2013
flashback database to scn 1139432
Flashback Restore Start
Flashback Restore Complete
Flashback Media Recovery Start
 started logmerger process
Parallel Media Recovery started with 2 slaves
Flashback Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_229_8LD38Q8S_.ARC
Flashback Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_230_8LD3D92G_.ARC
```

```
Flashback Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_231_8LD82YX9_.ARC
Flashback Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_232_8LD8325C_.ARC
Flashback Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_233_8LD83H2X_.ARC
Fri Feb 22 19:40:35 2013
Incomplete Recovery applied until change 1139433 time
02/21/2013 21:26:04
Flashback Media Recovery Complete
Completed: flashback database to scn 1139432
```

（3）角色切换，将闪回后的数据库（原先的主数据库）切换为物理备用数据库，代码如下：

```
SYS@BJING>alter database convert to physical standby;

Database altered.

SYS@BJING>select database_role from v$database;
select database_role from v$database
                          *
ERROR at line 1:
ORA-01507: database not mounted

SYS@BJING>shutdown
ORA-01507: database not mounted

ORACLE instance shut down.
SYS@BJING>startup mount
ORACLE instance started.

Total System Global Area  167387136 bytes
Fixed Size                  1373320 bytes
Variable Size             100666232 bytes
Database Buffers           58720256 bytes
Redo Buffers                6627328 bytes
Database mounted.
SYS@BJING>select database_role from v$database;
```

```
DATABASE_ROLE
----------------
PHYSICAL STANDBY
```

（4）启动托管恢复 Redo Apply。新的物理备用数据库一旦启动了托管恢复，主数据库（对应数据库实例 SHHAI）的 ARCH 的 ping 备用数据库操作（或物理备用数据库中的 FAL 操作）会探测到日志间隔，从而进行日志传输，进而执行重做应用。

```
SYS@BJING>alter database recover managed standby database
  2  using current logfile disconnect;

Database altered.

SYS@BJING>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     11
Next log sequence to archive   0
Current log sequence            11
SYS@BJING>select client_process,process,sequence#,status
from v$managed_standby;

CLIENT_P PROCESS    SEQUENCE# STATUS
-------- --------- ---------- ------------
ARCH     ARCH               9 CLOSING
ARCH     ARCH               0 CONNECTED
ARCH     ARCH              10 CLOSING
ARCH     ARCH               0 CONNECTED
N/A      RFS                0 IDLE
LGWR     RFS               11 IDLE

6 rows selected.
```

值得注意的是，经过托管恢复 Redo Apply，日志序列号已到达 11（经检查已与主数据库保持一致）。在 11g 中，物理备用数据库切换为主数据库时，日志序列号也会被重置。但请注意，前面的启动托管恢复操作成功，但在最后的进程查询中，虽然存在 RFS 进程，但并没有发现托管恢复进程 MRP。进一步查看告警日志，发现原来托管恢复进程在日志流中遭遇 EOR 标记（在原备用数据库切换为主数据库的过程中），导致 MRP 进程终止，参见下面的详细信息。

```
Fri Feb 22 20:00:03 2013
RFS[7]: Assigned to RFS process 2156
RFS[7]: Identified database type as 'physical standby': Client
is ARCH pid 3568
RFS[7]: Selected log 3 for thread 1 sequence 9 dbid -345029378
branch 808006678
Fri Feb 22 20:00:06 2013
RFS[8]: Assigned to RFS process 2172
RFS[8]: Identified database type as 'physical standby': Client
is LGWR ASYNC pid 1996
Primary database is in MAXIMUM PERFORMANCE mode
Re-archiving standby log 3 thread 1 sequence 9
RFS[8]: Selected log 5 for thread 1 sequence 10 dbid
-345029378 branch 808006678
Fri Feb 22 20:00:08 2013
Archived Log entry 462 added for thread 1 sequence 9 ID
0xeb7bed05 dest 1:
Fri Feb 22 20:00:08 2013
Archived Log entry 463 added for thread 1 sequence 10 ID
0xeb7bed05 dest 1:
RFS[8]: Selected log 3 for thread 1 sequence 11 dbid
-345029378 branch 808006678
Fri Feb 22 20:01:03 2013
RFS[9]: Assigned to RFS process 1892
RFS[9]: Identified database type as 'physical standby':
Client is ARCH pid 3568
Fri Feb 22 20:01:09 2013
alter database recover managed standby database
using current logfile disconnect
Fri Feb 22 20:01:09 2013
MRP0 started with pid=17, OS id=2820
 started logmerger process
Fri Feb 22 20:01:14 2013
Managed Standby Recovery starting Real Time Apply
Parallel Media Recovery started with 2 slaves
Media Recovery start incarnation depth :
1, target inc# : 2, irscn : 1139434
Waiting for all non-current ORLs to be archived...
```

```
All non-current ORLs have been archived.
Clearing online redo logfile 1 /DATABASE/BJING/RD1BJING.LOG
Clearing online log 1 of thread 1 sequence number 11
Completed: alter database recover managed standby database
using current logfile disconnect
Clearing online redo logfile 1 complete
Clearing online redo logfile 2 /DATABASE/BJING/RD2BJING.LOG
Clearing online log 2 of thread 1 sequence number 10
Clearing online redo logfile 2 complete
Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_234_8LGPTBC7_.ARC
Identified End-Of-Redo for thread 1 sequence 234
Resetting standby activation ID 3949950974 (0xeb6f77fe)
Media Recovery End-Of-Redo indicator encountered
Media Recovery Applied until change 1139434
Media Recovery archivelogs detected beyond End-Of-REDO
```

进一步在备用数据库中执行如下查询可以了解到，托管恢复 MRP 进程停止在何处。

```
SYS@BJING>select  sequence#,applied,first_change#,next_
change#,fal
from v$archived_log;

SEQUENCE# APPLIED     FIRST_CHANGE#  NEXT_CHANGE#       FAL
--------- ----------- -------------- ------------------ ---
243       NO          1142697        1142704            NO
234       YES         1139433        1139436            YES
1         NO          1139435        1139439            YES
2         NO          1139439        1139467            YES
3         NO          1139467        1150067            YES
4         NO          1150067        1152495            YES
5         NO          1152495        1152570            YES
6         NO          1152570        1156276            YES
7         NO          1156276        1156389            YES
8         NO          1156389        1158718            NO
9         NO          1158718        1158814            NO
10        NO          1158814        1158930            NO
```

从上面的告警文件内容，以及对 V$ARCHIVED_LOG 的查询结果可以清楚地看

出，MRP 进程在序列号为 234 的日志文件中遭遇 EOR，Redo Apply 停止在 SCN 号为 1139434 的位置（序列号为 234 的日志文件的开头）。另外，还可以看出当前物理备用数据库接收到的重做归档文件中哪些是通过 FAL 机制获得的。

注意到上面的告警日志内容的最后一行：

```
Media Recovery archivelogs detected beyond End-Of-REDO
```

这句记录的含义是 Oracle 已经检测到 EOR 标记之后存在可以应用的重做日志。

要使得 MRP 进程越过 EOR 标记重新执行 Redo Apply，只需重新启动当前物理备用数据库的托管恢复即可。

```
SYS@BJING>alter database recover managed standby database
  2  using current logfile disconnect;

Database altered.
```

此次启动托管恢复重新执行重做应用的过程如下：

```
Fri Feb 22 21:04:45 2013
alter database recover managed standby database
using current logfile disconnect
Fri Feb 22 21:04:45 2013
MRP0 started with pid=17, OS id=3716
 started logmerger process
Fri Feb 22 21:04:50 2013
Managed Standby Recovery starting Real Time Apply
Parallel Media Recovery started with 2 slaves
Waiting for all non-current ORLs to be archived...
All non-current ORLs have been archived.
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_1_8LGPTGC7_.
ARC
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_2_8LGPTGC7_.
ARC
Completed: alter database recover managed standby database
using current logfile disconnect
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_3_8LGPTGCQ_.
ARC
Fri Feb 22 21:05:05 2013
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_4_8LGPTKG5_.
ARC
```

```
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_5_8LGPTKM1_.
ARC
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_6_8LGPTL1Y_.
ARC
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_7_8LGPTLDP_.
ARC
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_8_8LGPTD2X_.
ARC
Media Recovery Log +GFRA/BJING/ARCHIVELOG/O1_MF_1_9_8LGQG7CQ_.
ARC
Media Recovery Log
+GFRA/BJING/ARCHIVELOG/O1_MF_1_10_8LGQG8PG_.ARC
Media Recovery Waiting for thread 1 sequence 11 (in transit)
```

进一步查询也可验证上面的重做应用过程。

```
SYS@BJING>select client_process,process,sequence#,status
from v$managed_standby;

CLIENT_P PROCESS          SEQUENCE#     STATUS
-------- --------- ---------- ------------
ARCH     ARCH             9             CLOSING
ARCH     ARCH             0             CONNECTED
ARCH     ARCH             10            CLOSING
ARCH     ARCH             0             CONNECTED
N/A      RFS              0             IDLE
LGWR     RFS              11            IDLE
N/A      MRP0             11            APPLYING_LOG
```

至此，新的物理备用数据库（实例 BJING）已经完全跟上了新的主数据库（实例 SHHAI），新的主数据库已经受到物理备用数据库的保护。如果要使主数据库再次回到原来的主机上（即实例 BJING 为主数据库运行、实例 SHHAI 为物理备用数据库运行），做一次正常的角色切换即可完成任务。

在故障转移过程中，如果选择转移至逻辑备用数据库（现为主数据库），原先的主数据库要想配合其成为新的逻辑备用数据库，需要经过更多的处理步骤，其原因是逻辑备用数据库本质上是主数据库之外的独立数据库，并不像物理备用数据库是主数据库的精确副本。在作者看来，这些处理步骤的繁杂程度不亚于重建一个新的逻辑备用数据库，且这种情况下原先 Data Guard 环境中的其他物理备用数据库也不能利用，需要直接重建。如遇此类情况，建议直接利用备份重新建立一个新的物理备用数据库和逻辑备用数据库。

第11章

Data Guard的主要参数、视图与管理指令

到目前为止，本书已经比较全面地介绍了Data Guard的核心内容。本章将总结配置与管理Data Guard环境所涉及的初始化参数、数据字典视图和相关联的一系列管理指令，既可作为对Data Guard要点内容的总结，也可作为日常管理Data Guard环境的参考。

11.1 初始化参数及其设置

11.1.1 Data Guard主要参数

Data Guard的配置涉及一系列初始化参数，理解这些参数的含义并正确设置这些参数是保证Data Guard正常运行的前提。

表11-1列出与Data Guard参数及其角色适应性。

表11-1 Data Guard参数及其角色适应性

参数名称	适用Data Guard角色	备注
DB_UNIQUE_NAME	主数据库、物理备用数据库、逻辑备用数据库	唯一性
LOG_ARCHIVE_CONFIG	主数据库、物理备用数据库、逻辑备用数据库	必需
LOG_ARCHIVE_DEST_n	主数据库	必需
CONTROL_FILES	主数据库、物理备用数据库、逻辑备用数据库	必需
REMOTE_LOGIN_PASSWORDFILE	主数据库、物理备用、逻辑备用	必需
FAL_SERVER	物理备用、逻辑备用	建议
FAL_CLIENT	主数据库	可选
STANDBY_FILE_MANAGEMENT	物理备用数据库	建议
STANDBY_ARCHIVE_DEST	物理备用数据库、逻辑备用数据库	可选
DB_FILE_NAME_CONVERT	物理备用数据库	可选

续表

参数名称	适用 Data Guard 角色	备注
LOG_FILE_NAME_CONVERT	物理备用数据库	可选
COMPATIBLE	主数据库、物理备用数据库、逻辑备用数据库	一致性
LOG_ARCHIVE_TRACE	主数据库、物理备用数据库、逻辑备用数据库	可选

下面重点介绍两个参数的设置，其他参数请参见本书相关章节。

11.1.2 参数 LOG_ARCHIVE_DEST_n

Oracle 数据库通过参数 LOG_ARCHIVE_DEST_n 控制归档日志文件的本地存储和远程传输。该参数通过一系列的属性控制日志归档的操作特性，其属性及意义如表 11-2 所示。

表 11-2　参数属性及意义

属性设置	属性含义
LOCATION=path_name	指定本地日志归档位置
SERVICE=service_name	指定远程日志归档目的地
MANDATORY	该归档位置或目的地必须归档成功
SYNC | ASYNC	远程归档的同步或异步
AFFIRM | NOAFFIRM	LNS 进程是否等待 RFS 进程 I/O 完成的反馈
NET_TIMEOUT=seconds	LNS 进程等待 RFS 进程反馈的时限
REOPEN =seconds	设置主数据库重新尝试连接备用端的时间间隔
VALID_FOR=(log_type,db_role)	根据日志类型和角色判断是否使用该归档参数
DB_UNIQUE_NAME	唯一数据库名，DG 中的所有数据库必须唯一
DELAY =minutes	指定备用端日志接收后应用的延迟时间
NOREGISTER	设置备用端是否注册接收的日志文件
TEMPLATE=template	覆盖远程参数 LOG_ARCHIVE_FORMAT
ALTERNATE=LOG_ARCHIVE_DEST_n	当前目的地归档失败后的替代目的地
MAX_FAILURE=count	设置尝试连接远程备用端的次数
MAX_CONNECTIONS=count	处理日志间隔时主备用端的并行连接数
COMPRESSION={ENABLE|DISABLE}	控制归档日志是否压缩传输

此处对属性 VALID_FOR=(redo_log_type,database_role) 做出说明。

该属性只有当日志类型 redo_log_type 和数据库角色 database_role 两者都有效时，对应的归档目标才有效。

（1）日志类型 redo_log_type 的取值如下。

※ ONLINE_LOGFILE：该归档目标归档联机重做日志时有效。

※ STANDBY_LOGFILE：该归档目标归档备用重做日志时有效。

※ ALL_LOGFILES：该归档目标归档联机重做日志或备用重做日志时都有效。

（2）数据库角色 database_role 的取值如下。

※ PRIMARY_ROLE：当角色为主数据库时有效。

※ STANDBY_ROLE：当角色为备用数据库时有效。

※ ALL_ROLES：当角色为主数据库和备用数据库时都有效。

11.1.3　参数 LOG_ARCHIVE_TRACE

当处理 Data Guard 的故障时，特别是与主备用端的日志归档与重做传输有关的问题时，我们需要了解其内部活动的细节。参数 LOG_ARCHIVE_TRACE 用来设置和跟踪数据库系统日志归档与重做传输活动的记录级别：

```
LOG_ARCHIVE_TRACE=trace_level
```

跟踪级别及其含义如表 11-3 所示。

表 11-3　跟踪级别及其含义

trace_level	跟踪内容
0	默认设置，此值禁用对日志归档与重做传输活动的跟踪
1	跟踪归档日志文件
2	根据归档日志文件目的地跟踪归档状态
4	跟踪归档操作阶段
8	跟踪归档日志目的地的活动
16	同样跟踪归档日志目的地的活动，但内容比级别 8 更详细
32	跟踪重做日志归档目标的参数修改
64	跟踪归档进程 ARCn 的活动及其状态
128	跟踪 FAL Server 进程的活动
256	跟踪逻辑备用数据库端进程 RFS 的活动
512	跟踪进程 LGWR 的远程日志传输活动
1024	跟踪物理备用数据库端进程 RFS 的活动
2048	跟踪 RFS/ARCn 进程的 ping 心跳活动
4096	跟踪备用数据库端重做数据的实时应用活动

续表

trace_level	跟踪内容
8192	跟踪介质恢复或物理备用数据库端的 Redo Apply 活动
16384	跟踪归档操作 I/O buffers 的使用信息
32768	跟踪 LogMiner 字典的归档活动

设置该参数时，可以通过将跟踪级别的数值相加来获得数据库同时启动多个级别的跟踪，如设置 LOG_ARCHIVE_TRACE=12，则同时启动级别 4 和级别 8 的归档跟踪。

11.2 Data Guard 视图

数据字典视图反映 Oracle 数据库内部的活动和各种运行状况。与 Data Guard 有关的视图是数据字典的一个子集，DBA 通过对这些视图的访问，可以有效地了解和监控主备用数据库的运行状况，如表 11-4 和表 11-5 所示。

表 11-4　监控 Data Guard 环境的视图一（通用）

视图名称	主要内容
V$DATAGUARD_STATUS	数据库告警日志和服务器跟踪中关于 DG 的主要信息
V$DATAGUARD_STATS	关于 Data Guard 的各类统计信息
V$DATAGUARD_CONFIG	列出 Data Guard 环境中的 DB_UNIQUE_NAME
V$DATABASE	关于数据库的整体信息
V$MANAGED_STANDBY	关于日志传输的进程及其状态信息
V$STANDBY_LOG	备用重做日志信息
V$ARCHIVED_LOG	归档重做日志信息
V$ARCHIVE_DEST	归档目标的设置信息
V$ARCHIVE_DEST_STATUS	关于归档目标的运行及其各种状态信息
V$ARCHIVE_GAP	备用数据库的归档日志间隔
V$LOGFILE	联机日志文件和备用日志文件的信息

表 11-5　监控 Data Guard 环境的视图二（逻辑备用数据库专用）

视图名称	主要内容
DBA_LOGSTDBY_UNSUPPORTED	逻辑备用数据库不支持的表和列
DBA_LOGSTDBY_NOT_UNIQUE	缺乏主键或唯一性索引的表
DBA_LOGSTDBY_SKIP	被 SQL Apply 忽略的表、对象或 SQL 语句
DBA_LOGSTDBY_PARAMETERS	控制 SQL Apply 运行的参数设置

续表

视图名称	主要内容
DBA_LOGSTDBY_EVENTS	包含逻辑备用数据库的活动信息
DBA_LOGSTDBY_LOG	在逻辑备用数据库中注册的日志文件
V$LOGSTDBY_PROCESS	显示 SQL Apply 日志挖掘和应用进程的活动信息
V$LOGSTDBY_PROGRESS	显示 SQL Apply 的应用进展（SCN 和时间）
V$LOGSTDBY_STATS	关于逻辑备用数据库 SQL Apply 的统计信息

11.3　Data Guard 的管理指令

在配置和管理 Data Guard 的过程中，涉及一系列对主备用数据库操作，这些操作都可通过 Oracle 支持的扩展的 SQL 指令去实现，与常规的 SQL 语言 DDL、DML、DCL 相比，这些扩展的 SQL 指令语法相对复杂、结构灵活多变，DBA 要掌握这套指令体系具有相当大的难度。本节对此做一次相对完整的总结。

Data Guard 环境的配置和管理主要涉及如下三大系列的指令。

※ 管理数据库：ALTER DATABASE ...

※ 管理实例：ALTER SYSTEM ...

※ 管理会话：ALTER SESSION ...

11.3.1　管理数据库

管理数据库系列指令用于 Data Guard 环境下的 ALTER DATABASE 指令，主要用于主备用数据库的配置管理，如修改数据库的日志模式、设置数据库的保护级别、启动和停止备用端的日志应用、主备用数据库的角色切换等。

（1）创建与恢复备用数据库控制文件：

```
alter database create physical standby controlfile as '...';
RMAN> backup current controlfile for standby;
RMAN> restore standby controlfile from '...';
```

（2）设置 Data Guard 的保护模式：

```
alter database set standby database to maximize performance;
alter database set standby database to maximize availability;
alter database set standby database to maximize protection;
```

（3）启动、停止、终止、切换物理备用数据库的 Redo Apply：

```
alter database recover managed standby database [using current
logfile] disconnect;
```

```
alter database recover managed standby database cancel;
alter database recover managed standby database finish;
```

将物理备用数据库切换为逻辑备用数据库：

```
alter database recover to logical standby new_db_name|keep
identity;
```

（4）启动、停止逻辑备用数据库的 SQL Apply：

```
alter database start logical standby apply [immediate];
alter database stop logical standby apply;
alter database abort logical standby apply;
```

（5）物理备用数据库与快照备用数据库之间的切换：

```
alter database convert to snapshot standby;
alter database convert to physical standby;
```

（6）主备用数据库角色之间的切换：

```
alter database commit to switchover to physical standby [with
session shutdown];
alter database commit to switchover to primary [with session
shutdown];
alter database commit to switchover to logical standby [with
session shutdown];
```

（7）激活备用数据库（切换为独立数据库）：

```
alter database activate physical standby database [finish
apply];
alter database activate logical standby database [finish
apply];
```

（8）主数据库与逻辑备用数据库角色切换的准备与取消：

```
alter database prepare to switchover to logical standby [with
session shutdown];
alter database prepare to switchover to primary [with session
shutdown];
alter database prepare to switchover cancel;
```

（9）向物理备用数据库或逻辑备用数据库注册重做日志：

```
alter database register [or replace] physical logfile 'file_
spec';
```

```
alter database register [or replace] logical logfile 'file_
spec';
```

（10）设置逻辑备用数据库的防护状态：

```
alter database guard all;
alter database guard standby;
alter database guard none;
```

（11）创建与删除备用日志组及其日志成员：

```
alter database add standby logfile [thread i] group n (member_
spec) size member_size_spec;
alter database add standby logfile member 'member_spec' to
group n;
alter database drop standby logfile group n;
alter database drop standby logfile member 'member_spec';
```

（12）设置数据库的强制日志、归档、闪回：

```
alter database force logging | noforce logging;
alter database archivelog | noarchivelog;
alter database flashback on | off;
```

（13）为数据库添加补充日志，为表添加非依赖约束：

```
alter database add supplemental log data(primary key,unique
index) columns;
alter table table_name add primary key(column_spec) reley
disable;
```

11.3.2　管理实例

管理实例系列指令用于 Data Guard 环境下的 ALTER SYSTEM 指令，主要用于主备用数据库实例的操作，如手工切换日志、将主数据库的重做日志刷新至备用数据库端等。

```
alter system switch logfile;
alter system archive log [instance instance_name]
            current | next | all | sequence seq_number;
alter system checkpoint [global | local];
alter system {disconnect | kill} session 'sid, serial#'
[immediate];
alter system flush redo to stdby_db_unique_name [confirm apply];
```

```
alter system set parameter_name = parameter_value
[scope=spfile];
```

11.3.3 管理会话

管理会话系列指令用于 Data Guard 环境下的 ALTER SESSION 指令，主要用于数据库会话特性的调整，如设置会话参数、更改防护状态等。

```
alter session set parameter_name = parameter_value;
alter session {enable | disable} guard;
alter session sync with primary;
```

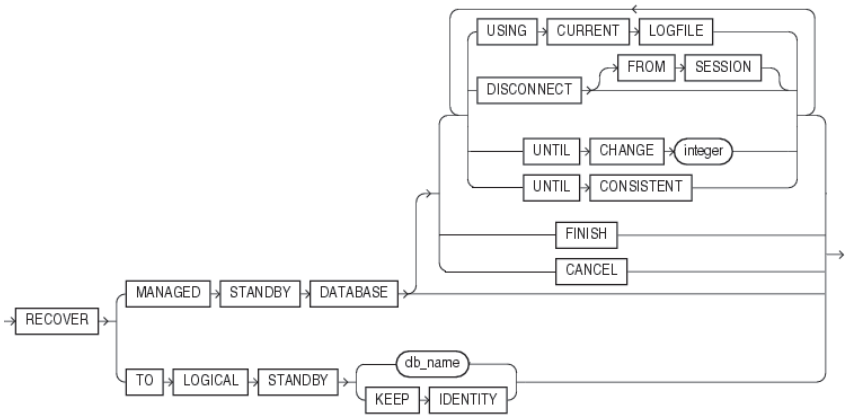
上述 ALTER SESSION 指令中，alter session enable|disable guard 仅用于逻辑备用数据库；alter session sync with primary 仅用于物理备用数据库（且要求日志传输模式为同步 SYNC，并启动实时应用），该指令主要用于物理备用数据库的实时查询环境，确保会话查询的结果与主数据库同步。为了确保在实时查询环境中获得与主数据库的同步查询结果，可以在数据库中创建一个会话级的触发器，类似如下：

```
CREATE TRIGGER tri_sess_logon_sync
AFTER LOGON ON user.schema
begin
if (SYS_CONTEXT('USERENV', 'DATABASE_ROLE') IN ('PHYSICAL
STANDBY')) then
     execute immediate 'alter session sync with primary';
end if;
end tri_sess_logon_sync;
```

11.3.4 常用指令流程图

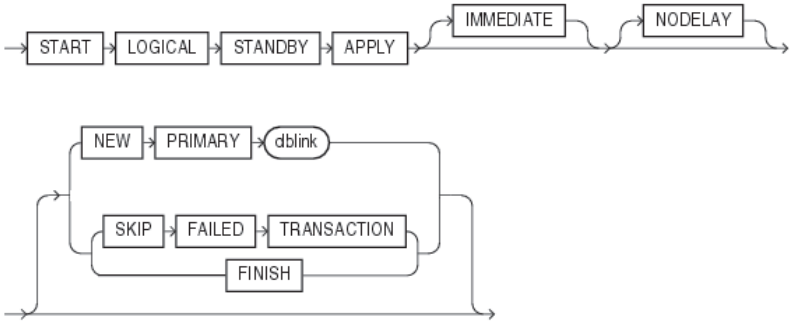
（1）启动物理备用数据库的托管恢复，或将物理备用数据库切换为逻辑备用数据库。

```
ALTER DATABASE ...
```



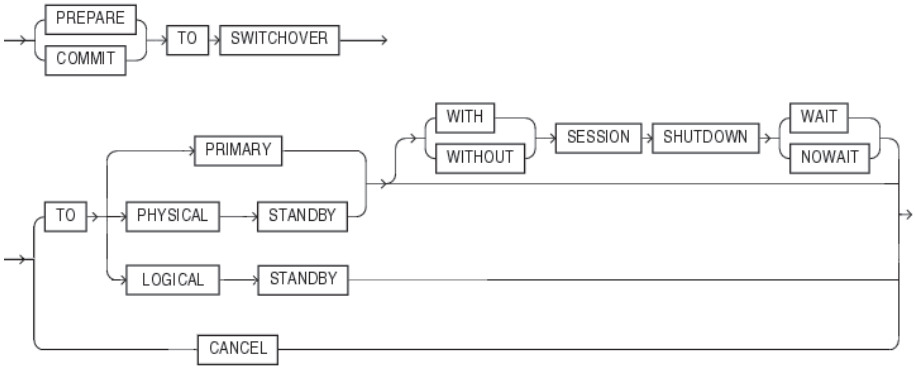
（2）启动逻辑备用数据库的 SQL Apply。

```
ALTER DATABASE ...
```



（3）Data Guard 数据库角色切换的准备与切换。

```
ALTER DATABASE ...
```

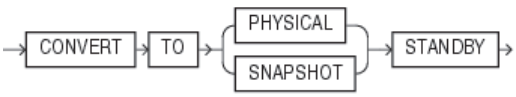


（4）直接激活物理备用或逻辑备用数据库。

```
ALTER DATABASE ...
```

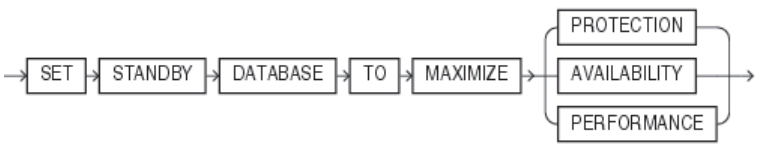
（5）物理备用数据库与快照备用数据库的相互转换。

```
ALTER DATABASE ...
```



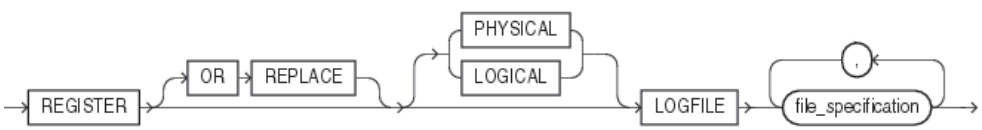
（6）设置主数据库的保护模式。

```
ALTER DATABASE ...
```



（7）在主备用数据库中注册重做日志文件。

```
ALTER DATABASE ...
```



11.4 PL/SQL 程序包

Data Guard 环境下 PL/SQL 程序包的使用主要体现在日志挖掘、逻辑备用数据库管理两个方面。在日志挖掘中，有两个需要用到的程序包：DBMS_LOGMNR 和 DBMS_LOGMNR_D，在逻辑备用数据库的配置和维护中，需要使用的是 DBMS_LOGSTDBY 程序包。对这 3 个 PL/SQL 程序包的熟悉和运用是 DBA 从事日志挖掘和逻辑备用数据库管理必备的技能。

11.4.1　DBMS_LOGMNR 程序包

DBMS_LOGMNR 程序包包含配置 LogMiner、启动和停止 LogMiner 会话的过程。

```
DBMS_LOGMNR.ADD_LOGFILE (
LogFileName IN VARCHAR2,
options IN BINARY_INTEGER default ADDFILE );
```

过程 ADD_LOGFILE 的参数 options 的取值是 DBMS_LOGMNR 程序包常量。

※ DBMS_LOGMNR.NEW：新建一个供挖掘的日志文件列表。

※ DBMS_LOGMNR.ADDFILE：给当前日志文件列表添加一个新的日志文件。

```
DBMS_LOGMNR.REMOVE_LOGFILE (
LogFileName IN VARCHAR2);

DBMS_LOGMNR.START_LOGMNR (
startScn            IN     NUMBER        default 0,
endScn              IN     NUMBER        default 0,
startTime           IN     DATE          default '01-jan-1988',
endTime             IN     DATE          default '31-dec-2110',
DictFileName        IN     VARCHAR2      default '',
Options             IN     BINARY_INTEGER      default 0 );
```

其中，参数 options 的取值是 DBMS_LOGMNR 常量，用来设置 LogMiner 字典的使用方式、LogMiner 日志挖掘行为及挖掘结果的输出格式等，常用的常量有：

※ DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG。

※ DBMS_LOGMNR.DICT_FROM_REDO_LOGS。

※ DBMS_LOGMNR.COMMITTED_DATA_ONLY。

※ DBMS_LOGMNR.SKIP_CORRUPTION。

※ DBMS_LOGMNR.DDL_DICT_TRACKING。

※ DBMS_LOGMNR.CONTINUOUS_MINE。

※ DBMS_LOGMNR.PRINT_PRETTY_SQL。

```
PROCEDURE END_LOGMNR();
```

该过程没有参数，用来结束 LogMiner 会话。

11.4.2 DBMS_LOGMNR_D 程序包

DBMS_LOGMNR_D 程序包的主要用途是在源数据库（或主数据库）中构建 LogMiner 字典并将其存储到一个特定的 OS 文件中或一个到多个归档日志文件中。

```
DBMS_LOGMNR_D.BUILD (
dictionary_filename IN VARCHAR2,
dictionary_location IN VARCHAR2,
options IN NUMBER);
```

其中，参数 options 是 DBMS_LOGMNR_D 程序包常量，有如下两个选项。

※ DBMS_LOGMNR_D.STORE_IN_FLAT_FILE：将 LogMiner 字典提取至数据库之外的 OS 文件。

※ DBMS_LOGMNR_D.STORE_IN_REDO_LOGS：将 LogMiner 字典提取至数据库的归档日志文件，由视图 V$ARCHIVED_LOG 的 DICTIONARY_BEGIN 和 DICTIONARY_END 字段标识。

当选择 STORE_IN_FLAT_FILE 方式时，必须指定参数 dictionary_filename 和参数 dictionary_location。位置参数 dictionary_location 指定的值必须和实例初始化参数 UTL_FILE_DIR 设置值一致。

11.4.3 DBMS_LOGSTDBY 程序包

DBMS_LOGSTDBY 程序包提供一系列子程序用于配置和管理 Data Guard 中的逻辑备用数据库环境，下面介绍其中的主要内容。

```
DBMS_LOGSTDBY.APPLY_SET (
inname IN VARCHAR,
value IN VARCHAR);
```

该过程通过一系列参数来设置逻辑备用数据库中 SQL Apply 的环境和行为。参数 inname 的取值常用的有：

※ LOG_AUTO_DELETE。

※ LOG_AUTO_DEL_RETENTION_TARGET，默认值是 1440 分钟。

※ MAX_SERVERS。

※ PREPARE_SERVERS。

※ APPLY_SERVERS。

※ MAX_SGA，设置 SGA 中的 LCR 缓存大小。

※ PRESERVE_COMMIT_ORDER。

参数 inname 的如下取值将影响 SQL Apply 运行过程中 DBA_LOGSTDBY_EVENTS 视图中的内容。

※ EVENT_LOG_DEST，取值 DEST_EVENTS_TABLE（默认）和 DEST_ALL，前者表示仅将与用户数据相关的“事件”记录并反映到 DBA_LOGSTDBY_EVENTS 中；后者表示将所有的“事件”同时记录并反映到 DBA_LOGSTDBY_EVENTS 视图和数据库告警日志（Alert Log）中。

※ MAX_EVENTS_RECORDED。

※ RECORD_APPLIED_DDL。

※ RECORD_SKIP_DDL。

※ RECORD_SKIP_ERRORS。

※ RECORD_UNSUPPORTED_OPERATIONS。

```
DBMS_LOGSTDBY.APPLY_UNSET (
inname IN VARCHAR);
```

使用该过程将 SQL Apply 的设置参数恢复到默认值。实际的参数设置值可查看数据字典 DBA_LOGSTDBY_PARAMETERS 视图。

```
DBMS_LOGSTDBY.BUILD();
```

该过程没有参数，该过程需要在主数据库中调用，其作用：一是提取 LogMiner 字典至归档日志；二是在主数据库中启动补充日志（最小、主键、唯一性索引）。

```
DBMS_LOGSTDBY.SKIP (
stmt IN VARCHAR2,
schema_name IN VARCHAR2 DEFAULT NULL,
object_name IN VARCHAR2 DEFAULT NULL,
proc_name   IN VARCHAR2 DEFAULT NULL,
use_like    IN BOOLEAN DEFAULT TRUE,
esc IN CHAR(1) DEFAULT NULL);
```

该过程控制逻辑备用数据库 SQL Apply 过滤（不应用）的内容。参数 stmt 通过一系列关键字标识一组或一类 SQL 语句；参数 schema_name 和 object_name 指定模式名和对象名，可以使用通配符（%）。

参数 proc_name 指定当 SQL Apply 过滤掉由参数 stmt、schema_name、object_name 设置的内容时调用的过程名，格式为 'schema.package.procedure'，该过程由 DBA 根据需要特别创建。

```
DBMS_LOGSTDBY.MAP_PRIMARY_SCN (
primary_scn NUMBER) RETURN NUMBER;
```

在逻辑备用数据库中调用该函数获得主数据库指定 SCN（参数 primary_scn）所对应的逻辑备用数据库的 SCN 值。注意，这两个 SCN 不是一一对应的关系，该函数返回一个保守的、可靠的 SCN（超前主数据库 SCN）。

调用这个函数的用途在于：当主数据库执行了数据库闪回操作（回到特定的 SCN 时刻）后，在逻辑备用数据库中需要做补偿操作（将逻辑备用数据库闪回到主数据库之前的 SCN 时间点）。通过该函数确定在逻辑备用数据库中需要补偿闪回的程度。

第12章

RMAN 备份与恢复技术

RMAN（Recovery Manager）是 Oracle 最初为实现数据库物理备份与介质恢复方案而提供的专门工具，且功能得到不断扩展，目前，RMAN 已是一个基于数据库备份管理的综合工具。在 Data Guard 环境的配置与管理过程中我们也多次用到该工具。本章介绍 RMAN 备份与恢复技术的主要使用方法。

12.1 熟悉 RMAN 环境

12.1.1 RMAN 环境的构成

第 2 章已经介绍了 Oracle 数据库基于重做日志的介质恢复的主要思想，RMAN 是实现 Oracle 数据库物理备份、日志管理、介质恢复的完美工具。RMAN 使用环境的主要构成如下：

※ RMAN 客户端程序（RMAN Client）。

※ RMAN 服务器进程（RMAN Server Process）。

※ 目标数据库（Target Database）。

※ 快照控制文件（Snapshot Controlfile）。

※ RMAN 存储库（RMAN Repository）。

※ 恢复目录（Catalog）及恢复目录数据库（Catalog Database）。

※ 通道（Channel）。

※ 快速恢复区（Flash/Fast Recovery Area）。

※ 辅助数据库实例（Auxiliary Database Instance）。

※ 介质管理层（Media Management Layer）。

RMAN 客户端程序（RMAN Client）是 Oracle 提供的 RMAN 可执行程序，该程序作为 Oracle 执行物理备份、还原、恢复等操作的客户端接口，用户可以在本地或远程启动该接口。

目标数据库（Target Database）就是 RMAN 需要对其进行备份的数据库。

RMAN 服务器进程（RMAN Server Process）是 RMAN 连接到目标数据库时在服务器上生成的进程，它执行 RMAN 的操作指令。RMAN 在目标数据库中使用两个 PL/SQL 程序包（DBMS_RCVMAN 和 DBMS_BACKUP_RESTORE）来实现 RMAN 的所有功能。当 RMAN 执行备份时，需要建立与目标数据库的连接，该连接会在目标数据库中创建两个会话（两个会话在 V$SESSION 视图中可见）。这两个会话对应两个服务器进程，一个主进程用于执行对程序包的调用，从而实现 RMAN 的备份、恢复等操作，另一个次进程用于轮询前一个进程的各种内部操作（Long-running Operations 或 Long-running Transaction）并及时将各种操作结果记录至 RMAN 存储库（Repository）。在 RMAN 执行操作期间，可以在视图 V$SESSION_LONGOPS 中看到次进程会话及其轮询的结果、进度等。

当 RMAN 操作分配通道（Channel）时，Oracle 会额外开辟对应的会话，对应在 V$SESSION 视图中可看到如下会话信息：

```
SQL>select username,sid,serial#,program,paddr,client_info
from v$session where username is not null;

USERNAME    SID    SERIAL#   PROGRAM   CLIENT_INFO
-----------------------------------------------------
SYS         88     44        rman
SYS         90     7         rman      rman channel=ORA_DISK_1
SYS         91     29        rman
SYS         105    3         sqlplus

SQL> select sid,serial#, opname,sofar,totalwork,
round(100*sofar/totalwork,2) "%_progress"
from v$session_longops
where opname like '%RMAN%'
and totalwork <> 0;

 SID    SERIAL# OPNAME          SOFAR  TOTALWORK    %_progress
-----------------------------------------------------
...
90      7   RMAN: full datafile backup   54679   76800   71.20
...
```

从上述查询中可以了解到，通道（Channel）事实上是 RMAN 在目标数据库中建立的用于操作备份目的地（备份设备）的特定会话。利用该会话，Oracle 将 RMAN 备份结果以特定格式写入备份目的地。根据备份目的地的不同，RMAN 的通道分为磁盘通道（备份设备类型为磁盘）和磁带通道（备份设备类型为磁带），且在某一时间，RMAN 只能使用一种通道执行备份操作，即 RMAN 不能同时分配两种类型的通道。

快照控制文件（Snapshot Controlfile）是 RMAN 在执行联机数据库备份时必须创建的特定控制文件，它是联机控制文件在备份时的静态版本（或称联机控制文件的一致性视图），RMAN 依据其中的内容执行后续的备份操作。数据库在运行过程中，控制文件是处于动态读 / 写的状态，其中的某些内容处于时刻变化当中（如检查点信息、SCN 信息等），一旦启动了对数据库的备份，RMAN 需要一个稳定的、一致状态下的控制文件信息，因此，RMAN 每次在启动联机数据库备份时都会创建快照控制文件，以供本次的备份过程使用。

RMAN 在执行备份的过程中，除产生备份结果本身外，还会产生关于备份的元数据（Metadata），RMAN 关于备份与恢复的元数据的集合称为恢复存储库（Recovery Repository）。这个存储库在 RMAN 备份与恢复的环境中有两个存储地，一个是目标数据库的控制文件，另一个是恢复目录（Recovery Catalog）。恢复目录是一个特定的用户模式，RMAN 通过该模式下的一套库表和视图（表现为 RC_* 形式）专门用来存储和管理 Repository 中的信息。恢复目录应存在于目标数据库之外的数据库中，存储恢复目录的数据库称为恢复目录数据库。

辅助数据库实例（Auxiliary Database Instance）是 RMAN 环境中在执行某些特定操作（如数据库复制、创建备用数据库等）需要的特定数据库实例，RMAN 连接到该实例后，可以根据目标数据库并结合用户需要构建特定用途的数据库。

介质管理层（Media Management Layer）是 RMAN 使用磁带设备所必需的，它是 RMAN 与磁带设备之间的软件层（通常包括 RMAN 接口、磁带设备本身的驱动等），RMAN 通过介质管理层软件来读 / 写磁带设备。

快速恢复区（Flash/Fast Recovery Area）是从 Oracle 10g 开始诞生的一个专门的存储区域，它专门用来存储、管理与数据库备份与恢复有关的各类数据。当目标数据库设置了快速恢复区后，数据库归档日志默认的本地存储目的地就是快速恢复区，RMAN 默认的备份目的地也是快速恢复区。快速恢复区中的存储文件采用 OMF（Oracle Managed File）形式管理。启动快速恢复区后，在目标数据库和 RMAN 中可以配置 FRA 文件和备份的保留策略。快速恢复区可以存储如下类型的文件：

※ 控制文件的联机镜像（CONTROL FILE）。

※ 联机重做日志组的成员（REDO LOG）。

※ 归档重做日志（ARCHIVED LOG）。

※ RMAN 备份片（BACKUP PIECE）。

※ RMAN 的映像复制（IMAGE COPY）。

※ 数据库闪回日志（FLASHBACK LOG）。

※ 逻辑备用数据库备用重做日志的归档（FOREIGN ARCHIVED LOG）。

下面的两个查询给出快速恢复区的使用信息。前者给出快速恢复区的整体使用信息，后者按上面的文件类型给出空间占用和可回收情况。

```
SQL> select * from v$recovery_file_dest;
SQL> select * from v$recovery_area_usage;
```

12.1.2 目标数据库的配置

目标数据库应做如下配置：

（1）配置数据库使用快速恢复区，并设置足够的存储空间。

（2）数据库处于归档模式运行（自动归档），这是 RMAN 执行联机数据库备份的必要条件。

（3）启动数据库闪回功能，并设置 db_flashback_retention_target 参数。

（4）设置参数 control_file_record_keep_time 配合 RMAN 的保留策略。该参数设置控制文件中的 Repository 信息保留的天数，这个设置值应该大于一个备份周期备份保留的天数。

12.1.3 目录数据库的配置

虽然恢复目录在 RMAN 环境中是可选的，但 Oracle 建议使用恢复目录，特别是在多数据库环境、Data Guard 环境中，因为使用恢复目录，DBA 可以实现更安全、灵活的备份管理和恢复策略，包括备份脚本的管理，同时，也可实现多个数据库备份信息的集中管理。在 Data Guard 环境中，如果要同时监控和管理主备用数据库的备份与恢复，就应该使用恢复目录，在主备用数据库之外创建恢复目录数据库，从而集中管理 Data Guard 环境下的主备用数据库。

使用 RMAN 恢复目录的主要优点如下：

（1）恢复目录可以集中存储多个数据库的备份 Repository 信息。

（2）使用恢复目录，可以在 Repository 中永久保存某些重要的备份，同时也可以持久保存 RMAN 的配置信息（如通道设置、备份策略等）。

（3）如果要使用 RMAN 存储脚本，则必须使用恢复目录。

创建与配置恢复目录的主要步骤如下：

（1）创建恢复目录数据库，即恢复目录需要驻留的数据库，该数据库应该位于目标数据库之外的独立的主机上。

（2）在恢复目录数据库中创建一个专门的永久表空间用于存储恢复目录。

（3）创建恢复目录模式用户，代码如下：

```
SYS@CATDB>create user catowner identified by catowner
  2  default tablespace catalog_tbs;

User created.

SYS@CATDB>grant connect,resource,recovery_catalog_owner
  2  to catowner;

Grant succeeded.
```

（4）创建恢复目录模式对象，代码如下：

```
rman target / catalog catowner/password@catdb

Recovery Manager: Release 11.2.0.1.0 - Production on Wed Mar 6
15:57:24 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORADB (DBID=3949937918)
connected to recovery catalog database

RMAN> create catalog tablespace catalog_tbs;

recovery catalog created
```

下面的查询可以查看恢复目录中的模式对象：

```
SYS@CATDB> connect catowner/password@catdb
CATOWNER@CATDB> select object_name,object_type from user_
objects;
...
```

（5）将目标数据库注册至恢复目录。

向恢复目录中注册目标数据库的实质是，将目标数据库控制文件中的已有Recovery Repository信息写入恢复目录模式中的相关库表。

```
rman target / catalog catowner/password@catdb

RMAN> register database;

database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

相关的其他操作，如同步恢复目录、将归档日志注册至恢复目录、将之前的手工备份注册至恢复目录、向恢复目录注册某个位置下的所有文件或注册当前快速恢复区的所有内容，等等。示例如下：

```
RMAN> resync catalog;

starting full resync of recovery catalog
full resync complete

RMAN> catalog backuppiece 'files_spec';
RMAN> catalog archivelog 'files_spec';
RMAN> catalog start with 'directory_spec';
RMAN> catalog datafilecopy 'files_spec';
RMAN> catalog recovery area;

searching for all files in the recovery area

List of Files Unknown to the Database
=====================================
...

Do you really want to catalog the above files (enter YES or
NO)? yes
cataloging files...
cataloging done

List of Cataloged Files
=======================
```

```
...
```

取消某个数据库在恢复目录中的注册，代码如下：

```
rman target / catalog catowner/password@catdb

RMAN> unregister database [db_name];

database name is "ORADB" and DBID is 3949937918

Do you really want to unregister the database (enter YES or
NO)? yes
database unregistered from the recovery catalog
```

当恢复目录中注册有多个同名的数据库时，需要事先设置 DBID 信息，代码如下：

```
rman catalog catowner/password@catdb

RMAN> run {
      set DBID 3949937918;
      unregister database [db_name] noprompt; }
```

恢复目录在维护已注册的目标数据库的 Recovery Repository 信息时，会记录所有的历史备份信息，包括超出 RMAN 保留策略、已删除的备份信息（带 delete 标记的记录），这些信息的积累会使得恢复目录模式不断扩张，影响 RMAN 在恢复目录中检索信息的效率。为此，Oracle 提供了一个专门的脚本用于清理这些信息，该脚本位于如下位置：$ORACLE_HOME/rdbms/admin/ prgrmanc.sql，必要时 DBA 可手工运行该脚本。

12.1.4　配置 RMAN 执行环境

在使用 RMAN 对目标数据库执行备份与恢复操作之前，我们需要了解 RMAN 本身的一系列配置和选项，这些配置和选项会在多个方面影响 RMAN 默认的操作行为。下面是一个 Oracle 数据库关于 RMAN 的默认配置列表。

```
rman target /

Recovery Manager: Release 11.2.0.1.0 - Production on Wed Mar 6
10:41:58 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
```

```
connected to target database: ORADB (DBID=3949937918)

RMAN> show all;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name
ORADB are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK
TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO
BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1;
# default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE
'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/DATABASE/SNCFORADB.
ORA'; # default
```

上述代码是 RMAN 环境的默认设置（行结尾处有 #default 指示），由此可以了解 RMAN 需要配置的主要内容。结合上述 show all 的输出循序，下面依次给出每一行代表的配置含义。

※ 备份的保留策略（RETENTION POLICY）。

※ 是否执行备份优化（BACKUP OPTIMIZATION）。

※ 存储备份的默认设备类型（DEFAULT DEVICE TYPE）。

※ 数据库控制文件和 SPFILE 的自动备份及其设备类型、命名格式。

※ 默认的执行备份的并行度（PARALLELISM）、备份结果的类型。

※ 备份是否压缩(COMPRESSED BACKUPSET)及其压缩算法(COMPRESSION ALGORITHM)。

※ 数据文件和归档日志映像复制的存储设备类型及其复制数(BACKUP COPIES)。

※ 备份集的最大尺寸(MAXSETSIZE)。

※ 备份是否加密及其加密算法(ENCRYPTION ALGORITHM)。

※ 归档日志的保留策略(ARCHIVELOG DELETION POLICY)。

※ 快照控制文件的存储路径及文件名(SNAPSHOT CONTROLFILE)。

下面结合实例介绍 RMAN 的主要配置。

(1)配置备份的保留策略,代码如下:

```
RMAN> configure retention policy to recovery window of 7 days;

new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
new RMAN configuration parameters are successfully stored

RMAN> report obsolete;

RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 7 days
Report of obsolete backups and copies
Type           Key  Completion Time    Filename/Handle
--------------------------------------------------------------------
Backup Piece   956  21-FEB-13  ...20130221T101934_8LC11QFO_.BKP

RMAN> delete obsolete;

RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 7 days
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=115 device type=DISK
Deleting the following obsolete backups and copies:
Type         Key  Completion Time    Filename/Handle
---------------- ------ ------------------ ---------------
Backup Piece   956  21-FEB-13 ...20130221T101934_8LC11QFO_.BKP
```

```
Do you really want to delete the above objects (enter YES or
NO)? yes
deleted backup piece
backup piece handle=...20130221T101934_8LC11QFO_.BKP RECID=18
STAMP=807963575
Deleted 1 objects
```

（2）配置归档日志的删除策略，代码如下：

```
RMAN> configure archivelog deletion policy to backed up 1 times
to device type disk;

new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 1 TIMES TO
DISK;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> configure archivelog deletion policy to applied on
standby;

old RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 1 TIMES TO
DISK;
new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON STANDBY;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
```

备注：配置归档日志删除策略为 applied on standby 指的是当归档日志传输并应用到至少一个备用数据库后，该归档日志被标记为 obsolete，可以被删除。该项配置要求主数据库至少有一个 MANDATORY 远程归档目的地，否则，给出如下提示：

```
RMAN-08591: WARNING: invalid archived log deletion policy.
```

（3）配置数据库无须备份的表空间，代码如下：

```
RMAN> configure exclude for tablespace example;
```

```
Tablespace EXAMPLE will be excluded from future whole database
backups
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
```

如果在备份时需要包含此处配置的排除表空间，在 backup 指令中使用 noexclude 关键字即可，如 backup database noexclude; 该指令执行对整个数据库的备份，包含前面配置的 example 表空间。

（4）配置备份压缩。默认情况下，RMAN 使用 NULL Data Block 压缩，即备份过程中对未使用的数据块（即空块）不进行备份，即使是执行映像复制也是如此。

对于备份集的压缩，RMAN 提供如下压缩级别的选择：

※ 低级压缩（LOW）。

※ 中级压缩（MEDIUM）。

※ 高级压缩（HIGH）。

需要注意的是，这里的压缩级别选择会影响到执行备份过程中 RMAN 对 CPU 资源的消耗，压缩级别越高，对 CPU 资源的消耗越高。

```
RMAN> show compression algorithm;

RMAN configuration parameters for database with db_unique_name
ORADB are:
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE
'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default

RMAN> configure compression algorithm "low";
RMAN> configure compression algorithm "medium";
RMAN> configure compression algorithm "high";
RMAN> configure compression algorithm clear;
```

12.1.5　配置备份通道与存储

通道是 RMAN 与目标数据库的连接在数据库中开辟的会话，负责执行将 RMAN 备份结果以特定格式写入备份设备。默认情况下，RMAN 的默认通道指向磁盘设备、存储的目的地是数据库的快速恢复区。

```
RMAN> configure default device type to sbt;

old RMAN configuration parameters:
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
new RMAN configuration parameters:
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT_TAPE';
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> configure default device type to disk;

old RMAN configuration parameters:
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT_TAPE';
new RMAN configuration parameters:
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> configure device type disk backup type to copy;
RMAN> configure device type disk backup type to backupset;
RMAN> configure device type disk backup type to compressed
backupset;
RMAN> configure device type sbt backup type to backupset;

RMAN> configure device type disk parallelism 2;

old RMAN configuration parameters:
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO
BACKUPSET;
new RMAN configuration parameters:
CONFIGURE DEVICE TYPE DISK PARALLELISM 2 BACKUP TYPE TO
BACKUPSET;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
```

```
RMAN> configure channel device type disk
format '/db_fra/backup_%U';

new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/db_fra/backup_%U';
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> configure channel 1 device type disk
format '/db_fra/ch1/backup_%U';
RMAN> configure channel 2 device type disk
format '/db_fra/ch1/backup_%U';
RMAN> configure channel 3 device type disk maxpiecesize 100m
format '/db_fra/ch3/backup_%U';

RMAN> configure maxsetsize to 500m;
```

影响 RMAN 备份结果（备份集、备份片）及其文件大小的有 maxpiecesize、maxsetsize、filesperset 三个参数，其中，filesperset 参数需要在 RMAN 的 backup 指令中设置。

```
RMAN> configure exclude for tablespace example clear;
RMAN> backup tablespace example filesperset 1;
```

12.1.6　磁带通道的磁盘模拟

在实际的工程实践中，如果备份系统配置了磁带机，RMAN 需要配置磁带设备，备份结果可以直接写至磁带，也可将磁盘设备上的备份结果备份至磁带。RMAN 与磁带设备交互需要 Oracle 与介质管理层软件的衔接，如图 12-1 所示为 RMAN 与磁带机交互的框架。

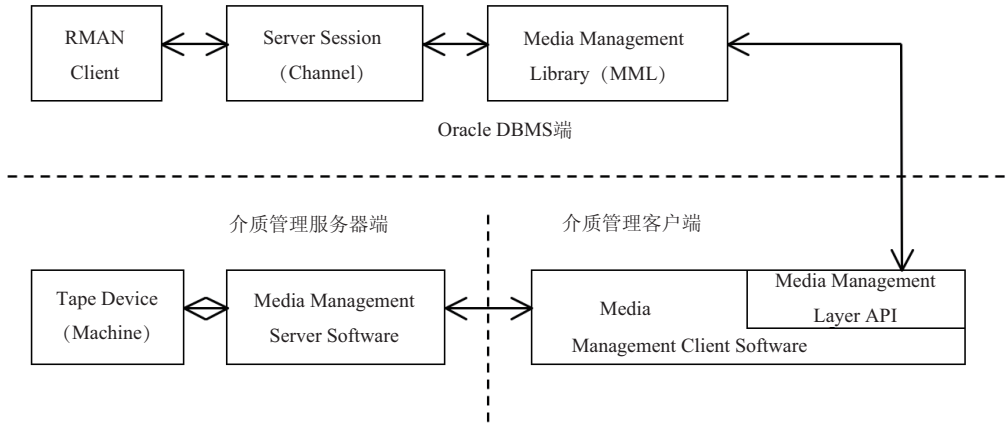


图 12-1　RMAN 与介质管理

图 12-1 中的介质管理库（MML）实际上是一个库文件，对应于配置 RMAN 磁带通道中的设置 SBT_LIBRARY=... 选项，它将 RMAN 的备份或还原操作的请求解释为对介质管理服务器软件的具体调用（Media Management Layer API），以实现 RMAN 要求的操作。MML 与 MM 服务器端软件、MM 客户端软件一样都是磁带设备供应商提供，需要 Oracle 之外的单独购买许可。

在对磁带通道的配置过程中，如果在参数 PARMS 中不指定 SBT_LIBRARY 选项，Oracle 在分配磁带通道时将加载一个默认的 MML 库文件，这个库文件与平台相关：

※ 在 UNIX/Linux 平台，该库文件是 libobk.so。

※ 在 Windows 平台，该库文件是 orasbt.dll。

实际工程中的磁带通道配置，应该类似如下：

```
RMAN> configure channel device type sbt
PARMS='SBT_LIBRARY=/lib/libobk.so
ENV=(NSR_SERVER=tape_srv, NSR_GROUP=oracle_tapes)';
```

或者在执行备份时分配通道：

```
RMAN> RUN {
  allocate channel c1 device type sbt
  PARMS='SBT_LIBRARY=/lib/libobk.so
  ENV=(NSR_SERVER=tape_srv, NSR_GROUP=oracle_tapes)';
  backup ... ; }
```

为了方便测试，Oracle 提供了通过磁盘设备来模拟磁带通道的默认库 oracle.disksbt，利用它可以使用磁盘目录来模拟磁带的存储。下面的示例使用磁盘目录 /DB_FRA/sbt 模拟磁带通道的磁带存储。

```
RMAN> RUN {
2> allocate channel ch_sbt device type sbt
3>   PARMS  'SBT_LIBRARY=oracle.disksbt,
4>   ENV=(BACKUP_DIR=/DB_FRA/sbt)';
5> backup tablespace example;
6> }

allocated channel: ch_sbt
channel ch_sbt: SID=115 device type=SBT_TAPE
channel ch_sbt: WARNING: Oracle Test Disk API

Starting backup at 08-MAR-13
channel ch_sbt: starting full datafile backup set
channel ch_sbt: specifying datafile(s) in backup set
input datafile file number=00004 name=/DATABASE/ORADB/EXMORADB.
DBF
channel ch_sbt: starting piece 1 at 08-MAR-13
channel ch_sbt: finished piece 1 at 08-MAR-13
piece handle=0qo41rk5_1_1 tag=TAG20130308T221533 comment=API
Version 2.0,MMS Version 8.1.3.0
channel ch_sbt: backup set complete, elapsed time: 00:00:03
Finished backup at 08-MAR-13
released channel: ch_sbt
```

12.2　RMAN 备份的主要形式

12.2.1　RMAN 备份的优点

RMAN 执行对数据库的物理备份。使用 RMAN 备份数据库要优于用户手工对数据库的备份，这主要体现在如下几个方面：

（1）RMAN 自动生成关于备份的元数据（Metadata），便于对备份结果的管理。

（2）根据目标数据库的控制文件，RMAN 可以自动跟踪数据库的物理存储。

（3）虽然是物理备份，但 RMAN 只会备份数据库使用过的数据块，对于空数据块，则不会对其备份，即使是映像复制。

（4）备份过程中 RMAN 会执行对数据块的检测，也可单独对要备份的内容执行物理的和逻辑的数据块检测。

（5）RMAN 可以执行联机数据库备份（数据库需要在归档模式下运行），当然

也可以执行脱机（mount 状态）的数据库备份。

（6）RMAN 可以执行增量备份，也可以执行压缩备份，还可以执行加密备份。

（7）DBA 可以方便监控 RMAN 对数据库的备份操作。如 RMAN 备份的会话（通道）及其备份进度可以通过视图 V$SESSION_LONGOPS 查看，RMAN 备份过程的输出可以通过视图 V$RMAN_OUTPUT 查看，等等。

12.2.2 备份目标和备份结果

RMAN 可以备份整个数据库，也可以备份数据库的某个部分，甚至是 RMAN 的备份结果也可以是 RMAN 的备份内容。RMAN 可以备份的内容如下：

※ 数据库（Database）。

※ 表空间（Tablespace）。

※ 数据文件（Datafile）。

※ 归档日志文件（Archivelog）。

※ 控制文件（Controlfile）。

※ 初始化参数文件（Spfile）。

※ 数据文件复制（DatafileCopy）。

※ 控制文件复制（ControlfileCopy）。

※ RMAN 的备份集（Backupset）。

※ 快速恢复区（Recovery Area）。

RMAN 对数据库的备份结果有如下 4 种形式：

※ 备份集（Backupset）。

※ 压缩备份集（Compressed Backupset）。

※ 映像复制（Image Copy）。

※ 备份片（Backup Piece）。

12.2.3 执行 RMAN 备份 BACKUP

RMAN 的备份操作通过 BACKUP 指令实现，该类指令有一系列的选项可以控制备份目标、备份结果、备份过程等。RMAN 指令可以有两种使用模式：一种是交互模式（Interactive Mode），直接使用 RMAN 的单个命令（Stand-alone）；另一种是批处理模式（Batch Mode），将一组相关的 RMAN 指令包含在 RUN{...} 块中执行。

```
RMAN> backup database plus archivelog;

RMAN> run {
2> allocate channel ch1 device type disk format '/backup/
ORADB_%U';
3> backup database plus archivelog;
4> }
```

RMAN 环境的 backup 指令的基本格式如下：

```
BACKUP ... PLUS ARCHIVELOG [backupSpecOperand];
```

下面列出常用的、基本的 RMAN 备份指令的使用形式。通过下面的形式 DBA 指定 RMAN 需要执行备份的内容。

```
RMAN> backup database;
RMAN> backup tablespace ...;
RMAN> backup datafile ...;
RMAN> backup current controlfile;
RMAN> backup spfile;
RMAN> backup archivelog ...;
RMAN> backup ... plus archivelog;
RMAN> backup backupset ...;
RMAN> backup recovery area;

RMAN> backup datafilecopy '...';
RMAN> backup controlfilecopy  '...';
RMAN> backup copy of database;
RMAN> backup copy of tablespace ...;
RMAN> backup copy of datafile ...;
```

通过 backup 指令的 skip 选项手工筛选掉部分的备份内容。

```
RMAN> backup ... skip readonly;
RMAN> backup ... skip offline;
RMAN> backup ... skip inaccessible;
```

通过 backup 指令的 not backed up 选项让 RMAN 在指定备份内容的基础上自动筛选出需要备份的内容。

```
RMAN> backup not backed up ...;
RMAN> backup not backed up x times ...;
RMAN> backup not backed up since time 'sysdate - n' ...;
```

通过 backup 指令的 as ... 选项控制备份结果的输出形态、设备类型及其文件的命名格式。此处的选项指定将覆盖 RMAN 的默认配置。

```
RMAN> backup as backupset ...;
RMAN> backup as compressed backupset ...;
RMAN> backup as copy ...;
RMAN> backup device type disk|sbt ...;
RMAN> backup format '...' ...;
```

通过 backup 指令的 keep 选项覆盖 RMAN 默认的备份保留策略。

```
RMAN> backup ... keep forever;
RMAN> backup ... keep until time 'sysdate + n';
```

12.2.4 Plus Archivelog 解析

RMAN 的 backup 指令可以附加对归档日志的备份，即存在如下形式的备份指令：

```
RMAN> backup ... plus archivelog;
```

典型地，DBA 使用 backup database plus archivelog; 指令备份整个数据库及其归档日志。为了便于读者理解，下面解释该指令的内部执行过程。

检查当前数据库的日志序列号：

```
SYS@ORADB>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     74
Next log sequence to archive   75
Current log sequence           75
```

检查数据库已有的归档日志：

```
RMAN> list archivelog all;

List of Archived Log Copies for database with db_unique_name
ORADB
=====================================================

Key         Thrd   Seq          S            Low Time
-------     ----   -------      -            ---------
582         1      73           A            10-MAR-13
```

```
        Name: .../O1_MF_1_73_8MS3G2XS_.ARC
590         1      74           A            10-MAR-13
        Name: .../O1_MF_1_74_8MS3HKBQ_.ARC
```

在 RMAN 中执行备份：

```
RMAN> backup database plus archivelog;

Starting backup at 10-MAR-13
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=73 RECID=582
STAMP=809732547
input archived log thread=1 sequence=74 RECID=590
STAMP=809732593
input archived log thread=1 sequence=75 RECID=592
STAMP=809732805
channel ORA_DISK_1: starting piece 1 at 10-MAR-13
channel ORA_DISK_1: finished piece 1 at 10-MAR-13
piece handle=.../O1_MF_ANNNN_TAG20130310T214645_8MS3P600_.BKP
tag=TAG20130310T214645 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 10-MAR-13

Starting backup at 10-MAR-13
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=/DATABASE/ORADB/SYSORADB.
DBF
input datafile file number=00002 name=/DATABASE/ORADB/AUXORADB.
DBF
input datafile file number=00004 name=/DATABASE/ORADB/EXMORADB.
DBF
input datafile file number=00003 name=/DATABASE/ORADB/UNDORADB.
DBF
```

```
input datafile file number=00005 name=/DATABASE/ORADB/USRORADB.
DBF
channel ORA_DISK_1: starting piece 1 at 10-MAR-13
channel ORA_DISK_1: finished piece 1 at 10-MAR-13
piece handle=.../O1_MF_NNNDF_TAG20130310T214647_8MS3P7KL_.BKP
tag=TAG20130310T214647 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:55
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 10-MAR-13
channel ORA_DISK_1: finished piece 1 at 10-MAR-13
piece handle=.../O1_MF_NCSNF_TAG20130310T214647_8MS3QZRF_.BKP
tag=TAG20130310T214647 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 10-MAR-13

Starting backup at 10-MAR-13
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=76 RECID=593
STAMP=809732865
channel ORA_DISK_1: starting piece 1 at 10-MAR-13
channel ORA_DISK_1: finished piece 1 at 10-MAR-13
piece handle=.../O1_MF_ANNNN_TAG20130310T214745_8MS3R1BQ_.BKP
tag=TAG20130310T214745 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 10-MAR-13

RMAN> list backup;

List of Backup Sets
```

```
====================

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
46      344.00K    DISK        00:00:01     10-MAR-13
        BP Key: 46   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130310T214645
              Piece Name: ...\O1_MF_ANNNN_
TAG20130310T214645_8MS3P600_.BKP

  List of Archived Logs in backup set 46
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    73      1571543    10-MAR-13 1571966    10-MAR-13
  1    74      1571966    10-MAR-13 1572016    10-MAR-13
  1    75      1572016    10-MAR-13 1572277    10-MAR-13

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------- ---- -- ---------- ----------- ------------ ----------
47      Full    974.22M    DISK        00:00:46     10-MAR-13
        BP Key: 47   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130310T214647
              Piece Name: ...\O1_MF_NNNDF_
TAG20130310T214647_8MS3P7KL_.BKP
  List of Datafiles in backup set 47
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1    Full 1572285  10-MAR-13 /DATABASE/ORADB/SYSORADB.DBF
  2    Full 1572285  10-MAR-13 /DATABASE/ORADB/AUXORADB.DBF
  3    Full 1572285  10-MAR-13 /DATABASE/ORADB/UNDORADB.DBF
  4    Full 1572285  10-MAR-13 /DATABASE/ORADB/EXMORADB.DBF
  5    Full 1572285  10-MAR-13 /DATABASE/ORADB/USRORADB.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
48      Full    9.95M      DISK        00:00:02     10-MAR-13
        BP Key: 48   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130310T214647
```

```
Piece Name: .../O1_MF_NCSNF_TAG20130310T214647_8MS3QZRF_.BKP
  SPFILE Included: Modification time: 10-MAR-13
  SPFILE db_unique_name: ORADB
  Control File Included: Ckp SCN: 1572338  Ckp time: 10-MAR-13

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
49      29.50K     DISK        00:00:00     10-MAR-13
        BP Key: 49   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130310T214745
Piece Name: .../O1_MF_ANNNN_TAG20130310T214745_8MS3R1BQ_.BKP

  List of Archived Logs in backup set 49
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    76      1572277    10-MAR-13 1572345    10-MAR-13

SYS@ORADB>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     76
Next log sequence to archive   77
Current log sequence           77
```

检查上述 RMAN 备份指令及其随后查询的输出内容，可以清楚地看到本次备份产生了 4 个备份集，对应的备份集编号分别为 46、47、48、49（测试备份之前数据库没有备份），备份过程中导致数据库产生了两次日志切换（测试的备份过程中无用户操作导致日志切换），日志序列号由备份前的 75 变为当前的 77。具体的产生过程如下：

（1）归档当前的联机日志（序列号为 75），产生了一次日志切换。此步相当于执行 alter system archivelog current 指令。

（2）备份数据库已有的归档日志，对应序列号为 73、74、75，产生第一个备份集 46。

（3）备份数据库的数据文件，产生第二个备份集 47。

（4）备份数据库的 SPFILE 和控制文件，产生第三个备份集 48。

（5）再次归档当前的联机日志（备份期间产生的日志），导致第二次日志切换，

产生新的归档日志，对应日志序列号为 76。

（6）备份前一步操作产生的归档日志，产生第四个备份集 49，至此，整个备份结束。

由上述备份过程可以清楚地看出，执行 backup database plus archivelog; 备份，备份结果中包含了足够的归档日志，确保利用本次备份可以将数据库恢复至备份结束时的状态。如果每次备份执行如下指令，还可以及时删除已经备份的归档日志。

```
RMAN> backup database plus archivelog delete input;
```

或者

```
RMAN> backup database plus archivelog delete all input;
```

12.3　执行 RMAN 增量备份

能够执行增量备份是 RMAN 区别于其他备份工具的一个显著优点。RMAN 的增量备份仅备份那些经历改变的数据块。使用增量备份带来的好处有：显著减少备份集的大小、节约备份时间等。但在恢复过程中使用增量备份也会显示出不利的一面，那就是增加了数据库的恢复时间，因为单独的增量备份不能执行数据库恢复（即需要多个备份集）。

12.3.1　启动块改变跟踪

RMAN 增量备份的本质是只备份自上次备份以来发生改变的数据块，因此，在默认情况下执行增量备份时，Oracle 需要搜寻所有的数据文件以找出内容已经变更的数据块，此操作会增加执行备份的时间。

为提高执行增量备份的效率，Oracle 提供了块改变跟踪（Block Change Tracking）功能，用于将发生改变的数据块信息记录到特定的文件中，利用该文件，Oracle 可以显著提高执行增量备份的效率。

启动数据库的块改变跟踪功能，代码如下：

```
SYS@ORADB>alter database enable block change tracking
  2  using file '/db_fra/ORADB_bct.ora';

Database altered.
```

已经启动了块改变跟踪功能的数据库，若要了解块改变跟踪文件的信息，可以通过查询视图 V$BLOCK_CHANGE_TRACKING 获得。禁用块改变跟踪及重新启动：

```
SYS@ORADB>alter database disable block change tracking;

Database altered.
```

```
SYS@ORADB>alter database enable block change tracking
  2  using file '/db_fra/ORADB_bct.ora' reuse;

Database altered.
```

块改变跟踪文件的初始大小与数据库大小有关，在使用过程中该文件的具体容量取决于两次 level 0 备份之间的增量备份的数量。增量备份越多，块改变跟踪文件中需要记录的改变块信息及其元数据量就会越大。块改变跟踪文件最多可以记录连续 8 次备份（1 次零级备份和 7 次增量备份）的块跟踪信息，因此，在实际的备份方案中，两次零级备份之间增量备份的次数一般不要超过 7 次。

12.3.2 增量备份的类别

执行增量备份，需要有一个完全备份的基础，这个基础就是 Level 0 的增量备份，在此基础上可以执行 Level 1 的增量备份。注意，虽然从备份的内容（需要备份的数据块）上来说，一次完全备份和 Level 0 的增量备份没有不同，但两者在 RMAN Repository 中的元数据的记录则不同，Level 1 的增量备份只能依赖于 Level 0 的增量备份，而不能依赖于某个完全备份。

根据需要备份的“增量”部分的不同，Level 1 的增量备份又分为差异增量备份（默认情况）和累积增量备份（Cumulative）。差异增量备份是备份自上次 Level 0 或 Level 1 的增量备份以来已经改变的数据块；累积增量备份是忽略已经执行的增量备份，备份自上次 Level 0 备份以来所有的已经变更的数据块。

※ 执行 Level 0 的增量备份：

```
RMAN> backup incremental level 0 database;
```

※ 执行 Level 1 的差异增量备份：

```
RMAN> backup incremental level 1 database;
```

※ 执行 Level 1 的累积增量备份：

```
RMAN> backup incremental level 1 cumulative database;
```

Level 0 的增量备份是 Level 1 增量备份的基础。如果在执行 Level 1 的增量备份时，数据库并没有 Level 0 增量备份，则 Oracle 自动将 Level 1 更改为 Level 0 执行增量备份。

上面的增量备份实例是在数据库级别执行的。RMAN 的增量备份也可以在表空间和数据文件级别执行。

虽然可以执行 Level 大于 1 的增量备份，但作者建议不要这样做，也不需要这样做。Level 大于 1 的增量备份只会增加管理增量备份的复杂性。

12.3.3　基于 SCN 的增量备份

这是一种特殊类型的增量备份，指定备份从特定 SCN 之后的所有的已变更的数据块。典型的用途在 Data Guard 环境中。由于某种原因，物理备用数据库已经严重滞后于主数据库，为了快速使物理备用数据库与主数据库同步，或者物理备用数据库缺乏必要的归档日志与主数据库同步，此时可先查询物理备用数据库的 SCN（V$DATABASE. CURRENT_SCN），然后，从主数据库中执行基于 SCN 的备份，将获得的增量备份集注册至物理备用数据库，启动介质恢复（必须带 NOREDO 选项），将物理备用数据库恢复至与主数据库同步的状态。

```
RMAN> backup incremental from scn scn_number_spec database;
...
SYS@PHYDB> recover database noredo;
```

12.4　RMAN 备份的其他选项

这里讨论的是 backup 指令中的 backupSpecOperand 选项。

12.4.1　Backup 指令的其他选项

（1）控制备份集的大小及其数据库文件数量。

```
RMAN> backup ... filesperset n maxsetsize xm;
```

（2）归档日志备份后删除对应的文件。

```
RMAN> backup archivelog ... delete input;
RMAN> backup archivelog ... delete all input;
```

数据库的归档日志常常有多个归档目的地。两者的区别在于 delete input 仅删除 backup 涉及的归档目的地的归档日志，而 delete all input 同时删除所有归档目的地的备份归档日志。

（3）选项 force 和 noexclude。Backup 指令的 force 选项是针对启动了备份优化配置的环境，该选项的含义是备份内容强制包含备份优化省略的备份内容（如包含未更改的数据文件、只读表空间等）。

选项 noexclude 是针对预先配置了 configure exclude for tablespace ... 的情况，该选线使备份内容包含 exclude 的表空间。

```
RMAN> backup ... force;
RMAN> backup ... noexclude;
```

（4）给备份添加标签。备份指令的 tag 选项用于给备份集贴上一个有意义的标签，方便对备份集的管理。在某些情况下（如后面介绍的增量更新备份）使用标签是必需的。

```
RMAN> backup database tag 'weekly_full_backup';
```

（5）检查数据块的选项。默认情况下，RMAN 在备份时执行对数据块的数据校验计算和检查。Checksum 是 Oracle 根据数据块内容计算出的数据校验值。当初始化参数 DB_BLOCK_CHECKSUM 设置为 typical 时，实例在读 / 写数据块时同样执行数据校验检查。

RMAN 的 backup 指令还可对数据块执行进一步的物理检查和逻辑检查，以判断待备份的数据块是否出现物理的或逻辑的讹误（Corruption）。此时，RMAN 仅执行数据块的检查，并不真正执行备份操作。

```
RMAN> backup nochecksum ...;
RMAN> backup validate ...;
RMAN> backup check logical ...;
```

实例：RMAN> backup validate check logical database;

（6）控制备份片的文件命名。默认情况下，RMAN 使用 OMF 特性自动（或使用通配符 %U）命名备份片，并存储于数据库的快速恢复区。也可使用 format 选项人为控制备份片的存储位置与命名规则，以覆盖 RMAN 环境中默认的或预先配置的命名规则。

常用的两个确保备份片唯一性命名的单一通配符是 %U 和 %F，两者都可以用来对备份片命名，%F 是 DBID 和时间（年月日及其备份顺序）信息的组合，默认用于对控制文件自动备份的命名，%U 是 8 位字符的时间表示（%u）和备份片编号（%p）、备份片的复制数（%c）三者的组合，默认用于对其他备份片的命名。

除此之外，还可根据需要使用其他意义的通配符命名，前提是命名规则的通配符组合必须满足唯一性命名的规则。

※ %d 或 %n：数据库名称或 8 位字符的数据库名。

※ %I：数据库 DBID。

※ %s：备份集编号。

※ %p：备份集下的备份片编号。

※ %c：备份集下的备份片复制数。

※ %Y：4 位年份 YYYY。

※ %M：两位月份 MM。

※ %D：两位日 DD。

※ %T：8 位日期格式 YYYYMMDD。

※ %u：8 位字符的时间表示。

※ %t：备份的时间戳。

※ %%：文件名中本身使用 % 符号。

12.4.2　限定 RMAN 备份的过程

如果直接在生产数据库中执行备份，为了避免 RMAN 的备份操作对生产系统的影响，可以在执行备份时设置对 RMAN 备份行为的限制，这个限制主要是两个方面，一是备份持续时间的限制，二是备份负荷的限制。为此，可以在 backup 指令中使用 duration、minimize 和 partial 选项。

```
RMAN> backup duration hh:mm ...;
RMAN> backup duration hh:mm partial ...;
RMAN> backup minimize time ...;
RMAN> backup minimize load ...;
```

下面的示例设置备份数据库的操作限制在一个半小时，minimize time 指示 RMAN 全速备份，超过一个半小时备份终止。Partial 选项指示 RMAN 由于时间限制而终止的备份并不会作为失败的备份处理，已经生成的备份片正常注册至 Repository，恢复时可用。

```
RMAN> backup duration 01:30 minimize time partial database;
```

12.5　管理 RMAN 的备份结果

在目标数据库中，可以查看 V$BACKUP_... 这样一类视图获得关于备份结果的信息，如视图 V$BACKUP_SET、V$BACKUP_PIECE 等，这些信息来源于目标数据库的控制文件，受到参数 control_file_record_keep_time 的限制。

在 RMAN 环境中，当连接至目标数据库后（或者同时连接至恢复目录数据库），可以使用一系列专门的指令来查看和管理 RMAN 的备份结果，比较常用的指令如下：

※ 指令 LIST。

※ 指令 REPORT。

※ 指令 DELETE。

※ 指令 CROSSCHECK。

※ 指令 CATALOG。

※ 指令 CHANGE。

12.5.1 查看备份结果 LIST

通过检索目标数据库控制文件或恢复目录中的相关内容，LIST 指令可以给出我们需要的关于备份结果的信息。

主要语法如下：

```
list [expired|recoverable] 查看对象 [of 数据库成分] [by file|summary];
```

这里的“查看对象”有：

```
backup, backupset, backuppiece, copy, archivelog, controlfilecopy, datafilecopy
```

这里的“数据库成分”有：

```
database, tablespace, datafile, archivelog, controlfile, spfile
```

```
RMAN> list backup;

List of Backup Sets
===================

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
784     Full    1.98M      DISK        00:00:00     09-JUN-13
        BP Key: 785   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130609T094008
        Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_09/O1_MF_
NNNDF_TAG20130609T094008_8V7Q7SBG_.BKP
  List of Datafiles in backup set 784
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  4      Full 859174     09-JUN-13 /DATABASE/PHYDB/EXAMPLE.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
845     Full    271.10M    DISK        00:00:13     09-JUN-13
        BP Key: 848   Status: EXPIRED  Compressed: NO  Tag:
TAG20130609T102742
```

```
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_09/O1_MF_
NNNDF_TAG20130609T102742_8V7T0Z0T_.BKP
  List of Datafiles in backup set 845
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 863862     09-JUN-13 /DATABASE/PHYDB/SYSTEM.DBF
  2       Full 863862     09-JUN-13 /DATABASE/PHYDB/SYSAUX.DBF
  3       Full 863862    09-JUN-13 /DATABASE/PHYDB/UNDOTBS.DBF
  4       Full 863862    09-JUN-13 /DATABASE/PHYDB/EXAMPLE.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
846     Full    9.83M      DISK        00:00:01     09-JUN-13
        BP Key: 849   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130609T102742
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_09/O1_MF_
NCSNF_TAG20130609T102742_8V7T1HDP_.BKP
  SPFILE Included: Modification time: 09-JUN-13
  SPFILE db_unique_name: PHYDB
   Standby Control File Included: Ckp SCN: 863862         Ckp
time: 09-JUN-13

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ----------
1182    Full    1.98M      DISK        00:00:00     10-JUN-13
        BP Key: 1185   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130610T143821
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_10/O1_MF_
NNNDF_TAG20130610T143821_8VBX2XKL_.BKP
  List of Datafiles in backup set 1182
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  4     Full 910589    10-JUN-13 /DATABASE/PHYDB/EXAMPLE.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ----------
1192    Incr 0  1.98M      DISK        00:00:15     10-JUN-13
```

```
        BP Key: 1194   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130610T143844
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_10/O1_MF_
NNND0_TAG20130610T143844_8VBX444F_.BKP
  List of Datafiles in backup set 1192
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  4    0  Incr 910589  10-JUN-13 /DATABASE/PHYDB/EXAMPLE.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ----------
1212    Incr 1  2.83M      DISK        00:00:16     10-JUN-13
        BP Key: 1214   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130610T144316
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_10/O1_MF_
NNND1_TAG20130610T144316_8VBXDN30_.BKP
  List of Datafiles in backup set 1212
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  4    1  Incr 918398   10-JUN-13 /DATABASE/PHYDB/EXAMPLE.DBF

RMAN> list backup of tablespace example summary;

List of Backups
===============
Key  TY LV S Device Completion Time #Pieces #Copies Compressed Tag
-------------------------------------------------------------------
784  B  F  A DISK 2013-06-09 09:40:09 1 1 NO TAG20130609T094008
845  B  F  X DISK 2013-06-09 10:27:55 1 1 NO TAG20130609T102742
1182 B  F  A DISK 2013-06-10 14:38:21 1 1 NO TAG20130610T143821
1192 B  0  A DISK 2013-06-10 14:39:00 1 1 NO TAG20130610T143844
1212 B  1  A DISK 2013-06-10 14:43:32 1 1 NO TAG20130610T144316
```

上述示例中，Key 列是备份的唯一标识。

LV 列表示备份的级别 Level，取值有 F 和 A，以及 0 和 1 值。F 即完全备份（Full），A 表示归档日志的备份，数字 0 表示增量备份中的 Level=0 的备份，数字 1 表示 Level=1 的增量备份。

S 列表示备份结果的状态（Status），取值有 A 和 X，A 表示备份可用

（Available），X 表示备份不可用（eXpired）。备份结果的状态为 Expired 意味着备份片被人工删除。Completion Time 备份完成时间列的显示格式取决于 NLS_DATE_FORMAT 环境变量。

TY 列表示备份的类型，取值有 B 和 P，B 表示备份集，P 表示代理备份（Proxy）。

```
RMAN> list recoverable backup of tablespace example;

List of Backup Sets
===================


BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
1182    Full    1.98M      DISK        00:00:00     2013-06-10
14:38:21
        BP Key: 1185   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130610T143821
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_10/O1_MF_
NNNDF_TAG20130610T143821_8VBX2XKL_.BKP
  List of Datafiles in backup set 1182
  File LV Type Ckp SCN    Ckp Time            Name
  ---- -- ---- ---------- ------------------- ----
  4  Full 910589  2013-06-10 09:40:34 /DATABASE/PHYDB/EXAMPLE.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
1192  Incr 0  1.98M    DISK   00:00:15     2013-06-10 14:39:00
        BP Key: 1194   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130610T143844
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_10/O1_MF_
NNND0_TAG20130610T143844_8VBX444F_.BKP
  List of Datafiles in backup set 1192
  File LV Type Ckp SCN    Ckp Time            Name
  ---- -- ---- ---------- ------------------- ----
  4 0 Incr 910589 2013-06-10 09:40:34 /DATABASE/PHYDB/EXAMPLE.DBF

BS Key  Type LV Size  Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -----------
1212    Incr 1  2.83M    DISK  00:00:16  2013-06-10 14:43:32
```

```
        BP Key: 1214   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130610T144316
         Piece Name: /DB_FRA/PHYDB/BACKUPSET/2013_06_10/O1_MF_
NNND1_TAG20130610T144316_8VBXDN30_.BKP
  List of Datafiles in backup set 1212
  File LV Type Ckp SCN    Ckp Time            Name
  ---- -- ---- ---------- ------------------- ----
  4  1  Incr 918398 2013-06-10 14:42:04 /DATABASE/PHYDB/EXAMPLE.DBF
```

List Recoverable 指令用于列出那些能够用于还原和恢复数据库的备份或映像副本，它只给出当前数据库 Incarnation 的、且状态为 Available 的备份结果。如果增量的备份没有对应的父备份（Level=0），则该备份不会包含在该指令的输出中。

在此列举 List 指令的一些其他用法：

```
list archivelog all;
list backup of archivelog all;
list backup of controlfile;
list backup of spfile;
list copy;
list copy of database;
list copy of tablespace userdata;
list copy of archivelog all;
list backup of archivelog sequence between 100 and 123;
list backup of archivelog from sequence 100 until sequence
123;
list backup of archivelog until time 'sysdate-1';
list backup of archivelog from time 'sysdate-7';
list backup completed after 'sysdate-1';
list backup completed between 'sysdate-7' and 'sysdate';
```

12.5.2 关于备份的报告

Report 指令用于生成关于备份结果的报告。Report 指令可以对备份结果或数据库进行适当的分析后给出我们需要的信息。

4 种基本的 Report 用法如下：

```
report need backup;
report obsolete;
report schema;
```

```
      report unrecoverable;

RMAN> report need backup;

RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
Report of files with less than 1 redundant backups
File #bkps Name
---- ----- -----------------------------------------
1    0     /DATABASE/PHYDB/SYSTEM.DBF
2    0     /DATABASE/PHYDB/SYSAUX.DBF
3    0     /DATABASE/PHYDB/UNDOTBS.DBF

RMAN> report obsolete;

RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
Report of obsolete backups and copies
Type                 Key    Completion Time    Filename/Handle
-------------------- ------ ------------------------ ---
Datafile Copy      1111   2013-06-10 09:53:51 .../O1_MF_
EXAMPLE_8VBDFHLY_.DBF
Backup Set           1182   2013-06-10 14:38:21
  Backup Piece         1185    2013-06-10 14:38:21 .../O1_MF_
NNNDF_TAG20130610T143821_8VBX2XKL_.BKP
Backup Set           1192   2013-06-10 14:39:00
  Backup Piece         1194    2013-06-10 14:39:00 .../O1_MF_
NNND0_TAG20130610T143844_8VBX444F_.BKP
Backup Set           1212   2013-06-10 14:43:32
  Backup Piece         1214    2013-06-10 14:43:32 .../O1_MF_
NNND1_TAG20130610T144316_8VBXDN30_.BKP

RMAN> report schema;

Report of database schema for database with db_unique_name
PHYDB

List of Permanent Datafiles
```

```
============================
File Size(MB) Tablespace   RB segs Datafile Name
---- -------- -------------------- ------- ------------------
1    181      SYSTEM       YES     /DATABASE/PHYDB/SYSTEM.DBF
2    96       SYSAUX       NO      /DATABASE/PHYDB/SYSAUX.DBF
3    44       UNDOTBS      YES     /DATABASE/PHYDB/UNDOTBS.DBF
4    10       EXAMPLE      NO      /DATABASE/PHYDB/EXAMPLE.DBF

List of Temporary Files
=======================
File Size(MB) Tablespace   Maxsize(MB) Tempfile Name
---- -------- -------------------- ----------- --------------
1    10      TEMPTBS      32767   /DATABASE/PHYDB/TEMPTBS.DBF

RMAN> report unrecoverable;

Report of files that need backup due to unrecoverable
operations
File Type of Backup Required Name
---- ----------------------- -----------------------------------
...
```

如果目标数据库中执行了某些不可恢复的操作，如使用 NOLOGGING 选项、UNRECOVERALBLE 选项创建数据库对象，使用直接路径加载一个表的数据等，都会导致对应的数据文件处于不可恢复状态，需要对其进行备份，否则，一旦损坏，就不能利用之前的备份及之后的数据库日志对其进行完全恢复。

12.5.3 备份的状态与交叉检验

对备份结果的交叉检验是指根据目标数据库控制文件或恢复目录中的备份记录检查对应的备份结果是否真实存在，若存在，则备份结果的状态标记为 Available（可用）；若不存在，则将备份结果的状态标记为 Expired（过期或不可用）。

```
RMAN> crosscheck backup;

allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=72 device type=DISK
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=
.../O1_MF_NNNDF_TAG20130609T102742_8V7T0Z0T_.BKP
```

```
RECID=6 STAMP=817640863
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=
.../O1_MF_NCSNF_TAG20130609T102742_8V7T1HDP_.BKP
RECID=7 STAMP=817640879
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=
.../O1_MF_NNNDF_TAG20130610T143821_8VBX2XKL_.BKP
RECID=8 STAMP=817742301
...
```

其他形式的 crosscheck 指令示例如下：

```
crosscheck backup of tablespace userdata;
crosscheck backup of datafile 3;
crosscheck backup completed after 'sysdate-1';
crosscheck backup completed between 'sysdate-7' and 'sysdate';
```

RMAN 根据备份保留策略，将不再需要的备份标记为 Obsolete（过时的）；通过交叉检查，将物理不存在的备份记录标记为 Expired（过期的）。可以通过 delete 指令删除备份结果及其备份记录。Delete 指令的基本形式如下：

```
delete backup ...;
delete copy ...;
delete obsolete ...;
delete expired ...;
```

另外，可以根据需要人工将某个备份或某些备份标记为 Unavailable，这样的状态表示 RMAN 的 Restore 指令不会利用该备份对数据库进行还原。

```
change ... available;
change ... unavailable;
change ... uncatalog;
change ... keep forever;
change ... keep until time '...';
```

12.5.4　在 RMAN 中注册备份

当在 RMAN 环境中使用 backup 指令执行某种备份时，RMAN 会自动将备份结果注册至目标数据库的控制文件或恢复目录。在某些情况下，可以使用 Catalog 指令手工注册手工备份结果、归档日志或额外的 RMAN 备份结果。

```
catalog archivelog 'logfilespec';
catalog backuppiece 'piecefilespec';
```

```
catalog datafilecopy 'copyfilespec';
catalog controlfilecopy 'copyfilespec';
catalog recovery area;
catalog start with 'dirspec';
```

第13章

RMAN与Data Guard

Data Guard通过备用数据库实现对主数据库的保护，RMAN是实现Oracle数据库备份与恢复的主要工具，两者的有机结合可以为用户实现完整的用于Oracle数据库故障恢复和高可用性的解决方案。本章结合Data Guard环境介绍RMAN在主备用数据库备份与恢复中的应用。

13.1 RMAN复制与备用数据库

前面的章节曾利用RMAN对主数据库的备份来创建物理备用数据库。逻辑备用数据库是由物理备用数据库转化而来的。RMAN与Data Guard高度集成，为了进一步简化创建物理备用数据库的过程，RMAN还提供了专门的复制（Duplicate）指令，本节对这一过程进行介绍。

13.1.1 复制的必要准备

本质上，物理备用数据库是主数据库物理存储的副本，因此，最初的物理备用数据库可以通过“复制”手段生成。RMAN的复制手段可以对源数据库（Source Database）生成如两种类型的数据库。

※ 复制数据库（A Duplicate Database）：源数据库的物理副本，但具有与源数据库不同的DBID。

※ 备用数据库（A Standby Database）：源数据库的物理副本，具有与源数据库相同的DBID。

使用Duplicate指令创建物理备用数据库的准备工作和常规的复制数据库一致，RMAN需要连接两个数据库（实例），一是目标数据库，即复制术语中的源数据库，二是辅助（Auxiliary）数据库实例，它是用来创建物理备用数据库对应的实例。

使用复制手段创建物理备用数据库与常规的复制数据库不同的是，需要事先为物理备用数据库创建备用控制文件。在主数据库中创建备用控制文件的两种方法如下：

```
SQL> alter database create standby controlfile as '...';
RMAN> backup current controlfile for standby;
```

13.1.2 为物理备用而复制

（1）准备初始化参数及其辅助实例。与手工创建物理备用数据库相同，如果复制产生的物理备用数据库与主数据库位于同一主机，则需要在辅助实例的初始化参数中设置 db_file_name_convert 和 log_file_name_convert 参数。

（2）在主数据库中创建备用控制文件（可选），代码如下：

```
rman target sys/internal@oradb

Recovery Manager: Release 11.2.0.1.0 - Production on Tue Jun
11 10:59:51 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORADB (DBID=3954687123)

RMAN> backup as copy current controlfile for standby
2> format '/database/phydb/CTLPHYDB.ORA';

Starting backup at 2013-06-11 14:46:12
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=5 device type=DISK
channel ORA_DISK_1: starting datafile copy
copying standby control file
output file name=/database/phydb/CTLPHYDB.ORA
tag=TAG20130611T144613 RECID=7 STAMP=817829174
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:01
Finished backup at 2013-06-11 14:46:14
```

（3）对主数据库执行全库备份，代码如下：

```
RMAN> connect target sys/internal@oradb

connected to target database: ORADB (DBID=3954687123)
```

```
RMAN> backup database;

Starting backup at 11-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=133 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=/database/oradb/SYSTEM.DBF
input datafile file number=00002 name=/database/oradb/SYSAUX.DBF
input datafile file number=00003 name=/database/oradb/UNDOTBS.
DBF
input datafile file number=00004 name=/database/oradb/EXAMPLE.
DBF
channel ORA_DISK_1: starting piece 1 at 11-JUN-13
channel ORA_DISK_1: finished piece 1 at 11-JUN-13
piece handle=.../O1_MF_NNNDF_TAG20130611T163936_8VFRL98V_.BKP
tag=TAG20130611T163936 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:15
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 11-JUN-13
channel ORA_DISK_1: finished piece 1 at 11-JUN-13
piece handle=.../O1_MF_NCSNF_TAG20130611T163936_8VFRLSGW_.BKP
tag=TAG20130611T163936 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 11-JUN-13
```

（4）完成数据库复制过程。需要注意的是，在生成备用数据库的复制过程中，实际上也是根据主数据库的备份生成备用数据库的物理存储的，包括备用控制文件，因此，步骤（2）是可选的。同时，生成的备用数据库的物理存储还包括备用数据库的联机日志文件和备用日志文件。注意，在下面的复制过程中使用的两个备份片就是上一步全库备份产生的备份结果。

```
RMAN> duplicate target database for standby;
```

```
Starting Duplicate Db at 11-JUN-13
using target database control file instead of recovery catalog
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=5 device type=DISK

contents of Memory Script:
{
   restore clone standby controlfile;
}
executing Memory Script

Starting restore at 11-JUN-13
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: reading from backup piece .../O1_MF_
NCSNF_TAG20130611T163936_8VFRLSGW_.BKP
channel ORA_AUX_DISK_1: piece handle=.../O1_MF_NCSNF_
TAG20130611T163936_8VFRLSGW_.BKP tag=TAG201306
11T163936
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time:
00:00:02
output file name=/database/phydb/CTLPHYDB.ORA
Finished restore at 11-JUN-13

contents of Memory Script:
{
   sql clone 'alter database mount standby database';
}
executing Memory Script

sql statement: alter database mount standby database

contents of Memory Script:
{
```

```
    set newname for tempfile  1 to "/database/phydb/TEMPTBS.
DBF";
   switch clone tempfile all;
   set newname for datafile  1 to "/database/phydb/SYSTEM.DBF";
   set newname for datafile  2 to "/database/phydb/SYSAUX.DBF";
    set newname for datafile  3 to "/database/phydb/UNDOTBS.
DBF";
    set newname for datafile  4 to "/database/phydb/EXAMPLE.
DBF";
   restore  clone database;
}
executing Memory Script

executing command: SET NEWNAME

renamed tempfile 1 to /database/phydb/TEMPTBS.DBF in control
file

executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME

Starting restore at 11-JUN-13
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/database/phydb/SYSTEM.DBF
channel ORA_AUX_DISK_1: restoring datafile 00002 to
/database/phydb/SYSAUX.DBF
channel ORA_AUX_DISK_1: restoring datafile 00003 to
/database/phydb/UNDOTBS.DBF
channel ORA_AUX_DISK_1: restoring datafile 00004 to
/database/phydb/EXAMPLE.DBF
channel ORA_AUX_DISK_1: reading from backup piece
```

```
.../O1_MF_NNNDF_TAG20130611T163936_8VFRL98V_.BKP
channel ORA_AUX_DISK_1: piece handle=
.../O1_MF_NNNDF_TAG20130611T163936_8VFRL98V_.BKP
tag=TAG20130611T163936
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time:
00:00:16
Finished restore at 11-JUN-13

contents of Memory Script:
{
   switch clone datafile all;
}
executing Memory Script

datafile 1 switched to datafile copy
input datafile copy RECID=9 STAMP=817836428 file name
=/database/phydb/SYSTEM.DBF
datafile 2 switched to datafile copy
input datafile copy RECID=10 STAMP=817836428 file name
=/database/phydb/SYSAUX.DBF
datafile 3 switched to datafile copy
input datafile copy RECID=11 STAMP=817836428 file name
=/database/phydb/UNDOTBS.DBF
datafile 4 switched to datafile copy
input datafile copy RECID=12 STAMP=817836428 file name
=/database/phydb/EXAMPLE.DBF
Finished Duplicate Db at 11-JUN-13
```

上面的复制过程是通过还原主数据库的备份来生成物理备用数据库的。如果直接从主数据库生成物理备用数据库，duplicate 指令需要使用 from active database 选项，即使用如下指令：

```
RMAN> duplicate target database for standby from active
database;
```

接下来，启动物理备用数据库的托管恢复即可。

```
SYS@PHYDB>alter database recover managed standby database
disconnect;
Database altered.
```

此时，备用数据库缺少临时表空间的数据文件，当该库以只读方式打开时，Oracle 会自动创建与主数据库对应的临时文件。

13.2　Data Guard 环境中的 RMAN 配置

在主备用数据库环境中，物理备用数据库不仅是主数据库的物理副本，同时，也与主数据库具有相同的 DBID 和 Incarnation，作为 RMAN 备份，物理备用数据库的备份与主数据库的备份是可以相互通用的。但要使得在一个数据库服务器上获得的备份能够在另外一个数据库服务器上还原，应该使用 RMAN 的恢复目录。因此，典型的 Data Guard 环境中的 RMAN 配置需要使用恢复目录数据库，通过恢复目录，RMAN 备份信息可以在物理备用数据库控制文件和主数据库控制文件之间同步。

13.2.1　唯一性标识 DB_UNIQUE_NAME

在 Data Guard 环境中，RMAN 通过 DB_UNIQUE_NAME 来区别主备用数据库，因此，每一个 DG 中的数据库 DB_UNIQUE_NAME 应该具有唯一性（从 11g 开始强制要求 DG 中的数据库必须具有唯一的 DB_UNIQUE_NAME）。

由于物理备用数据库与主数据库具有相同的 DBID 和 Incarnation，因此，物理备用数据库与主数据库都可以作为 RMAN 的目标数据库连接，但只有主数据库需要注册到恢复目录，物理备用数据库无须注册。

如果向恢复目录注册物理备用数据库，会获得如下错误信息：

```
RMAN> register database;

RMAN-00571: ===========================================
RMAN-00569: ========== ERROR MESSAGE STACK FOLLOWS ==========
RMAN-00571: ===========================================
RMAN-03002: failure of register command at 06/12/2013 08:06:57
RMAN-01005: Mounted control file type must be CURRENT to
register the database
```

至于分别从物理备用数据库或主数据库产生的备份结果，它们分别与对应数据库的 DB_UNIQUE_NAME 建立联系，也可以使用 change 指令更改备份结果与指定 DB_UNIQUE_NAME 的关联。

```
RMAN> list backup summary for db_unique_name oradb;

List of Backups for database with db_unique_name ORADB
===============
Key TY LV S Device Completion Time #Pieces #Copies Compressed
Tag
```

```
-----------------------------------------------------
506  B  F  A DISK 31-MAY-13  1 1  NO  TAG20130531T104322
507  B  F  A DISK 31-MAY-13  1 1  NO  TAG20130531T104322
1795 B  A  A DISK 10-JUN-13  1 1  NO  TAG20130610T205311

RMAN> change backup tag='TAG20130610T205311' for db_unique_
name ORADB reset db_unique_name to PHYDB;

change backup piece db_unique_name
backup piece handle=
.../O1_MF_ANNNN_TAG20130610T205311_8VCM1RFW_.BKP
RECID=5 STAMP=817764792
Changed 1 objects db_unique_name

RMAN> list backup summary for db_unique_name oradb;

List of Backups for database with db_unique_name ORADB
===============
Key       TY  LV  S  Device  Completion  Time  #Pieces  #Copies
Compressed Tag
-----------------------------------------------------
506  B  F  A DISK  31-MAY-13  1 1  NO  TAG20130531T104322
507  B  F  A DISK  31-MAY-13  1 1  NO  TAG20130531T104322

RMAN> list backup summary for db_unique_name phydb;

List of Backups for database with db_unique_name PHYDB
===============
Key       TY  LV  S  Device  Completion  Time  #Pieces  #Copies
Compressed Tag
-----------------------------------------------------
...
1795  B A A DISK   10-JUN-13  1 1  NO  TAG20130610T205311
```

在上面的 change 演示中，如果当前目标数据库是主库，则指令“change backup tag='...' for db_unique_name ORADB reset db_unique_name to PHYDB;”中的 for db_unique_name ORADB 可以省略。

13.2.2　为主备库配置 RMAN

在非 Data Guard 环境中，可以为每一个目标数据库设计独立的 RMAN 配置。但在 Data Guard 环境中，物理备用数据库的 RMAN 配置对主数据库具有一定的依赖性，即主备用数据库的某些配置必须保持一致（此类配置需要在主数据库中进行），当更改了主数据库的某些 RMAN 配置后，物理备用数据库的 RMAN 配置也会随之更改，如配置备份的保留策略等。

```
rman target sys/internal@oradb catalog rman/rman@catdb
connected to target database: ORADB (DBID=3954687123)
connected to recovery catalog database

RMAN> show all;

starting full resync of recovery catalog
full resync complete
RMAN configuration parameters for database with db_unique_name
ORADB are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
...

RMAN> configure retention policy to recovery window of 7 days;

new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> show all for db_unique_name PHYDB;

RMAN configuration parameters for database with db_unique_name
PHYDB are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
...
```

如果在物理备用数据库中修改备份的保留策略，则会出现如下错误：

```
rman target sys/internal@phydb catalog rman/rman@catdb

RMAN> configure retention policy to redundancy 2;
```

```
RMAN-00571: ===========================================
RMAN-00569: ========== ERROR MESSAGE STACK FOLLOWS ==========
RMAN-00571: ===========================================
RMAN-03002: failure of configure command at 06/12/2013 09:57:10
RMAN-05021: this configuration cannot be changed for a BACKUP
or STANDBY control file
```

1．为主数据库配置 RMAN

某些 RMAN 配置必须在主数据库中配置（即主备用数据库的配置必须保持一致），如备份的保留策略、每个主备用数据库 db_unique_name 对应的连接字符串（网络服务名）等。

※ 配置备份保留策略。

※ 为主数据库和物理备用数据库配置连接字符串，以便 RMAN 能够执行远程连接，用以及时 RMAN 同步配置信息和恢复目录。应该在主数据库中为主数据库和每个物理备用数据库配置连接字符串。

```
RMAN> configure db_unique_name 'ORADB' connect identifier
'oradb';

new RMAN configuration parameters:
CONFIGURE DB_UNIQUE_NAME 'ORADB' CONNECT IDENTIFIER  'oradb';
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete

RMAN> configure db_unique_name 'PHYDB' connect identifier
'phydb';

new RMAN configuration parameters:
CONFIGURE DB_UNIQUE_NAME 'PHYDB' CONNECT IDENTIFIER  'phydb';
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
```

※ 为主数据库配置归档日志的删除策略。

当确保主数据库的归档日志成功传输到备用数据库端后，可以考虑删除主数据库本地的归档日志，以便主数据库不会因为归档日志的存储空间问题而挂机。

```
RMAN> configure archivelog deletion policy to shipped to
standby;

new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO STANDBY;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
RMAN-08591: WARNING: invalid archived log deletion policy

RMAN> configure archivelog deletion policy to shipped to all
standby;

old RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO STANDBY;
new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO ALL
STANDBY;
new RMAN configuration parameters are successfully stored
starting full resync of recovery catalog
full resync complete
```

前面的参考配置中，当配置归档日志的删除策略为 shipped to standby 时，RMAN 警告 invalid archived log deletion policy，原因是该策略要求至少一个备用归档目标为强制归档目标（即设置归档属性 mandatory），以防止出现归档日志没有及时传输到备用端而本地归档日志已被删除的情况。通常，这个强制归档目标应该是指向一个物理备用数据库。为了进一步保障数据的可恢复性，主数据库归档日志的删除策略还可以配置为 applied to standby 或 applied to all standby，这样，主数据库的归档日志不仅要求传输到备用端，而且还需要等到执行 Redo Apply 以后才能删除。

2．为物理备用数据库配置 RMAN

对生产数据库构建了物理备用数据库后，不但可以实现对主数据库的保护，还可以从主数据库卸载只读操作，如统计报表等，而且还可以将日常的数据备份操作卸载到物理备用数据库上，这样可以极大地减轻主数据库上的可能负荷。

为此，可以为物理备用数据库执行如下 RMAN 配置。

※ 配置控制文件的自动备份、备份优化等。

```
RMAN> configure controlfile autobackup on;
RMAN> configure backup optimization on;
```

※ 配置默认的磁带通道，以便在物理备用数据库上借助于介质管理软件使用磁带备份。

```
RMAN> configure channel device type sbt parms '<channel_
parametes>';
```

※ 配置物理备用数据库上的归档日志删除策略。

若 Data Guard 环境中存在多个物理备用数据库，执行备份任务的物理备用数据库站点和不执行备份任务的站点应该有所区别，前者可以不应用任何删除策略，让归档日志的存留交给 Oracle 对快速恢复区的管理，也可以应用备份后删除策略，即归档日志在经过指定备份次数（下面示例指令中的 n）后再删除；后者在归档日志成功应用后即可删除。

```
RMAN> configure archivelog deletion policy to none;
RMAN> configure archivelog deletion policy to

      backed up n times to disk;
RMAN> configure archivelog deletion policy to

      backed up n times to sbt;
RMAN> configure archivelog deletion policy to applied on
standby;
```

13.3 卸载备份至备用数据库

13.3.1 RMAN 备份的数据库相关性

当我们为主备用数据库分别对 RMAN 进行配置后，接下来就可以将物理备用数据库作为 RMAN 的目标数据库进行连接，将原先在主数据库上的备份操作完全转移至物理备用数据库。由于物理备用数据库本质上是主数据库的物理副本，因此，不论是在主数据库上的备份还是在物理备用数据库上的备份，在还原和恢复数据库时，两者是通用的、可以互换的，但需要注意的是，在 RMAN 中查看和管理备份结果时，它们还是和 DB_UNIQUE_NAME 相关联的。从下面的操作中可以清楚地看出这种关联性。

```
rman target sys/internal@phydb catalog rman/rman@catdb
Recovery Manager: Release 11.2.0.1.0 - Production on Fri Jun
14 09:23:46 2013
```

```
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORADB (DBID=3954687123, not
open)
connected to recovery catalog database

RMAN> list db_unique_name of database;

List of Databases
DB Key   DB Name   DB ID   Database Role    Db_unique_name
------------------------------------------------------------
2        ORADB     3954687123   PRIMARY     ORADB
2        ORADB     3954687123   STANDBY     PHYDB

RMAN> show all for db_unique_name oradb;
RMAN configuration parameters for database with db_unique_name
ORADB are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
...
CONFIGURE DB_UNIQUE_NAME 'ORADB' CONNECT IDENTIFIER  'oradb';
CONFIGURE DB_UNIQUE_NAME 'PHYDB' CONNECT IDENTIFIER  'phydb';
CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO ALL
STANDBY;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '.../SNCFORADB.ORA';

RMAN> show all for db_unique_name phydb;
RMAN configuration parameters for database with db_unique_name
PHYDB are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON;
...
CONFIGURE DB_UNIQUE_NAME 'ORADB' CONNECT IDENTIFIER  'oradb';
```

```
CONFIGURE DB_UNIQUE_NAME 'PHYDB' CONNECT IDENTIFIER  'phydb';
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '.../DATABASE/
SNCFPHYDB.ORA';

RMAN> resync catalog from db_unique_name phydb;

RMAN> resync catalog from db_unique_name oradb;

RMAN> list backup summary for db_unique_name phydb;

List of Backups for database with db_unique_name PHYDB
===============
Key       TY LV S Device Completion Time #Pieces #Copies
Compressed Tag
------ -- -- - ---------- -------------- ------- ------- --- ---
2150 B F A DISK   14-JUN-13   1 1  NO   TAG20130614T091521

RMAN> list backupset 2150;

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------- ---- -- ---------- ----------- ------------ -----------
2150    Full    9.83M      DISK        00:00:00     14-JUN-13
        BP Key: 2151   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130614T091521
        Piece Name: .../O1_MF_S_818067935_8VNVO9KG_.BKP
  Standby Control File Included: Ckp SCN: 1028235 Ckp time:
14-JUN-13
  SPFILE Included: Modification time: 14-JUN-13

RMAN> list backup summary for db_unique_name oradb;

List of Backups for database with db_unique_name ORADB
===============
```

```
Key       TY LV S Device Completion Time #Pieces #Copies
Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ----
1998 B  F  A DISK        11-JUN-13           1 1    NO
TAG20130611T163936
1999 B  F  A DISK 11-JUN-13           1 1    NO
TAG20130611T163936

RMAN> list backupset 1998;

RMAN-00571: ===========================================
RMAN-00569: ========== ERROR MESSAGE STACK FOLLOWS =========
RMAN-00571: ===========================================
RMAN-03002: failure of list command at 06/14/2013 09:49:19
RMAN-06160: no backup pieces found for backup set key: 1998

RMAN> list backupset 1998 for db_unique_name oradb;

List of Backup Set for database with db_unique_name ORADB
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------- ---- -- ---------- ----------- ------------ -----------
1998    Full    254.03M    DISK        00:00:11     11-JUN-13
        BP Key: 2000   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130611T163936
               Piece Name: ...\O1_MF_NNNDF_
TAG20130611T163936_8VFRL98V_.BKP
  List of Datafiles in backup set 1998
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 956590     11-JUN-13 /database/phydb/SYSTEM.DBF
  2       Full 956590     11-JUN-13 /database/phydb/SYSAUX.DBF
  3       Full 956590    11-JUN-13 /database/phydb/UNDOTBS.DBF
  4       Full 956590    11-JUN-13 /database/phydb/EXAMPLE.DBF

rman target sys/internal@oradb catalog rman/rman@catdb
```

```
connected to target database: ORADB (DBID=3954687123)
connected to recovery catalog database

RMAN> list backupset 1998;

List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------- ---- -- ---------- ----------- ------------ -----------
1998    Full    254.03M    DISK        00:00:11     11-JUN-13
        BP Key: 2000   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130611T163936
               Piece Name: ...\O1_MF_NNNDF_
TAG20130611T163936_8VFRL98V_.BKP
  List of Datafiles in backup set 1998
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 956590     11-JUN-13 /database/oradb/SYSTEM.DBF
  2       Full 956590     11-JUN-13 /database/oradb/SYSAUX.DBF
  3       Full 956590    11-JUN-13 /database/oradb/UNDOTBS.DBF
  4       Full 956590    11-JUN-13 /database/oradb/EXAMPLE.DBF
```

从上述对于编号为 1998 的备份集的查看结果可以看出，同样一份备份，备份片为 O1_MF_NNNDF_TAG20130611T163936_8VFRL98V_.BKP，当 RMAN 先后连接至不同的目标数据库（物理备用数据库、主数据库）时，给出的备份集中的内容（物理文件的路径）是不同的，这实际上反映的是利用该备份集还原数据库时生成的数据库文件的存储路径。由此可见，同样一份备份集（不论是从主数据库备份还是从物理备用数据库备份），既可以用于还原主数据库，也可以用于还原物理备用数据库。

13.3.2 使用备用端备份恢复主数据库

本节以前面介绍的 Data Guard 环境为基础，演示如何使用物理备用数据库的备份来恢复主数据库，其中，主数据库的标识为（DB_NAME=ORADB，DB_UNIQUE_NAME=ORADB），物理备用数据库的标识为（DB_NAME=ORADB，DB_UNIQUE_NAME=PHYDB）。

为了说明在存在物理备用数据库的情况下，备份可以完全脱离数据库，现将基

于主数据库的备份完全删除，代码如下：

```
RMAN> delete backupset ... for db_unique_name oradb;
...
RMAN> list backup summary for db_unique_name ORADB;
specification does not match any backup in the repository
```

将物理备用数据库作为目标数据库连接，同时，连接恢复目录数据库，执行对物理备用数据库的备份，代码如下：

```
RMAN> backup database plus archivelog;

Starting backup at 14-JUN-13
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
...
input archived log thread=1 sequence=328 RECID=14
STAMP=818067937
input archived log thread=1 sequence=329 RECID=16
STAMP=818067940
input archived log thread=1 sequence=330 RECID=17
STAMP=818074245
input archived log thread=1 sequence=331 RECID=18
STAMP=818092830
channel ORA_DISK_1: starting piece 1 at 14-JUN-13
channel ORA_DISK_1: finished piece 1 at 14-JUN-13
piece handle=.../O1_MF_ANNNN_TAG20130614T164447_8VOP011J_.BKP
tag=TAG20130614T164447 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:03
Finished backup at 14-JUN-13

Starting backup at 14-JUN-13
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001 name=/database/phydb/SYSTEM.DBF
input datafile file number=00002 name=/database/phydb/SYSAUX.DBF
```

```
input datafile file number=00003 name=/database/phydb/UNDOTBS.
DBF
input datafile file number=00004 name=/database/phydb/EXAMPLE.
DBF
channel ORA_DISK_1: starting piece 1 at 14-JUN-13
channel ORA_DISK_1: finished piece 1 at 14-JUN-13
piece handle=.../O1_MF_NNNDF_TAG20130614T164452_8VOP04T6_.BKP
tag=TAG20130614T164452 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:15
Finished backup at 14-JUN-13

Starting backup at 14-JUN-13
using channel ORA_DISK_1
specification does not match any archived log in the repository
backup cancelled because there are no files to backup
Finished backup at 14-JUN-13

Starting Control File and SPFILE Autobackup at 14-JUN-13
piece handle=.../O1_MF_S_818092828_8VOP0OYQ_.BKP comment=NONE
Finished Control File and SPFILE Autobackup at 14-JUN-13

RMAN> list backup summary for db_unique_name phydb;

List of Backups for database with db_unique_name PHYDB
===============
Key       TY LV S Device Completion Time #Pieces #Copies
Compressed Tag
-------------------------------------------------------
2409 B  A  A  DISK  14-JUN-13  1  1  NO     TAG20130614T164447
2410 B  F  A  DISK  14-JUN-13  1  1  NO     TAG20130614T164452
2446 B  F  A  DISK  14-JUN-13  1  1  NO     TAG20130614T164509
```

关闭主数据库，人为损坏主数据库的数据文件，模拟主数据库故障，代码如下：

```
SYS@ORADB>select name from v$datafile;

NAME
--------------------------------------
/database/oradb/SYSTEM.DBF
```

```
/database/oradb/SYSAUX.DBF
/database/oradb/UNDOTBS.DBF
/database/oradb/EXAMPLE.DBF
```

这里我们人为损坏数据文件 /database/oradb/EXAMPLE.DBF，此时，主数据库的故障如下：

```
SYS@ORADB>shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@ORADB>startup
ORACLE instance started.

Total System Global Area  267227136 bytes
Fixed Size                  2174888 bytes
Variable Size             121634904 bytes
Database Buffers          138412032 bytes
Redo Buffers                5005312 bytes
Database mounted.
ORA-01157: cannot identify/lock data file 4 - see DBWR trace
file
ORA-01110: data file 4: '/database/oradb/EXAMPLE.DBF'
```

下面使用物理备用数据库的备份还原并恢复主数据库的损坏部分，之后，主数据库正常打开运行。

```
$rman target sys/internal@oradb catalog rman/rman@catdb

Recovery Manager: Release 11.2.0.1.0 - Production on Fri Jun
14 17:02:50 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORADB (DBID=3954687123, not
open)
connected to recovery catalog database

RMAN> list backup summary;
```

```
specification does not match any backup in the repository

RMAN> list backup summary for db_unique_name phydb;

List of Backups for database with db_unique_name PHYDB
===============
Key       TY LV S Device Completion Time #Pieces #Copies
Compressed Tag
-------------------------------------------------------
2409 B  A  A  DISK  14-JUN-13  1  1  NO     TAG20130614T164447
2410 B  F  A  DISK  14-JUN-13  1  1  NO     TAG20130614T164452
2446 B  F  A  DISK  14-JUN-13  1  1  NO     TAG20130614T164509

RMAN> list backupset 2410;

RMAN-00571: ===========================================
RMAN-00569: ========== ERROR MESSAGE STACK FOLLOWS ===========
RMAN-00571: ===========================================
RMAN-03002: failure of list command at 06/14/2013 17:05:40
RMAN-06160: no backup pieces found for backup set key: 2410

RMAN> list backupset 2410 for db_unique_name phydb;

List of Backup Set for database with db_unique_name PHYDB
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------- ---- -- ---------- ----------- ------------ -----------
2410    Full    268.62M    DISK        00:00:13     14-JUN-13
        BP Key: 2412   Status: AVAILABLE  Compressed: NO  Tag:
TAG20130614T164452
           Piece Name: .../O1_MF_NNNDF_TAG20130614T164452
_8VOP04T6_.BKP
  List of Datafiles in backup set 2410
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 1043039    14-JUN-13 /database/oradb/SYSTEM.DBF
  2       Full 1043039    14-JUN-13 /database/oradb/SYSAUX.DBF
```

```
  3       Full 1043039   14-JUN-13 /database/oradb/UNDOTBS.DBF
  4       Full 1043039   14-JUN-13 /database/oradb/EXAMPLE.DBF

RMAN> catalog backuppiece
'.../O1_MF_NNNDF_TAG20130614T164452_8VOP04T6_.BKP';

cataloged backup piece
backup piece handle=
.../O1_MF_NNNDF_TAG20130614T164452_8VOP04T6_.BKP
RECID=10 STAMP=818096885

RMAN> list backup summary;

List of Backups
===============
Key       TY LV S Device Completion Time #Pieces #Copies
Compressed Tag
------- -- -- - ----------- ------------- ------- ------- -----
2410  B  F  A DISK  14-JUN-13  1  1  NO    TAG20130614T164452

RMAN> restore datafile 4 from tag='TAG20130614T164452';

Starting restore at 14-JUN-13
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=5 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00004 to
/database/oradb/EXAMPLE.DBF
channel ORA_DISK_1: reading from backup piece
.../O1_MF_NNNDF_TAG20130614T164452_8VOP04T6_.BKP
channel ORA_DISK_1: piece handle=
.../O1_MF_NNNDF_TAG20130614T164452_8VOP04T6_.BKP
tag=TAG20130614T164452
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 14-JUN-13
```

```
RMAN> recover datafile 4;
Starting recover at 14-JUN-13
using channel ORA_DISK_1
starting media recovery
media recovery complete, elapsed time: 00:00:01
Finished recover at 14-JUN-13

RMAN> alter database open;
database opened

SYS@ORADB>select instance_name,status from v$instance;

INSTANCE_NAME      STATUS
---------------- ------------
oradb              OPEN
```

13.3.3 使用备用端数据文件恢复主数据库

假设这样一种情形：主数据库的数据文件遭到损坏，但主数据库和物理备用数据库都没有备份可以利用。此时，如何恢复损坏的主数据库中的数据文件呢？

由于物理备用数据库是主数据库的物理副本，本质上，整个物理备用数据库就是主数据库的备份，从 RMAN 的角度来看，某时刻的物理备用数据库就是某时刻主数据库的映像复制（Image Copy），而且该映像复制还是动态的。上述情形可以直接使用物理备用数据库的数据文件来还原主数据库损坏的数据文件。过程说明如下。

（1）主备用数据库都无备份可以利用。

```
RMAN> list backup summary for db_unique_name phydb;
specification does not match any backup in the repository

RMAN> list backup summary for db_unique_name oradb;
specification does not match any backup in the repository
```

（2）主数据库的数据文件损坏。

```
SYS@ORADB>startup
ORACLE instance started.

Total System Global Area  267227136 bytes
Fixed Size                  2174888 bytes
Variable Size             138412120 bytes
```

```
Database Buffers          121634816 bytes
Redo Buffers                5005312 bytes
Database mounted.
ORA-01157: cannot identify/lock data file 4 - see DBWR trace
file
ORA-01110: data file 4: '/database/oradb/EXAMPLE.DBF'
```

（3）利用物理备用数据库的数据文件还原主数据库损坏的数据文件。

在 RMAN 中将物理备用数据库作为目标数据库，将主数据库作为辅助数据库连接，同时连接恢复目录数据库。

```
$rman target sys/internal@phydb auxiliary sys/internal@oradb
catalog rman/rman@catdb

Recovery Manager: Release 11.2.0.1.0 - Production on Sat Jun
15 15:18:37 2013

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORADB (DBID=3954687123)
connected to recovery catalog database
connected to auxiliary database: ORADB (DBID=3954687123, not
open)

RMAN> backup as copy datafile 4
auxiliary format '/database/oradb/example_from_phydb.dbf';

Starting backup at 15-JUN-13
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=8 device type=DISK
channel ORA_DISK_1: starting datafile copy
input datafile file number=00004 name=/DATABASE/PHYDB/EXAMPLE.
DBF
output file name=/DATABASE/ORADB/EXAMPLE_FROM_PHYDB.DBF
tag=TAG20130615T152132
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:01
Finished backup at 15-JUN-13
```

（4）在主数据库中注册获得的文件复制（可选）。

```
$rman target sys/internal@oradb catalog rman/rman@catdb

connected to target database: ORADB (DBID=3954687123, not
open)
connected to recovery catalog database

RMAN> catalog datafilecopy
'/database/oradb/example_from_phydb.dbf';

cataloged datafile copy
datafile copy file name=/database/oradb/example_from_phydb.dbf
RECID=12 STAMP=818231096
```

（5）利用已注册的数据文件复制（Image Copy）还原损坏的数据文件（可选）。

```
RMAN> restore datafile 4;

Starting restore at 16-JUN-13
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=5 device type=DISK

channel ORA_DISK_1: restoring datafile 00004
input datafile copy RECID=12 STAMP=818231096 file name=
/DATABASE/ORADB/EXAMPLE_FROM_PHYDB.DBF
destination for restore of datafile 00004:
/DATABASE/ORADB/EXAMPLE.DBF
channel ORA_DISK_1: copied datafile copy of datafile 00004
output file name=/DATABASE/ORADB/EXAMPLE.DBF RECID=0 STAMP=0
Finished restore at 16-JUN-13
```

或者直接利用从物理备用数据库获得的数据文件复制：

```
RMAN> run {
set newname for datafile 4 to
"/DATABASE/ORADB/EXAMPLE_FROM_PHYDB.DBF";
switch datafile 4; }

executing command: SET NEWNAME
```

```
datafile 4 switched to datafile copy
input datafile copy RECID=12 STAMP=818231096 file
name=/DATABASE/ORADB/EXAMPLE_FROM_PHYDB.DBF
starting full resync of recovery catalog
full resync complete
```

（6）恢复损坏的数据文件，打开主数据库。

```
SYS@ORADB>recover datafile 4;
ORA-00279: change 1060086 generated at 06/14/2013 22:01:23
needed for thread 1
ORA-00289: suggestion : .../O1_MF_1_338_8VR3LH80_.ARC
ORA-00280: change 1060086 for thread 1 is in sequence #338

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
auto
ORA-00279: change 1080088 generated at 06/15/2013 14:48:46
needed for thread 1
ORA-00289: suggestion : .../O1_MF_1_339_8VR3LLNK_.ARC
ORA-00280: change 1080088 for thread 1 is in sequence #339

ORA-00279: change 1080110 generated at 06/15/2013 14:48:50
needed for thread 1
ORA-00289: suggestion : .../O1_MF_1_340_8VR4PBKO_.ARC
ORA-00280: change 1080110 for thread 1 is in sequence #340

Log applied.
Media recovery complete.
SYS@ORADB>alter database open;
Database altered.
```

第14章

闪回数据库与 Data Guard

Oracle 先后推出了一些列闪回功能（Flashback），实际上它是一个系列功能，称为闪回家族（Flashback Family），包括闪回查询（Flashback Query）、闪回表（Flashback Table）、闪回版本查询（Flashback Version Query）、闪回事务查询（Flashback Transaction Query）、闪回删除（Flashback Drop）、闪回数据库（Flashback Database）等。虽然都称为闪回，但这一系列功能内部的实现机制则不尽相同，如闪回查询依赖于回滚信息（Undo Data）、闪回删除依赖于回收站（Recyclebin）、闪回数据库依赖于闪回日志（Flashback Log）。

闪回数据库的功能与 Data Guard 有着较密切的关系，本章将介绍闪回数据库及其在数据库恢复中的应用，包括闪回数据库在 Data Guard 中的应用。数据库的闪回特性，在保持 Data Guard 环境下的主备用数据库同步、故障恢复、角色转换等方面都起到了重要作用，两者的有机结合，可以使得 Data Guard 的应用更灵活。

14.1 数据库闪回功能

一般来说，数据库的状态随着时间的变化而变化，可以说，数据库的状态是时间的函数。Oracle 数据库的闪回功能使得数据库的状态在一定范围内随着时间向前回绕，就像磁带一样，可以“倒带”以追溯数据库在过去某个时间点的特定状态。

与 Data Guard 相比，备用数据库是从空间的角度实现数据库的远程数据保护的，而数据库闪回是从时间的角度实现数据库的远程数据保护的。

14.1.1 闪回日志与数据库闪回

Oracle 闪回数据库特性在物理级别上提供了一个比传统的基于时间点数据库恢复（Database Point-in-time Recovery，即 DBPITR）更有效的数据保护，这是通过闪回日志的机制开发的新功能。启动闪回数据库功能后，实例中会增加一个称为 RVWR（Recovery Writer）的后台进程，它会按照数据修改的状况不定期

地将所有更改的数据块映像（Image）写入一个磁盘文件，这就是闪回日志。

闪回日志在时间上是离散的，RVWR 的每一次记录都会产生一个闪回日志文件。与之相对应，数据库的事务日志是连续记录的，它会记录数据库每一个事务产生的数据变更。因此，在具体实现数据库闪回功能时，单独使用闪回日志是不行的，往往还要部分地依赖于事务日志。要将数据库回退到过去的某个特定时刻（闪回点），首先，Oracle 反向应用一系列闪回日志将数据库快速回退到距离这个时刻最近的闪回日志记录点（该记录点一定位于闪回点之前）；然后，在此基础上再应用事务日志将数据库前滚到用户指定的闪回点。因此，在实际应用中，闪回数据库的操作仅依靠闪回日志是不行的，往往还要同时依赖闪回日志和部分事务日志。

利用数据库闪回技术将数据库回退到过去某个特定的时间点，要比传统的基于备份和介质恢复技术将数据库恢复到该时间点快捷得多，这也是利用闪回日志实现数据库闪回的明显优势。

Oracle 通过后台进程 RVWR 按一定的时间间隔自动记录闪回日志，并且闪回日志只能记录到快速闪回恢复区（Flash Recovery Area，FRA）。闪回日志在 FRA 的保存时限受快速恢复区的空间配置和初始化参数 db_flashback_retention_target 的控制。

14.1.2　还原点和闪回窗口

由于闪回日志是保存在闪回恢复区的，而闪回恢复区的空间是有限的，这样，当闪回恢复区出现空间压力时，Oracle 就会考虑重用闪回日志文件占用的空间，覆盖删除最老的闪回日志，这种情况和 Oracle 重用归档日志占用的 FRA 空间、重用回滚信息占用的回滚表空间类似。回滚信息在回滚表空间的保留目标由参数 undo_retention 控制，类似地，闪回日志的保留目标由参数 db_flashback_retention_target 控制。

在没有空间压力的情形下，数据库能够闪回的最早时间点由保存在 FRA 中的最早的闪回日志决定，这和表的闪回查询情形类似。数据库的闪回窗口是指由当前时间点到能够闪回的最早时间点的时间范围（或 SCN 范围）。一般来说，数据库的闪回窗口由存储在 FRA 中的最早的闪回日志决定，但某些特殊的操作会破坏这种情形，如删除表空间、收缩数据文件（Shrink a Datafile）等，数据库的闪回时限不能跨越这些操作。

值得注意的是，与回滚信息的保留控制一样，db_flashback_retention_target 参数只是设置闪回日志的保留目标，当 FRA 出现空间压力时，这个保存时限可能会被突破。因此，在实际应用中，一个数据库的闪回窗口既可能大于参数 db_flashback_retention_target 设置的时限范围，也可能小于这个时限范围，这取决于 FRA 中的空间使用状况。

Oracle 数据库的还原点（Restore Point）是和数据闪回相关联的数据保

护机制。一个还原点是对应于数据库特定 SCN 或特定时刻的标签，和我们日常使用的书签一样，它仅代表闪回窗口中的某个特定位置。根据还原点的保留限制，还原点又分为常规还原点（Normal Restore Point）和保证还原点（Guaranteed Restore Point）。常规还原点的存留受 FRA 中闪回日志保留策略的限制，当没有足够闪回日志支持时，这个还原点也是不能被闪回的。因此，在数据库的闪回窗口范围内，特定的还原点、特定的时间点、特定的 SCN 具有相同的还原意义，只是还原点具有一个有意义的名称标识。

要确保数据库可以还原到过去的某个时刻，应该创建保证还原点，该还原点可以不受 FRA 中闪回日志保留策略的限制，是一个有保障的闪回时刻点，直到该还原点被人工删除。保证还原点特别有意义之处还在于：即使取消了数据库的闪回功能，数据库也可以确保在之后的某个时刻将数据库回退到之前的特定保证还原点。不同的是，如果既创建了保证还原点，又启动了数据库闪回功能，则可以将数据库闪回到保证还原点至当前时刻的任意时间点，而不是仅仅只能闪回到特定的保证还原点。

保证还原点的一个重要用途在于，当要对数据库执行某些有一定风险的操作前（如大批量的数据更新、数据库升级、安装补丁等），可以创建一个保证还原点，从数据恢复的等效意义上，保证还原点相当于保留了一份操作前的数据库存储快照（Storage Snapshot）。在 Data Guard 环境中的主数据库、物理备用数据库上，都可以根据需要在适当时刻创建保证还原点，以便之后撤销有风险的操作后果，将其还原到之前的常规状态，保证主备用数据库的正常运行。

保证还原点和通常意义上的数据库闪回功能既可以单独使用，也可以联合使用，两种方式的区别在于背后闪回日志的记录有所不同。

1．创建保证还原点但没有启动闪回日志

这种情形下，保证还原点之后的数据块的第一次数据变更之前的数据映像（Image）会被记入闪回日志，以便 Oracle 日后能够利用此映像将数据块还原到保证还原点。数据块后续的数据变更并不会记入闪回日志，这样可以显著节约闪回日志的存储空间。

优点：节约闪回日志的存储空间。

缺点：只能闪回到保证还原点，不能闪回到其他时刻。

2．创建保证还原点同时启动闪回日志

这种情形下，闪回日志被正常有规律地记录。保证还原点影响其后闪回日志的保留策略，凡是在保证还原点之后的闪回日志及其相关事务日志都需要保留，当 FRA 出现空间压力时，也不受保留策略 db_flashback_retention_target 参数的影响。

优点：可以将数据库闪回到保证还原点至当前时刻的任一时刻点。

缺点：对 FRA 存储空间的要求会增大。即使 FRA 出现空间压力时，保证还原点需要的闪回日志和事务日志也不能被删除。此时，特别需要保持对 FRA 使用空间的监控。

14.2　配置数据库闪回和还原点

14.2.1　闪回和保证还原点的条件

要启动闪回数据库功能，需要如下基本条件：

（1）数据库必须处于归档模式运行，因为闪回操作需要必要的事务日志的参与。

（2）必须为数据库配置快速恢复区（Flash/Fast Recovery Area，FRA），因为数据库的闪回日志只能存储在 FRA 中。

（3）RAC 集群环境下，FRA 必须存储于集群文件系统或 ASM 上。

使用常规的还原点对数据库没有特殊要求，保证还原点则不同。由前面的介绍可知，保证还原点功能和数据库闪回功能可以相对独立使用。要在数据库中创建保证还原点，数据库需要满足如下条件：

（1）初始化参数 Compatible 的值至少设置为 10.2 及其以后的版本。

（2）数据库同样需要运行在归档模式。闪回到保证还原点操作时，Oracle 也需要用到还原点附件的事务日志。

（3）必须配置闪回恢复区，因为保证还原点之后，Oracle 同样需要记录闪回日志，即使闪回日志功能没有启动。

（4）当数据库闪回功能没有启动时，第一次创建保证还原点数据库必须处于 mount 状态，内在的原因是第一次创建保证还原点时需要修改数据库的闪回状态（FLASHBACK_ON 的状态介于 OFF 和 ON 之间，为 Guaranteed Restore Point）。

14.2.2　启动数据库闪回功能

本节介绍如何启动数据库的闪回功能，以及如何创建还原点、保证还原点。

要启动数据库闪回功能，除了要确保数据库处于归档方式运行外，还需要配置足够大小的闪回恢复区。对于那些事务处理非常频繁的数据库来说，具有足够大小的 FRA 尤其重要，因为频繁的数据修改可以导致闪回日志快速积累。

（1）确保数据库处于归档模式运行。

```
SYS@ORADB>archive log list;
```

```
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      359
Next log sequence to archive    360
Current log sequence            360
```

（2）设置闪回日志的保留策略参数（可选）。参数 db_flashback_retention_target 以分钟为单位设置在 FRA 中闪回日志的保存目标。下面的示例设置闪回日志的保存目标为 24 小时。

```
SYS@ORADB>alter system set db_flashback_retention_target=1440
sid='*' scope=both;

System altered.
```

（3）可以在数据库处于 mount 状态或 open 状态下调整数据库的闪回状态。若数据库处于 mount 状态，则需要确保数据库上次的关闭是正常关闭。

```
SYS@ORADB>alter database flashback on;

Database altered.

SYS@ORADB>select flashback_on from v$database;

FLASHBACK_ON
------------------
YES
```

（4）关闭某些表空间的闪回日志记录（可选）。默认情况下，启动数据库的闪回功能，Oracle 会启动所有永久（Permanent）表空间的闪回日志记录。如果某些表空间不需要闪回功能，可以关闭这个表空间的闪回日志记录，以减少闪回日志的记录量。

```
SYS@ORADB>alter tablespace example flashback off;

Tablespace altered.
```

这里需要注意的是，如果要重新启动某个表空间的闪回功能，数据库则需要在 mount 状态下执行此修改。

```
SYS@ORADB>alter tablespace example flashback on;
alter tablespace example flashback on
```

```
*
ERROR at line 1:
ORA-01126: database must be mounted in this instance and not
open in any instance
```

另外，在某个表空间或某些表空间闪回功能关闭的情形下，在执行数据库闪回功能前，即执行 flashback database 指令前，需要显式地将这些表空间或表空间对应的数据文件置于 offline 状态。

成功启动了数据库闪回功能后，可以通过数据字典视图清楚地查看当前数据库的闪回窗口，以及系列闪回日志文件。

```
SYS@ORADB>desc v$flashback_database_log
 Name                                      Null?    Type
 ----------------------------------------------------------------
 OLDEST_FLASHBACK_SCN                               NUMBER
 OLDEST_FLASHBACK_TIME                              DATE
 RETENTION_TARGET                                   NUMBER
 FLASHBACK_SIZE                                     NUMBER
 ESTIMATED_FLASHBACK_SIZE                           NUMBER
```

其中，FLASHBACK_SIZE 指示当前闪回日志的大小，ESTIMATED_FLASHBACK_SIZE 是 Oracle 根据当前闪回日志保留目标 db_flashback_retention_target 估计的闪回日志的存储空间需求。

```
SYS@ORADB>alter session set nls_date_format='YYYY-MM-DD
HH24:MI:SS';

Session altered.

SYS@ORADB>select oldest_flashback_time,retention_target,
  2  (sysdate-oldest_flashback_time)*24*60 flashback_window
  3  from v$flashback_database_log;

OLDEST_FLASHBACK_TI RETENTION_TARGET FLASHBACK_WINDOW
------------------- ---------------- ----------------
2013-06-17 15:00:34             1440          1589.95

SYS@ORADB>select * from v$flashback_database_logfile;

NAME   LOG# THREAD#  SEQUENCE# BYTES FIRST_CHANGE# FIRST_TIME
```

```
------------------------------------------------------------
.../O1_MF_8VXF0KT2_.FLB  1  1  1  8192000  1140133  2013-06-17
15:00:34
...
```

创建数据库的常规还原点或保证还原点的语法如下：

```
create restore point restore_point_name
[guarantee flashback database];
```

指定 guarantee flashback database 选项，指示创建的还原点为保证还原点，否则，为常规还原点。通过视图 v$restore_point 可以查看已创建的所有还原点的信息。在 RMAN 中还可以使用 list restore point 指令查看目标数据库的还原点信息。

```
SYS@ORADB>desc v$restore_point;
 Name                           Null?    Type
 ------------------------------ -------- -------------
 SCN                                     NUMBER
 DATABASE_INCARNATION#                   NUMBER
 GUARANTEE_FLASHBACK_DATABASE            VARCHAR2(3)
 STORAGE_SIZE                            NUMBER
 TIME                                    TIMESTAMP(9)
 RESTORE_POINT_TIME                      TIMESTAMP(9)
 PRESERVED                               VARCHAR2(3)
 NAME                                    VARCHAR2(128)
```

注意，一旦创建了保证还原点，其后产生的闪回日志必须保证物理地存在。目前，数据库的闪回日志不能由 RMAN 对其执行备份，闪回日志只能存储在 FRA 中，并且只能由 Oracle 根据保留策略自动维护。如果由于某种原因人为地删除或丢失了保证还原点需要的闪回日志，那么，数据库是无法打开的，在打开时产生类似如下错误：

```
ORA-38760: This database instance failed to turn on flashback
database.
```

当出现上述错误时，即使按照错误提示将数据库的闪回功能关闭，也不能排除故障。解决的方法是查询控制文件中记录的保证还原点信息，删除相关的保证还原点，然后才能正常打开数据库。

14.3　闪回恢复区的维护

数据库的闪回日志只能存储在闪回恢复区（Flash/Fast Recovery Area），但闪回恢复区不仅仅用于存储闪回日志，它是专门用来存储与数据库恢复有关的所有文件的集中区域，包括默认的数据库备份（备份片、映像复制）、控制文件和联机日志的镜像副本、本地归档日志、外部归档日志（Foreign Archived Log，此类日志用于逻辑备用数据库），以及这里讨论的闪回日志等。

闪回恢复区的内容是由 Oracle 维护的，但在某些情况下还是需要 DBA 手工管理的。

14.3.1　闪回恢复区的删除规则

Oracle 的闪回恢复区的存储内容根据存在的时限分为永久的（Permanent）和暂时的（Transient）两大类。前者一旦创建，始终存在于闪回恢复区，除非 DBA 通过管理指令手工删除它；后者只是临时性地存储在闪回恢复区，当出现空间压力时，Oracle 会根据 DBA 事先设置的删除规则删除相应的内容，释放其占用的存储空间。

闪回恢复区存储的内容有：控制文件的联机镜像（Multiplexed copies of the current control file）、在线重做日志文件（Online redo log files）、归档日志文件（Archived redo log files）、外部归档日志文件（Foreign archived redo log files）、数据文件的映像复制（Image coppies of datafiles）、控制文件的映像复制（Image copies of control files）、RMAN 备份片（Backup pieces）、数据库闪回日志（Flashback log），其中前两项为永久性内容，后 6 项为暂时性内容。

当闪回恢复区出现空间压力时，Oracle 根据如下规则删除其中的暂时性内容：

（1）根据备份的保留策略确定的过时（Obsolete）的内容。

（2）已经备份至磁带上的暂时内容。

（3）根据归档日志的删除策略确定的归档日志。

（4）结合重做日志的需求者（Consumers）确定的可以删除的归档日志，这里日志的需求者包括 RMAN、备用数据库、闪回数据库特性、流复制（Stream Replication）等。

（5）外部归档日志仅存在于逻辑备用数据库的闪回恢复区中，LogMiner 进程已经对其进行了日志挖掘后的外部归档日志。

根据上面的闪回恢复区的删除规则，主要有两项策略决定了闪回恢复区的候选删除内容，一项是备份的保留策略，另一项是归档日志的删除策略。

```
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;

RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 2
TIMES TO DEVICE TYPE DISK;
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 2
TIMES TO DEVICE TYPE SBT;
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON
STANDBY;
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO SHIPPED TO
STANDBY;
```

值得注意的是，当在 Data Guard 环境中的主数据库中配置归档日志的删除策略为 APPLIED ON STANDBY 或 SHIPPED TO STANDBY 时，会出现如下警告：

```
RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON
STANDBY;

new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON STANDBY;
new RMAN configuration parameters are successfully stored
RMAN-08591: WARNING: invalid archived log deletion policy
```

出现此警告的原因是主数据库中并没有设置一个强制性的（Mandatory）归档日志传输目标。解决的方法有两种，一种是至少设置一个强制性的远程归档目标；另一种是设置如下隐含初始化参数：

```
SQL> alter system set "_log_deletion_policy"=ALL scope=spfile;
```

初始化参数 db_flashback_retention_target 确定闪回恢复区中的闪回日志的保存时限，它确定闪回日志的自动删除（注：闪回日志只能存储于闪回恢复区，即使使用 BACKUP RECOVERY AREA 指令，也不能将其备份至其他位置），也会影响到待删除的候选归档日志，凡是在 SYSDATE-'db_flashback_retention_target' 至当前时间范围内的归档日志需要保留在闪回恢复区，应为在数据库闪回的过程中需要相应的归档日志的参与。

另外，保证还原点（Guaranteed Restore Point）需要有配套的闪回日志和归档日志的支持，因此，任何保证还原点需要的闪回日志和归档日志不允许被删除。

14.3.2 监控与管理闪回恢复区

要了解闪回恢复区的空间使用状况，主要依赖于 V$RECOVERY_FILE_DEST 和 V$RECOVERY_AREA_USAGE 两个数据字典视图，前者反映了闪回恢复区的整体使

用信息，后者按内容的类别详细列出了各项存储文件的空间使用信息。

当闪回恢复区出现剩余空间不足时，Oracle 会通过 alert 发出空间预警，并记录于视图 DBA_OUTSTANDING_ALERTS。默认情况下，当 FRA 的剩余空间不足 15% 时发出一般告警（Warning Alert），当 FRA 的剩余空间不足 3% 时发出严重告警（Critical Alert）。当闪回恢复区空间耗尽、不能存储必要内容时，会出现如下错误信息：

```
ORA-19809: limit exceeded for recovery files
ORA-19804: cannot reclaim ... bytes disk space from ... limit
```

如下途径可解决闪回恢复区空间耗尽问题。

（1）扩展闪回恢复区的空间配额（通过参数 DB_RECOVERY_FILE_DEST_SIZE）。

（2）将 FRA 中的备份和归档日志再备份至另外的存储设备，如使用 Backup Recovery Area 指令将其中的暂时性内容备份至磁带。

（3）通过数据库之外的手段（如 OS、ASMCMD 等）删除闪回恢复区中的内容，然后更新存储库（首先执行 CROSSCHECK 操作，然后执行 DELETE EXPIRED 操作）。

（4）删除不必要的保证还原点（Guaranteed restore points）。

（5）重新审视备份的保留策略和归档日志的删除策略。调整这两个策略可以调整闪回恢复区中必须保存的内容。

14.4　闪回数据归档

在表上执行闪回操作将依赖于该表的回滚信息（Undo Data），如果没有足够的回滚信息支持，闪回操作是不能实现的，初始化参数 undo_retention 确定回滚信息的保留时限。回滚表空间（Undo Tablespace）的存储空间是循环使用的，超过时限的回滚信息就可以无条件地被覆盖。事实上，当回滚表空间出现空间压力时，该参数设置的回滚信息保留时限可能被突破，除非回滚表空间上启动了 retention guarantee 特性。

Oracle 的闪回数据归档（Flashback Data Archive）功能大大延长了基于表的数据闪回的时限，当回滚信息超过常规的保留时限时，Oracle 将其移入闪回数据归档，以便支持更长时间的闪回操作。闪回数据归档由一个或多个表空间构成，数据库可以存在一个或多个闪回数据归档。

闪回数据归档是基于表的，默认情况下，表中的闪回数据归档功能处于关闭状态，若需要启动特定表上的闪回数据归档功能，需要满足如下条件：

（1）用户需要在表中具有 Flashback Archive 对象权限。

（2）启动该功能的表不能是嵌套表（Nested）、聚簇表（Clustered）、临时表（Temporary）、远程表（Remote）、外部表（External）。

（3）表中不能有 Long 数据类型的列或嵌套列（Nested Column）。

创建一个闪回数据归档需要指定其名称、至少一个表空间（可选：空间配额）、闪回数据保留时限（天数）等。

```
create flashback archive [default] fba_name
tablespace tbs_name [quota ...] retention ... year|month|day;
```

在维护闪回数据归档的过程中，还可以根据需要删除部分历史的闪回归档数据。

```
alter flashback archive fba_name
purge before timestamp|scn ...;
```

在创建了闪回数据归档后，就可以启动针对特定表上的闪回数据归档功能。

```
create table ... flashback archive [fba_name];
alter table ... flashback archive [fba_name];
alter table ... no flashback archive;
```

需要注意的是，表的闪回操作支持大部分 DDL 语句，但有某些特定 DDL 语句闪回操作不能跨越，如表分区的移动或交换（Exchange）、alter table ... upgrade table 等。如果在启动闪回数据归档表中执行这些 DDL 语句，会返回如下 ORA-55610 错误：

```
ORA-55610: Invalid DDL statement on history-tracked table
```

上述错误的解决方法是在执行不支持的 DDL 语句之前和之后分别调用 DBMS_FLASHBACK_ARCHIVE 包的 DISASSOCIATE_FBA 过程和 REASSOCIATE_FBA 过程，即首先切断表与 FBA 的联系，DDL 后再重新建立两者的联系。

闪回数据归档的相关视图有：*_FLASHBACK_ARCHIVE、*_FLASHBACK_ARCHIVE_TS、*_FLASHBACK_ARCHIVE_TABLES 三个。

14.5 数据库闪回与数据恢复

从数据库恢复的角度看，闪回数据库（Flashback Database）也是一种对过去时刻数据库数据的一种保护措施，利用它可以将数据库暂时或永久地回到过去的某个特定时间点。从效果上看，它是一种基于时间点的数据恢复（Point-in-Time Recovery，PITR），但与传统的基于介质恢复的 PITR 相比，它的效率更高，速度更快。

闪回数据库操作有如下几种形式。

※ 闪回到特定 SCN:

```
flashback database to scn ...;
```

※ 闪回到特定时间点:

```
flashback database to timestamp ...;
```

※ 闪回到指定还原点:

```
flashback database to restore point ...;
```

※ 闪回到最近一次数据库被重置（Resetlogs）前的时刻:

```
flashback database to before resetlogs;
```

从上述数据库的闪回形式可以看出，本质上闪回数据库就是将数据库回退到过去某个特定的时刻，这个特定的时刻从数据库内部来说，就是指特定 SCN 或特定的还原点，而对数据库管理者来说，闪回到特定时间点更直观。但这里需要注意，后者没有前两者准确，因为在数据库内部的“时间”表示就是 SCN，为了将其转换为对应的时间，数据库内部大约每 3 秒做一次 SCN 到时间的匹配，因此，如果使用时间点来闪回数据库，数据库实际回退到的时间点与指点的时间点可能存在误差，其最大误差是 3 秒。

需要注意的是，虽然使用 SCN 来闪回数据库是精确的，但当数据库闪回的范围跨越数据库“重置”（即 Resetlogs 操作，它导致数据库产生不同的 Incarnation）点时，使用 SCN 会产生歧义，如图 14-1 所示，数据库在 Incarnation 1 阶段，SCN 由 1 推进到 2000，在 SCN 为 2000 的时刻，执行数据库闪回，将其闪回到 SCN 为 1000 的时刻，并重置打开，此时产生 Incarnation 2，在此阶段，SCN 由 1000 又推进到 3000，在 SCN 为 3000 的时刻，执行数据库闪回，将其闪回到 SCN 为 2000 的时刻，并重置打开，此时产生 Incarnation 3，在此阶段，SCN 又从 2000 再次推进到 SCN 为 3000 的时刻。

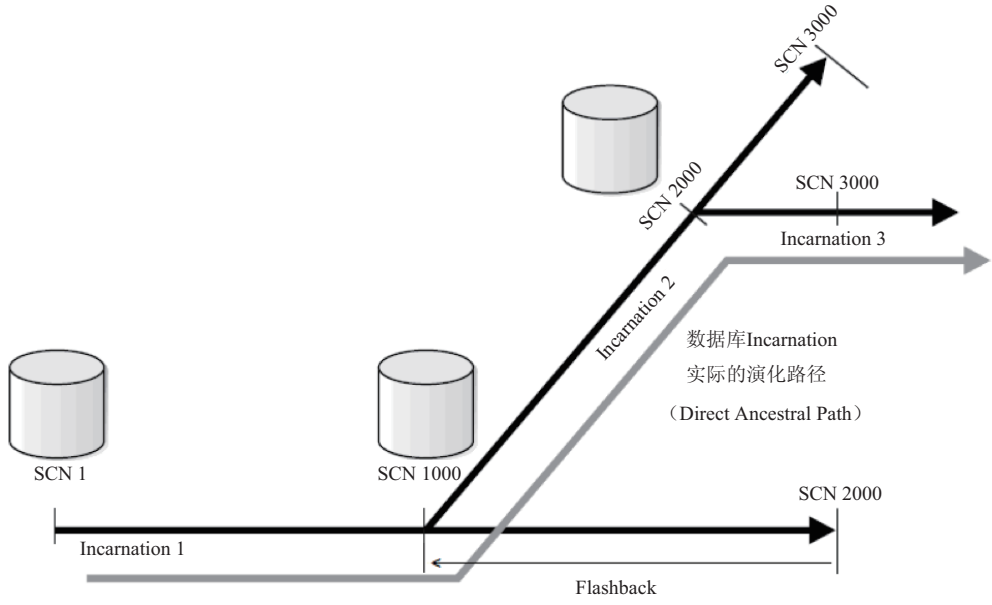


图 14-1　数据库闪回与重置示例

在上述情形下，当通过 SCN 闪回数据库时，指定的 SCN 在 1000 ~ 2000 或 2000 ~ 3000 的时间点都会产生歧义。若 SCN 在 1000 ~ 2000，数据库可以处在 Incarnation 1 阶段，也有可能处在 Incarnation 2 阶段，同样指定 2000 ~ 3000 的时间点，同样存在此类问题。因此，通过 SCN 闪回数据库，在跨越最近的“重置”点时，需要显示地将数据库重置到指定的 Incarnation，类似如下操作：

```
reset database to incarnation 2;
```

特别说明的是，使用时间点闪回数据库或使用还原点闪回数据库不存在上述问题，因为时间点（Timestamp）和还原点（Restore Point）总是和数据库特定的 Incarnation 关联起来，使用它们还原数据库时，不会像使用 SCN 那样产生歧义。

14.5.1　当前演化路径下的闪回

默认情况下，当使用 SCN 执行数据库闪回时，数据库是在实际的 Incarnation 的演化路径（Direct Ancestral Path）上闪回的，参见图 14-1。

（1）查看当前数据库的闪回窗口，代码如下：

```
SQL> select oldest_flashback_scn,
to_char(oldest_flashback_time,'yyyy-mm-dd hh24:mi:ss')
```

```
from v$flashback_database_log;

OLDEST_FLASHBACK_SCN TO_CHAR(OLDEST_FLAS
-------------------- -------------------
              325147 2013-08-10 15:56:59

SQL> select dbms_flashback.get_system_change_number from dual;

GET_SYSTEM_CHANGE_NUMBER
------------------------
                  391588
```

若有必要，还可以进一步通过视图 V$RESTORE_POINT 查看数据库中是否存在还原点可以利用，包括保证还原点。

（2）在执行具体的数据库闪回操作前，建议使用 RMAN 查看当前数据库备份通道（Channel）的设置，包括默认的备份通道，因为在数据库闪回期间，Oracle 可能需要借助于 RMAN 还原和应用必要的归档日志文件。同时，还需要检查必要的归档日志是否存在。

（3）执行数据库闪回操作是在 mount 状态下完成的，将需要闪回的数据库重新启动到 mount 状态。此时，可进一步确认数据库的闪回窗口。

（4）根据需要执行数据库闪回操作。

```
flashback database to scn|timestamp|restore point ...;
```

（5）闪回后的数据库可以临时性地以只读方式打开，查询相关业务数据，以便确认是否是我们需要返回的数据库时刻。

```
alter database open read only;
```

（6）接下来有 4 种选择：闪回的结果满意；进一步向前闪回；数据库向后倒退；取消闪回。下面分别说明：

第一种选择，如果闪回的结果经检查正是我们需要的，则关闭数据库，以“重置”的方式打开即可（注意，此时数据库产生新的 Incarnation）。

```
shutdown immediate;
startup mount
alter database open resetlogs;
```

第二种选择，步骤（5）的闪回结果经只读打开后检查，发现需要进一步向前闪回，此时同样关闭数据库，启动至 mount，再次执行闪回操作即可。

```
shutdown immediate;
startup mount
flashback database to scn earlier_scn ;
```

第三种选择，步骤（5）的闪回结果经检查，闪回得太多了，数据库需要在时间上后退，此时需要执行手工的不完全恢复。

```
shutdown immediate;
startup mount
recover database until scn later_scn ;
```

第四种选择，要完全取消步骤（5）的闪回结果，则需要执行完全的数据库恢复，将数据库恢复到闪回前的状态。经完全恢复后的数据库，完全消除了闪回操作对数据库的影响，数据库返回到当前的最新状态。

14.5.2 闪回到 Incarnation 演化分支

图 14-1 清楚地描述了数据库在演化和闪回的过程中会产生某些“废弃”的 Incarnation 分支，图示中的 Incarnation 1 的 SCN 1000 至 SCN 2000、Incarnation 2 的 SCN 2000 到 SCN 300 都是这样的废弃分支，如果要将数据库闪回到这样的分支上，需要特别处理，因为默认情况下，数据库闪回操作是在 Direct Ancestral Path 上进行的。

由视图 V$DATABASE_INCARNATION 提供的数据可以很容易绘制出数据库 Incarnation 的演化路径图，包括 Direct Ancestral Path 和废弃的 Incarnation 演化分支。这里讨论的闪回到 Incarnation 演化分支，实际上也包含闪回到状态为 ORPHAN 的 Incarnation 的某个时间点。

下面的查询给出了一个示例数据库的 Incarnation 的演化路径：

```
select incarnation# inc#,resetlogs_change# reset#,
prior_resetlogs_change# prior_reset#,
prior_incarnation# prior#,status,flashback_database_allowed
allowed
from v$database_incarnation;

INC#    RESET#    PRIOR_RESET#    PRIOR#  STATUS  ALLOWED
-----------------------------------------------------------------
1        1         0                0     PARENT  NO
2        247813    1                1     PARENT  YES
3        389402    247813           2     ORPHAN  YES
4        389392    247813           2     CURRENT YES
```

根据上面的查询，数据库在第 4 次“重置”过程中，让数据库闪回到 Incarnation 为 2 的起点，这样，Incarnation 为 3 的演化路径全部被废弃掉，其状态为 ORPHAN，但即使这样，数据库仍然可以闪回到 Incarnation 为 3 的某个时间点。

下面将数据库闪回到 Incarnation 为 3 的某个时间点处。结合上面的数据，将数据库闪回到 Incarnation 为 3 的分支上，执行如下环节：

（1）查询视图 V$FLASHBACK_DATABASE_LOG，查看数据库的闪回窗口。

```
select oldest_flashback_scn,
to_char(oldest_flashback_time,'yyyy-mm-dd hh24:mi:ss')
from v$flashback_database_log;

OLDEST_FLASHBACK_SCN TO_CHAR(OLDEST_FLAS
-------------------- -------------------
              327712 2013-08-10 16:27:03
```

（2）确认数据库当前 Incarnation 的前一个 Incarnation 编号。从上面的视图 v$database_incarnation 查询结果看，当前数据库 Incarnation 的前一个 Incarnation 编号是 2，但要将数据库闪回到 Incarnation 为 3 的某个时间点上，这一步查询确认了我们的操作目标。

```
select prior_incarnation# from v$database_incarnation
where status='CURRENT';
```

（3）关闭数据库，将数据库启动到 mount 状态。

（4）运行 RMAN，连接到目标数据库。将数据库 Reset 到特定的 Incarnation 上，这一步是此项操作的关键。

```
RMAN> reset database to incarnation 3;

using target database control file instead of recovery catalog
database reset to incarnation 3
```

（5）执行数据库闪回。通过前面对 v$database_incarnation 的查询，我们了解到数据库在 Incarnation　3 阶段，其 SCN 应该大于 389402，作为演示，我们将数据库闪回到 SCN　389410，此 SCN 时刻一定是在 Incarnation　3 阶段。

```
RMAN> flashback database to scn 389410;
Starting flashback at 12-AUG-13
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=69 device type=DISK
```

```
starting media recovery
archived log for thread 1 with sequence 86 is already
on disk as file /DB_FRA/BJING/ARCHIVELOG/2013_08_12/O1_
MF_1_86_90JHF9DZ_.ARC
archived log for thread 1 with sequence 1 is already on disk as
file /DB_FRA/BJING/ARCHIVELOG/2013_08_12/O1_MF_1_1_90JHFLSP_.
ARC
archived log for thread 1 with sequence 2 is already on disk as
file /DB_FRA/BJING/ARCHIVELOG/2013_08_12/O1_MF_1_2_90JHRLJD_.
ARC
media recovery complete, elapsed time: 00:00:05
Finished flashback at 12-AUG-13
```

此处确认一下闪回后的 Incarnation 的变化情况。从下面的查询结果可以清楚地看出，当前数据库闪回到 Incarnation 为 3，Incaranation 为 4 状态由之前的 CURRENT 变为 ORPHAN，这是很显然的，当前数据库的 Direct Ancestral Path 已不包含 Incaranation 4 阶段。

```
select incarnation# inc#,resetlogs_change# reset#,
prior_resetlogs_change# prior_reset#, prior_incarnation#
prior#,
status,flashback_database_allowed allowed
from v$database_incarnation;

INC#     RESET#   PRIOR_RESET#     PRIOR# STATUS  ALLOWED
---------------------------------------------------------------
1        1        0                0      PARENT  NO
2        247813   1                1      PARENT  YES
3        389402   247813           2      CURRENT YES
4        389392   247813           2      ORPHAN  YES
```

此后的操作与常规的在 Incarnation 演化路径（Direct Ancestral Path）上的数据库闪回是一致的，在此省略。

14.6 数据库闪回在 Data Guard 中的应用

在 Data Guard 中利用数据库闪回技术可以将备用数据库临时性地以读 / 写模式打开，用于真实模拟对主数据库的某种应用测试。当应用测试完毕后，再将数据库闪回到之前的状态，作为备用数据库正常运行。

14.6.1　基于保证还原点的应用

要使物理备用数据库能够以读 / 写模式打开，并能在之后再返回到备用数据库的状态，备用数据库必须启动数据库闪回特性。物理备用数据库一旦以读 / 写模式打开，它将相当于一个独立的数据库在运行（实际上相当于另外一个主数据库），利用数据库的闪回特性，可以将数据库在物理备用和独立数据库之间相互切换，以满足各种应用需求，这极大地提高了 Data Guard 中物理备用数据库在使用上的灵活性。

（1）取消物理备用数据库的托管恢复状态。

```
SYS@SHHAI>alter database recover managed standby database
cancel;
Database altered.
```

（2）设置备用数据库的闪回恢复区，启动数据库闪回特性。

```
SYS@SHHAI>alter system set db_recovery_file_dest='...'
scope=spfile;

SYS@SHHAI>alter system set db_recovery_file_dest_size=...
scope=spfile;

SYS@SHHAI>alter database flashback on;
Database altered.

SYS@SHHAI>select flashback_on from v$database;
FLASHBACK_ON
------------------
YES
```

（3）创建一个保证还原点。

保证还原点确保数据库不会因闪回恢复区的空间压力而自动删除还原点及其相关归档日志。

```
SYS@SHHAI>create restore point before_open_rw guarantee
flashback database;
Restore point created.
```

（4）在主数据库中将当前重做日志归档或切换日志。

此步可选，执行它是为了确保在备用数据库中闪回到上面的保证还原点需要的归档日志已传输到物理备用数据库端。

```
SYS@BJING>alter system archive log current;
System altered.
```

（5）将主数据库向物理备用数据库传输日志的归档目标状态设置为延迟状态（Defer）。

此步可选，设置它是为了避免因主数据库向物理备用数据库端传输日志受阻而在告警日志中产生错误报告。

```
SYS@BJING>show parameters log_archive_dest_2

NAME                                 TYPE        VALUE
------------------------------------------------------
log_archive_dest_2                   string
SERVICE=SHHAI LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=SHHAI

SYS@BJING>alter system set log_archive_dest_state_2='defer';
System altered.
```

（6）激活物理备用数据库。此步直接将物理备用数据库转换为独立运行的数据库，此时，物理备用数据库完全脱离了主数据库，实际上是另外一个主数据库。

```
SYS@SHHAI>alter database activate physical standby database;
Database altered.

SYS@SHHAI>alter database open;
Database altered.

SYS@SHHAI>select database_role,open_mode,flashback_on
from v$database;

DATABASE_ROLE    OPEN_MODE            FLASHBACK_ON
---------------- -------------------- ------------------
PRIMARY          READ WRITE           YES
```

（7）在激活后的物理备用数据库上执行必要的应用测试，如创建数据库对象、加载数据、运行应用执行各种数据处理等。

（8）测试完毕，将数据库还原到之前的保证还原点。

```
SYS@SHHAI>shutdown
SYS@SHHAI>startup mount
```

```
ORACLE instance started.
Total System Global Area  267227136 bytes
Fixed Size                  2174888 bytes
Variable Size             230686808 bytes
Database Buffers           29360128 bytes
Redo Buffers                5005312 bytes
Database mounted.
SYS@SHHAI>flashback database to restore point before_open_rw;
Flashback complete.
SYS@SHHAI>select database_role,open_mode,flashback_on
  2  from v$database;
DATABASE_ROLE    OPEN_MODE                    FLASHBACK_ON
---------------- -------------------- ------------------
PRIMARY            MOUNTED                          YES
```

注意，此时数据库仍然是一个独立运行的数据库，只是数据库中的数据状态被闪回到激活前的保证还原点状态。

（9）将数据库转换到物理备用数据库的角色。

```
SYS@SHHAI>alter database convert to physical standby;
Database altered.
SYS@SHHAI>shutdown
ORA-01507: database not mounted
ORACLE instance shut down.
SYS@SHHAI>startup mount
ORACLE instance started.
Total System Global Area  267227136 bytes
Fixed Size                  2174888 bytes
Variable Size             230686808 bytes
Database Buffers           29360128 bytes
Redo Buffers                5005312 bytes
Database mounted.
SYS@SHHAI>select database_role,open_mode,flashback_on
  2  from v$database;

DATABASE_ROLE            OPEN_MODE            FLASHBACK_ON
---------------- -------------------- ------------------
PHYSICAL STANDBY   MOUNTED              YES
```

（10）在主数据库中启动向备用数据库端传输日志。

```
SYS@BJING>alter system set log_archive_dest_state_2='enable';
System altered.
```

（11）启动物理备用数据库的托管日志恢复。

```
SYS@SHHAI>alter database recover managed standby database
disconnect;
Database altered.
```

（12）检查 Data Guard 的运行及备用数据库的日志应用情况。

14.6.2 使用快照备用数据库

前面在创建保证还原点后将物理备用数据库激活，以读 / 写模式打开，执行一些必要的应用测试和数据处理后，再利用数据库闪回技术将数据库闪回到保证还原点，之后再将其转换为物理备用数据库，这个过程是比较烦琐的，并且在物理备用数据库以读 / 写模式打开后，实际上是作为一个独立的数据库存在的，因此，这个期间数据库停止接收来自主数据库的重做日志，这实际上就失去了对主数据库的保护。

快照备用数据库（Snapshot standby database）的诞生消除了上述过程带来的缺点，它在使物理备用数据库处于可读 / 写状态的同时，可以继续接收来自主数据库的重做日志，只是暂时不应用重做日志。因此，物理备用数据库成为快照备用数据库期间，它暂时脱离了对主数据库的跟随，体现在两个方面：一是主数据库更新而快照备用数据库并没有同步更新；二是快照备用数据库本身的更新导致其数据进一步偏离主数据库的数据。不过这是暂时的，当结束快照备用数据库状态后，数据库可以重新返回到物理备用数据库的轨道上来。

1．将物理备用数据库转换至快照备用数据库

首先，需要停止物理备用数据库的托管恢复。如果物理备用数据库是 RAC 架构，则保留一个实例，关闭其他所有实例。

其次，确保实例处于 mount 状态。为了方便快照备用数据库期间接收来自主数据库的重做日志，建议备用数据库配置闪回恢复区（Flash/Fast Recovery Area）。

下列指令将物理备用数据库转换为快照备用数据库。

```
SYS@SHHAI>alter database convert to snapshot standby;
Database altered.
```

从上述指令的跟踪文件中可以看出其中的转换过程，代码如下：

```
Sat Aug 17 16:31:27 2013
alter database convert to snapshot standby
```

```
Created guaranteed restore point
SNAPSHOT_STANDBY_REQUIRED_08/17/2013 16:31:27
krsv_proc_kill: Killing 2 processes (all RFS)
Begin: Standby Redo Logfile archival
End: Standby Redo Logfile archival
RESETLOGS after complete recovery through change 413394
Resetting resetlogs activation ID 3966061407 (0xec654b5f)
Online log +SHGRP/shhai/onlinelog/group_1.283.823686323:
Thread 1 Group 1 was previously cleared
Online log +SHGRP/shhai/onlinelog/group_2.284.823686325:
Thread 1 Group 2 was previously cleared
Standby became primary SCN: 413392
Sat Aug 17 16:31:30 2013
Setting recovery target incarnation to 7
CONVERT TO SNAPSHOT STANDBY: Complete - Database mounted as
snapshot standby
Completed: alter database convert to snapshot standby
```

可以进一步验证，快照备用数据库本质上也是一个独立的主数据库。

```
SYS@SHHAI>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     1
Next log sequence to archive   1
Current log sequence           1
SYS@SHHAI>
SYS@SHHAI>select database_role,open_mode from v$database;

DATABASE_ROLE            OPEN_MODE
---------------- --------------------
SNAPSHOT STANDBY         MOUNTED

SYS@SHHAI>alter database open read write;
Database altered.

SYS@SHHAI>select database_role,open_mode from v$database;
DATABASE_ROLE            OPEN_MODE
---------------- --------------------
```

```
SNAPSHOT STANDBY   READ WRITE
```

以读 / 写方式打开后的快照备用数据库有如下两个方面需要注意：

一是在功能上它相当于一个独立的数据库，在转换到物理备用数据库之前，不能对其进行角色转换（Switchover 或 Failover）。

二是在最大保护模式下，不能将 Data Guard 环境中的唯一物理备用数据库转换为快照备用数据库。

2．转换快照备用数据库至物理备用数据库

在执行了必要的应用测试和数据处理后，快照备用数据库可以再次转换为物理备用数据库，可在 mount 状态下执行如下转换。如果是 RAC 架构，则仅加载其中的一个实例，关闭其他所有实例。

```
SYS@SHHAI>alter database convert to physical standby;
Database altered.
```

跟踪文件揭示了上述转换过程。本质上和使用保证还原点、数据库闪回技术的执行过程是一致的。

```
Sat Aug 17 16:54:08 2013
alter database convert to physical standby
krsv_proc_kill: Killing 2 processes (all RFS)
Flashback Restore Start
Flashback Restore Complete
Flashback Media Recovery Start
Sat Aug 17 16:54:12 2013
Setting recovery target incarnation to 5
 started logmerger process
Parallel Media Recovery started with 4 slaves
Flashback Media Recovery Log
 /DB_FRA/SHHAI/ARCHIVELOG/2013_08_17/O1_MF_1_16_90XRR7F4_.ARC
Flashback Media Recovery Log
 /DB_FRA/SHHAI/ARCHIVELOG/2013_08_17/O1_MF_1_17_90XRR7D3_.ARC
...
Flashback Media Recovery Log
 /DB_FRA/SHHAI/ARCHIVELOG/2013_08_17/O1_MF_1_20_90XS1PKM_.ARC
Sat Aug 17 16:54:17 2013
Incomplete Recovery applied until change 413394 time
08/17/2013 10:47:17
Flashback Media Recovery Complete
```

```
Setting recovery target incarnation to 7
Sat Aug 17 16:54:18 2013
Guaranteed restore point  dropped
Clearing standby activation ID 3966446547 (0xec6b2bd3)
The primary database controlfile was created using the
'MAXLOGFILES 32' clause.
There is space for up to 30 standby redo logfiles
Use the following SQL commands on the standby database to
create
standby redo logfiles that match the primary database:
ALTER DATABASE ADD STANDBY LOGFILE 'srl1.f' SIZE 10485760;
ALTER DATABASE ADD STANDBY LOGFILE 'srl2.f' SIZE 10485760;
ALTER DATABASE ADD STANDBY LOGFILE 'srl3.f' SIZE 10485760;
Completed: alter database convert to physical standby
```

上述转换之后，实例处于 nomount 状态，需要重新启动至 mount 状态。之后第一次启动托管恢复的过程如下：

```
Sat Aug 17 17:00:42 2013
RFS[3]: Assigned to RFS process 5624
RFS[3]: Identified database type as 'physical standby': Client
is ARCH pid 6696
Sat Aug 17 17:00:52 2013
alter database recover managed standby database disconnect
Sat Aug 17 17:00:52 2013
MRP0 started with pid=29, OS id=4364
 started logmerger process
Sat Aug 17 17:00:57 2013
Managed Standby Recovery not using Real Time Apply
Parallel Media Recovery started with 4 slaves
Completed: alter database recover managed standby database
disconnect
Waiting for all non-current ORLs to be archived...
All non-current ORLs have been archived.
Clearing online redo logfile 1
 +SHGRP/shhai/onlinelog/group_1.283.823686323
Clearing online log 1 of thread 1 sequence number 25
Clearing online redo logfile 1 complete
Clearing online redo logfile 2
 +SHGRP/shhai/onlinelog/group_2.284.823686325
```

```
Clearing online log 2 of thread 1 sequence number 26
Clearing online redo logfile 2 complete
Media Recovery Log
 /DB_FRA/SHHAI/ARCHIVELOG/2013_08_17/O1_MF_1_21_90YF3PB5_.ARC
...
Media Recovery Log
 /DB_FRA/SHHAI/ARCHIVELOG/2013_08_17/O1_MF_1_25_90YGWLBZ_.ARC
Media Recovery Waiting for thread 1 sequence 26 (in transit)
```

第15章

集群数据库系统 RAC

要实现一个数据库系统的高可用性（High Availability，HA），一个基本的途径就是实施数据冗余。Oracle 数据库集群系统的解决方案被称为 RAC（Real Application Cluster），它是从实例（Instance）的角度实现数据冗余。相比较而言，本书重点讨论的 Data Guard 是从数据库的角度实现数据冗余的，数据库和实例之间存在天然的密切联系，即实例需要数据库的存储支持，数据库需要实例作为运行的保障，正因为如此，在实际的数据库系统部署中，Data Guard 与 RAC 往往是作为一个 Oracle 数据库高可用系统的两个主要方面。

为了叙述 Data Guard 在 RAC 系统的应用，本章对 Oracle RAC 的核心内容进行比较系统的介绍，以便读者对 RAC 的架构和运行有一个比较全面的了解。

15.1 RAC 体系结构

对 Oracle 来说，集群数据库系统的本质是将同一个物理存储的数据库运行在由网络连接的不同的服务器上，以多个实例（Instance）的形态呈现，形成一个多实例的数据库环境。与传统的单实例数据库环境相比，多实例环境需要解决下面两个基本问题：

（1）用户访问数据库时总是连接某个特定的实例。多实例环境下如何协调多个实例在运行过程中的数据及数据处理，保障用户访问的数据一致性。

（2）单实例环境，实例是以独占（Exclusive）的方式访问数据库的。多实例环境下，多个运行的实例需要并行访问数据库，而操作系统支持的普通文件系统不能满足这一需要，这就对数据库的物理存储提出了更高的要求。

针对第一个问题，Oracle 开发出整合多服务器、多实例的集群件 Clusterware，可以认为集群件是一个集群操作系统，它将多个分立的服务器资源整合成一个逻辑上的整体，统一为用户提供资源管理及其服务。

针对第二个问题，Oracle 为多实例环境下的数据库提供了两种主要的存储方

案——自动存储管理（Automatic Storage Management，ASM）和集群文件系统（Oracle Cluster File System，OCFS），其中，ASM 是 Oracle 推荐的首选集群数据库存储方案，ASM 不仅是一种集群文件系统，同时还提供动态卷管理功能。

从 11g 版本开始，Oracle 将集群件 Clusterware 和存储方案 ASM 合并，对外统一称为 Grid Infrastructure（网格基础架构），这就是 Oracle 的集群解决方案，它不仅可以用于集群数据库，同时还可用于其他的集群环境。

15.1.1 RAC 总体框架图

集群环境中的每个服务器被称为节点（Node），本节以具有两个节点的集群环境为例说明 Oracle RAC 的总体架构，如图 15-1 所示。

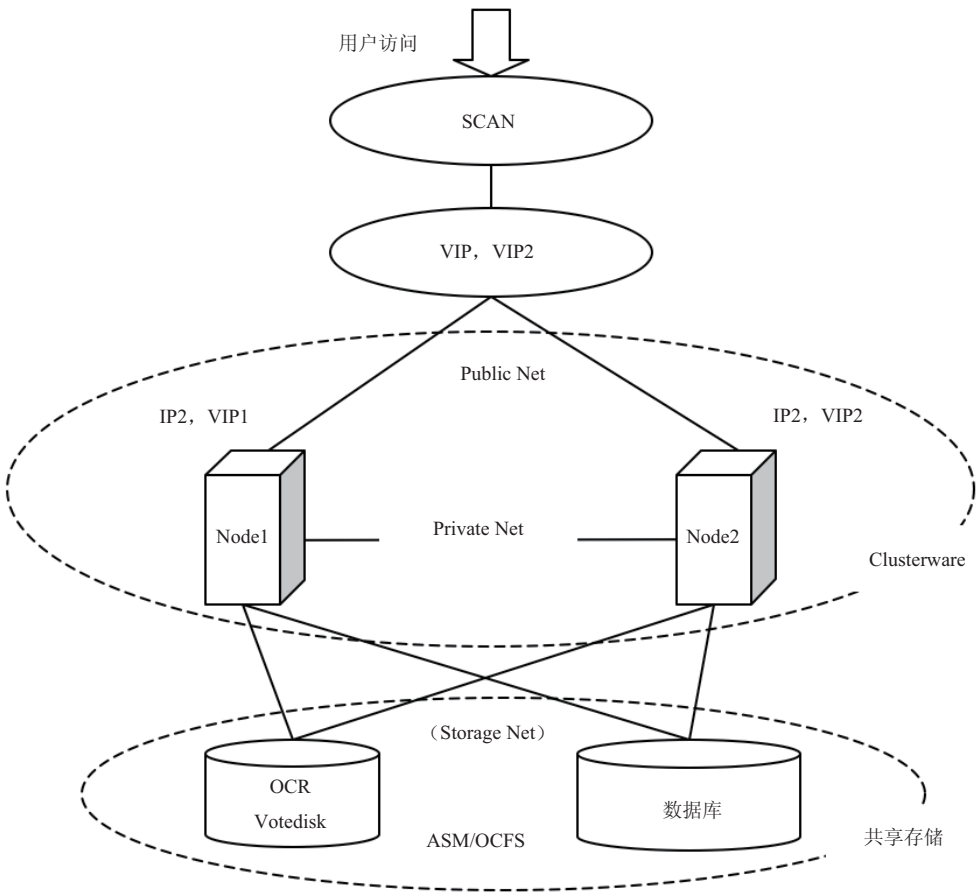


图 15-1 Oracle RAC 总体架构示意图

从图 15-1 中可以看出，Oracle 的集群数据库系统 RAC 由如下几个关键部分组成：

※ 共享存储部分。

※ 集群件 Clusterware。

※ 网络访问部分。

下面分别介绍各个部分的主要内容。

15.1.2　共享存储部分

在 Oracle 的集群系统中，每个节点服务器除了拥有自己的本地存储外，还需要能够访问一个共享的存储系统，集群中所有的节点都能够访问该存储系统。这个共享的存储系统的主要存储目的有两个，一是为集群这样一个逻辑上的整体提供关键信息的存储，如集群的配置信息、集群中的节点信息等，具体来说就是图 15-1 中的集群注册表（Oracle Cluster Registry，OCR）和表决磁盘 Votedisk（也写作 Voting Disk）；二是共享的数据库物理存储，Oracle 的 RAC 集群是以各节点共享数据库为前提的，所有的节点都需要能够访问同一个数据库，集群环境中启动的多个实例运行在不同的节点上。

很显然，这里的共享存储需要支持各个节点的并发访问，这就决定了共享存储的信息不能存放于普通的文件系统（仅支持单一节点的独占访问）上。从 Oracle 集群 RAC 的发展历史看，共享存储系统有如下 3 种选择：

※ 裸设备（Raw Device）。

※ 集群文件系统（Cluster File System，CFS）。

※ 自动存储管理（ASM）。

所谓裸设备，是没有格式化的磁盘或磁盘分区，其上没有建立某种文件系统。在 RAC 中，如果共享存储采用裸设备，Oracle 软件（Clusterware 或 DBMS）直接接管该磁盘，以块模式读 / 写。这种使用磁盘的方式由于越过了操作系统的环节，通常能够比文件系统有更快的访问速度。但由于这种方式缺乏常规的卷管理系统的支持，裸设备的使用缺乏灵活性，并且不利于 DBA 对存储系统的日常管理。早期的裸设备的使用在访问效率上显示出优势，但由于现代的很多卷管理与文件系统已经能够提供接近裸设备的访问速度，因此，从 Oracle 11g（11.2 版本）后已经放弃对裸设备的支持。

集群文件系统是 RAC 共享存储的另一种选择。集群文件系统能够提供多节点的并发访问，常见的集群文件系统有 Veritas Cluster filesystem、Sun Cluster、GlusterFS、GFS2、GPFS 等。Oracle 在 Linux 环境下为 RAC 提供了专门的集群文件系统 OCFS，目前的最新版本是 OCFS2，它主要是为 RAC 环境下

共享数据库物理存储而设计的。由于 ASM 的出现，在 Oracle RAC 中共享存储采用第三方的集群文件系统（包括 Oracle 自己的 OCFS）并不常见，Oracle 强烈推荐使用 ASM 作为 RAC 环境共享存储的首选，目前，ASM 已经支持包括 SPFILE、OCR、Voting Disk、Database、Flash Recovery Area 等所有需要共享内容的存储。

ASM 是一种综合的磁盘存储解决方案，它既是一种磁盘卷管理器，又是一种集群文件系统，它访问物理磁盘分区并管理分区中的内容，为 Oracle 共享存储需要创建的文件提供高效支持，其最大的特点是简化了 DBA 对物理存储的管理。在 ASM 中，使用磁盘组代替单个磁盘，一个磁盘组可以由多个磁盘构成，可以像使用普通磁盘一样使用磁盘组。Oracle 借助于存储虚拟层（Storage Virtualization Layer），自动跨越多个物理磁盘，将数据库的物理存储均匀分布于磁盘组，从而实现磁盘访问 I/O 平衡。

另外，在使用磁盘组进行实际的数据库物理存储时，Oracle 的 ASM 特性还提供了冗余存储数据库文件的方式，充分利用这种功能，在实际环境中显著降低了使用冗余磁盘阵列（RAID）的必要性，从而在不降低数据库可靠性的前提下，可以降低构建数据库服务器的硬件投资成本。

15.1.3 集群件（Clusterware）

集群件是实现任何集群系统的核心组件，Oracle 为集群系统提供的集群件称为 Clusterware，它是一款独立于数据库 DBMS 的软件产品。利用 Clusterware，不仅可以实现数据库集群，还可以为其他第三方软件提供集群平台，如实现应用服务器集群、Web 服务器集群等。反之，实现 Oracle 数据库集群，也可以基于其他的集群件产品，如 IBM 公司的 HACMP、惠普公司的 ServiceGuard 等，但是即使使用第三方集群平台，也不能完全脱离 Clusterware，还需要其部分功能的支持。

正因为上述原因，在实践中如果要实现 Oracle 数据库集群（RAC），Clusterware 和 Oracle 数据库软件的配合是最佳组合，而且可以完全脱离第三方集群管理软件。

Clusterware 集群件的主要功能是，为上层应用软件提供一个完整的集群平台，即为上层需要并行运行的任何软件资源提供高可用性。从逻辑上看，Clusterware 集群件将若干独立运行的服务器整合成一个有机的整体，将节点上各自运行的操作系统整合成一个集群操作系统，各种高可用资源运行在集群操作系统上。

在 RAC 环境的各个节点上，Clusterware 是运行在 Oracle DBMS 和节点操作系统（或称 OS Kernel）之间的一个中间层，首先，它会在 OS Kernel 之前截获数据库系统的各种请求，然后，通过集群中的私有网络和其他节点上的 Clusterware 协作，最终完成数据库的应用处理，从而顺利完成各节点对共享存储数据库的并发访问和数据的一致性处理。

15.1.4　网络访问部分

传统上，要想访问网络上的某个主机，是通过主机的 IP 地址实现的，即 IP 地址和物理主机存在确定的对应关系，这样的 IP 地址有时又称为 Public IP。从图 15-1 所示的 Oracle RAC 总体架构示意图可以看出，通过集群件 Clusterware 的作用，各个独立的节点服务器被联合成一个整体，即集群服务器。不论集群环境中存在多少个节点，集群服务器总是作为一个整体为用户服务的。在集群中，如何协调个体（节点主机）和整体（集群服务器）之间的关系？Oracle 的解决方案是通过虚拟 IP 地址（Virtual IP Address，简称 VIP）实现对集群的访问。

虚拟 IP 地址是和传统的物理 IP 地址相比较，它并不必然与某个确定的主机相对应，它是由软件实现的一种高可用资源，由 Clusterware 负责维护。对 RAC 集群外的环境而言，每个节点主机除了具有传统意义上的物理 IP 地址外，还具有一个由软件实现的虚拟 IP 地址。初始地，一个虚拟 IP 地址通常是和某个节点主机相关联的，但这种联系不是必然的，虚拟 IP 地址是一种由集群件维护的资源，当初始关联的主机不可用时（或关闭、或出现故障或由于某种原因被集群件踢出集群），虚拟 IP 地址会“漂移”至其他可用节点，从而实现资源的高可用性。

RAC 集群数据库的用户正是通过虚拟 IP 地址访问集群数据库的，因此，只要集群中有一个节点可用，用户就可以通过虚拟 IP 地址顺利访问 RAC 集群服务器。

从 Grid Infrastructure 11gR2 开始，Oracle 引入了一个新功能——SCAN(Single Client Access Name)，SCAN 像一个“虚拟主机”名称或域名一样，可以解析出 IP 地址（可以解析至少 1 个 IP，最多解析 3 个 IP），解析出的 IP 地址称为 SCAN IP。这个“虚拟主机”就是集群。

SCAN IP 也是一种虚拟 IP，SCAN IP 必须与 PUBLIC 和 VIP 在一个子网，同时，Oracle 建议使用 DNS 或者 GNS（Grid Naming Service，11gR2 新功能）来解析 SCAN。如果没有使用 DNS 或者 GNS，可以使用主机的 hosts 文件解析 SCAN IP，示例如下：

```
root> cat /etc/hosts
myscan.mydomain IN A 10.10.0.11 IN A 10.10.0.12 IN A
10.10.0.13
```

RAC 集群有了 SCAN 名称后，客户端就可以通过这个 SCAN 名字来访问 RAC 集群数据库，使用 SCAN 的优点是 SCAN 名称代表的集群，而不是集群中的某个节点。当集群中新增加了节点或者删除了节点时，不需要额外维护客户端。在 11gR2 上，客户端仍然可以继续使用原有的 VIP，但是 Oracle 推荐使用 SCAN。例如，我们可以在客户端使用轻松连接命名（EZConnect）、JDBC 等通过如下方式连接 RAC 集群数据库：

```
EZconnet: sqlplus system/manager@RAC-SCAN:1521/ORCL
```

```
JDBC connect: jdbc:oracle:thin:@ RAC-SCAN:1521/ORCL
```

这里的 RAC-SCAN 就是集群的 SCAN 名称，它相当于单实例环境下的主机名称。

RAC 集群中的 SCAN 总是与 SCAN IP 和 SCAN Listener 密切相关，并且 SCAN IP 和 SCAN Listener 总是成对出现的，也就是说如果有 3 个 SCAN IP，就会同时有 3 个 SCAN Listene。SCAN IP 就是 DNS 解析的 IP 地址，SCAN Listener 的作用是接受客户端的连接请求。

事实上，SCAN 是一种服务器端的负载均衡技术，不需要像之前那样在客户端配置文件中使用多个 VIP 以实现负载均衡。客户端发出连接数据库的请求，DNS 将 SCAN 解析出对应的 SCAN IP 并返回给客户端，客户端随机选择其中一个 SCAN IP 地址，首先，客户端通过这个 SCAN IP 访问对应的节点，当对应节点的 SCAN Listener 接收到请求后，SCAN Listener 会选择负荷最小的数据库实例，然后，这个数据库实例对应的本地监听 Local Listener 的地址将会返回给客户端，最后，这个本地监听为客户端请求建立数据库连接。

上面介绍的是公共网络部分（Public Net），RAC 集群中还有一个私有网络（Private Net）。集群环境中的私有网络是实现各个节点 Clusterware 相互协作的重要环节。通过私有网络，各个节点之间相互通信，以及时掌握彼此的状况；通过私有网络，由集群件维护的集群资源才能根据需要在节点间分布运行，从而实现资源的高可用性。正是集群件 Clusterware 加上私有网络的作用，才会将各个分立的节点整合成一个有机的整体对外提供服务。RAC 集群中的私有网络实现两个方面的作用：一是实现节点间的网络心跳（Network Heartbeat），以及时探测各节点的状态；二是各节点实例通过私有网络交换数据，实现缓存融合（Cache Fusion）。

15.2 OCR 与 Voting Disk

集群注册表（OCR）和表决磁盘（Voting Disk）是使用集群件 Clusterware 构造集群系统的关键部件，要认识 Oracle 集群系统，需要对这个关键部件有比较清晰的了解。

15.2.1 集群注册表（OCR）

OCR 是关于集群的元数据（Metadata）的存储中心，它记录了关于集群的配置信息，存储了集群就绪服务（Cluster Ready Service，CRS）维护的各种高可用资源、应用资源的配置及其状态信息，同时，还存储了 CRS 守护进程的配置信息。

OCR 中存储的内容有如下一些项目：

※ 集群的节点成员信息。

※ 高可用资源及其状态信息，如 VIP、Listener、SCAN_Listener 等。

※ RAC 数据库资源的配置信息，如 Database、Instance、Services 等。

※ 数据库服务的配置信息，如服务的 preferred/available 节点、负载均衡目标、TAF（Transparent Application Failover）策略等。

※ 由 CRS 维护的第三方应用的交互与管理信息。

※ ASM 实例及其磁盘组的信息。

※ Clusterware 的各种后台进程、CRS 守护进程的信息。

※ 集群所涉及的网络接口及其配置信息，如 Public Net、Private Net 等。

※ Voting Disk 的位置信息（11g and Later）。

※ 关于 OCR 本身的备份信息。

在集群运行期间，出于性能的原因，每个节点在本地内存中维护了 OCR 内容的一份复制，该块内存被称为本地 OCR Cache。Oracle 使用分布式共享内存架构（Distributed Share Cache Architecture）管理 OCR 的信息，本地的 CRSD 进程负责传播与刷新 OCR Cache 中的内容，但只有一个节点的 CRSD 进程可以读 / 写 OCR 磁盘（该节点被标识为 Master 节点）。一些 OCR 客户端程序（如 crsctl、srvctl 等）在使用时仅利用 CRSD 进程与本地的 OCR Cache 进行通信。

值得注意的是，当使用 ocrconfig 修改 OCR 配置（如 OCR 的存储位置发生变化、改变 OCR 镜像等）时，如果有节点关闭，这些修改的信息并不能传播到这些关闭的节点，即不能修改本地的 OCR 配置文件（大多数平台下 OCR 磁盘的位置信息记录在本地文件 /etc/oracle/ocr.loc 中）。当这些节点再次打开时，并不能感知 OCR 在节点关闭期间的变化，因此，该节点就不能正常加入集群。需要的处理是：在节点打开、CRS 关闭的情况下使用 ocrconfig -repair 指令修改 OCR 的配置信息。

OCR 中存储了集群中最关键的配置信息，OCR 是是集群的核心。为了维护 OCR 的安全性，Oracle 集群件每 4 个小时自动对其执行备份（物理备份），并保留最近 3 次的备份结果，同时还保留最近一天、最近一周的最后一次备份。注意，OCR 的自动备份仅在集群的其中一个节点（Master Node）上执行，备份的结果保存在 $GRID_HOME/cdata/<cluster name> 目录。当然，如果 Master 节点关闭，集群会产生新的 Master 节点。也可手工执行对 OCR 的物理备份。

利用 OCR 物理备份恢复 OCR 称为还原（Restore），OCR 的还原操作需要以超级用户（Root）身份执行。还原 OCR 时，CRS 需要以独占（Exclusive）模式启动（#crsctl start crs -excl），此种方式的启动 Oracle 并不需要 OCR 的存在，即在没有 OCR 的情形下，CRS 可以独占方式启动（并确保资源 ora.crsd 处于停止状态，#crsctl stop resource ora.crsd -init），在此种状态下，才可执行 OCR 的还原操作。

15.2.2 表决磁盘（Voting Disk）

集群是节点的集合。Oracle集群中的表决磁盘是节点作为集群成员的登记场所，没有这个登记信息，节点成员的身份就难以确定，集群就不能维持。每个节点在加入集群或离开集群（如节点启动、关闭、CRS 启动或停止、节点被隔离、节点重启等）时，CCSD 进程都需要负责向表决磁盘进行登记（Register/Unregister），以及时了解集群中的节点成员信息。

同时，表决磁盘也是集群中各节点之间相互通信的媒介，即通过表决磁盘，集群保持磁盘心跳（Disk Heartbeat）。每个节点在表决磁盘中有自己专有的心跳数据块（1 OS Block），集群同步服务守护进程（Cluster Services Synchronization Daemon，CSSD）负责监控与维护此数据块的内容，心跳计数器每秒更新对节点心跳的计数，如果一个心跳数据块持续没有更新，集群会认为该节点存在某种问题(Unhealthy)，有可能会导致节点重启以保护其上的高可用资源，如 RAC 数据库。一个监控的节点需要同时维持网络心跳（通过私有网络）和磁盘心跳（通过表决磁盘）。

表决磁盘中存储的内容有如下两类。

※ 静态数据：集群中的节点成员信息（Cluster Nodes Membership）。

※ 动态数据：节点的磁盘心跳记录（Disk Heartbeat Logging）。

由于网络心跳、磁盘心跳出现异常，或节点由于某种原因被 hang 住等，导致集群产生脑裂(Split-brain)问题，此时Clusterware必须检查表决磁盘中的内容，以决定集群分裂后的哪些部分被踢出、哪个部分接管集群。在这个过程中每个正常的节点会在表决磁盘中及时更新自己的内容，以向集群标识自己的投票身份（Vote）存在。例如，在配置了 3 个表决磁盘的两节点集群系统中，如果第一个节点在 3 个表决磁盘中及时标识了自己的 Vote 身份，而第二个节点由于某种原因或反应延迟只在其中的两个表决磁盘中标识了自己的 Vote，则在出现脑裂时，第二个节点会被踢出集群，第一个节点接管集群。因此，Oracle 建议为集群配置奇数个表决磁盘。

当集群配置了奇数个表决磁盘时，在任何时刻，集群中的节点必须能够及时更新超过半数的表决磁盘，否则，该节点会被踢出集群。如果由于表决磁盘本身的原因导致集群不能访问超过半数的表决磁盘，则整个集群将停止运行。

表决磁盘是 Clusterware 集群共享存储的关键部分之一，早期的表决磁盘只能存储于裸设备上，从 11gR2 开始，表决磁盘可以选择存储于 ASM 磁盘组或集群文件系统中，Oracle 已经放弃对裸设备的支持，并且建议使用 ASM 存储表决磁盘，且表决磁盘的冗余也可方便地通过 ASM 磁盘组的冗余方式实现。

这里有一个令人困扰的问题：集群中的节点在启动时需要访问表决磁盘，而只有在节点启动、ASM 实例启动后加载了磁盘组后 Oracle 才能访问磁盘组中的文件。

为了解决这个问题，Oracle 在每个 ASM 磁盘上的固定位置保留了若干数据块来存储表决磁盘的信息，这样，在 ASM 实例没有启动时集群同步服务就可以访问表决磁盘。表决磁盘在 ASM 磁盘上的存储位置是固定的，因此，在加载的 ASM 磁盘组上并不能看到表决磁盘对应的 ASM 文件（每个表决磁盘在 ASM 磁盘中有唯一的 File Universal ID）。如果 OCR 存储在 ASM 磁盘组中，则我们可以清楚地看到对应的 ASM 文件。

另外，如果选择外部（External）冗余的 ASM 磁盘组存储表决磁盘，ASM 磁盘组至少需要一块 ASM 磁盘，这是显然的，但如果选择常规（Normal）冗余的 ASM 磁盘组存储表决磁盘，则 ASM 磁盘组至少需要 3 块 ASM 磁盘，因为表决磁盘需要奇数冗余方式。同样的原因，如果选择高（High）冗余的 ASM 磁盘组存储表决磁盘，则 ASM 磁盘组至少需要 5 块 ASM 磁盘。

至于表决磁盘的备份，在 11gR2 之前的版本，需要使用操作系统的备份工具 dd 来完成手工备份。从该版本开始，表决磁盘已经不再需要手工备份了，表决磁盘的备份已经自动成为 OCR 备份的一部分。当 Oracle 执行对 OCR 的自动备份或手工对 OCR 执行备份时，Oracle 会自动备份表决磁盘的内容。

15.2.3　本地集群注册表（OLR）

从 11gR2 版本开始，Oracle 集群的 OCR、Voting Disk 都可以存储在 ASM 上，但这也带来一个问题，就是节点在启动时要访问 OCR、Voting Disk，ASM 磁盘组需要 ASM 实例（事实上，ASM 实例及其管理的磁盘组都是 Clusterware 管理的资源）启动后，才能访问 ASM 磁盘组。

正像前面所说，Voting Disk 是存储在 ASM 磁盘特定的位置，在没有 ASM 实例的时候，Oracle 也能够（借助工具如 kfed）访问 Voting Disk，于是解决了表决磁盘存放在 ASM 上的问题。解决 OCR 存放于 ASM 磁盘组产生的矛盾，Oracle 是使用安装在各个节点的非共享的操作系统文件（Oracle Local Registry，OLR）解决的。

当 OCR 存储于 ASM 上时，集群中的节点在启动 Clusterware 服务时访问的第一个文件就是节点本地的 OLR，由此了解该节点需要启动的节点资源。OLR 是 OCR 的本地版本，它存储与本节点有关的 OCR 资源，包括本地的 ASM 实例资源。当本地 ASM 实例启动、加载共享的磁盘组后，Oracle 就可以访问存储于共享磁盘组的 OCR 了，从此，Clusterware 就可以利用 OCR 管理集群所有节点的资源。

正因为如此，如果本地节点的 OLR 不能访问，则该节点的 Clusterware 服务是不能启动的。OLR 文件存储于 /etc/oracle/olr.loc，其默认文件位置是 $grid_home/cdata/<hostname>.olr，该文件被 Clusterware 的守护进程（ORACLE High Availability Service Daemon，OHASD）所使用。该文件中存储如下内容：

※ Clusterware 的版本。

※ Clusterware 关于集群的配置。

※ 本地节点需要启动的各种资源。

与共享的 OCR 一样，对 OLR 的管理也是使用工具 ocrconfig，但针对 OLR 需要使用 -local 参数。在集群正常运行的情况下，Oracle 会根据 OCR 的内容自动更新 OLR 中的内容，但不支持对 OLR 文件的自动备份。如果需要备份，同样可以利用 ocrconfig 工具手工执行备份（带 -local 和 -manualbackup 参数即可）。类似地，集群管理工具如 ocrcheck、ocrdump 等带 -local 参数都是针对 OLR 操作的。

15.3 构造 Clusterware 集群平台

在介绍了 Oracle 集群系统的相关概念后，本节带领大家构建一个具体的基于 Oracle 集群件 Clusterware 的集群平台。

15.3.1 主机环境概要

作为模拟 RAC 集群的生产环境，我们拟在 Linux 平台上构建一个具有两个节点的 Clusterware 集群平台，主机的构成如下。

※ 硬件平台：Intel(R) Core(TM) i5-3210M CPU@2.5GHz。

※ 操作系统：Oracle Enterprise Linux 5.10（Kernel 2.6.18-371.el5）。

※ 共享存储：Lsilogic SCSI 接口，ASM Diskgroup。

※ 双路网络：Intel(R) 82579LM Gigabit Network Connection。

※ 集 群 件：Oracle Clusterware 11gR2 (11.2.0.1.0)。

其他条件均满足 Oracle 集群环境验证工具 Cluvfy 的检查要求，如物理内存、Swap 空间、/tmp 文件系统、磁盘空间、OS 内核参数、OS 必要的 rpm 软件包、用户环境、ssh 节点间的信任关系等，在此不再赘述。在构建 Oracle 集群环境的过程中，善于利用 Cluvfy 对环境做必要的检查，会帮助我们对系统进行查漏补缺，少走弯路。下面是利用 Cluvfy 对测试环境执行检查的结果输出（从中可以看出检查的项目及其 Clusterware 的要求），可供读者参考。

```
[grid@rac1 grid]$ ./runcluvfy.sh stage -pre crsinst -n
rac1,rac2 -verbose

Performing pre-checks for cluster services setup
```

```
Checking node reachability...

Check: Node reachability from node "rac1"
  Destination Node                      Reachable?
  ------------------------------------  ------------------------
  rac2                                  yes
  rac1                                  yes
Result: Node reachability check passed from node "rac1"

Checking user equivalence...

Check: User equivalence for user "grid"
  Node Name                             Comment
  ------------------------------------  ------------------------
  rac2                                  passed
  rac1                                  passed
Result: User equivalence check passed for user "grid"

Checking node connectivity...

Checking hosts config file...
  Node Name     Status                    Comment
  ------------  ------------------------  ------------------------
  rac2          passed
  rac1          passed

Verification of the hosts config file successful

Interface information for node "rac2"
 Name    IP Address    Subnet    Gateway    Def. Gateway    HW
Address    MTU
------------------------------------------------------
 eth0     188.168.0.102     188.168.0.0        0.0.0.0
188.168.0.1     08:00:27:CC:10:61 1500
 eth1     192.168.1.102     192.168.1.0        0.0.0.0
188.168.0.1     08:00:27:BC:2B:8A 1500

Interface information for node "rac1"
```

```
 Name    IP Address    Subnet    Gateway    Def. Gateway    HW
Address    MTU
-----------------------------------------------------
 eth0    188.168.0.101    188.168.0.0       0.0.0.0
188.168.0.1    08:00:27:98:9F:BE 1500
 eth1    192.168.1.101    192.168.1.0       0.0.0.0
188.168.0.1    08:00:27:CB:38:BD 1500

Check: Node connectivity of subnet "188.168.0.0"
   Source                                        Destination
Connected?
  ------------------------  ----------------  ----------------
  rac2:eth0                 rac1:eth0         yes
Result: Node connectivity passed for subnet "188.168.0.0" with
node(s) rac2,rac1

Check: TCP connectivity of subnet "188.168.0.0"
  Source           Destination         Connected?
  -----------------------  -------------------- ----------------
  rac1:188.168.0.101       rac2:188.168.0.102  passed
Result: TCP connectivity check passed for subnet "188.168.0.0"

Check: Node connectivity of subnet "192.168.1.0"
  Source           Destination         Connected?
  -------------------  -------------------------  ------------
  rac2:eth1        rac1:eth1           yes
Result: Node connectivity passed for subnet "192.168.1.0" with
node(s) rac2,rac1

Check: TCP connectivity of subnet "192.168.1.0"
  Source           Destination         Connected?
  --------------------  ------------------------  ------------
  rac1:192.168.1.101    rac2:192.168.1.102    passed
Result: TCP connectivity check passed for subnet "192.168.1.0"


Interfaces found on subnet "188.168.0.0" that are likely
candidates for VIP are:
```

```
rac2 eth0:188.168.0.102
rac1 eth0:188.168.0.101

Interfaces found on subnet "192.168.1.0" that are likely
candidates for a private interconnect are:
rac2 eth1:192.168.1.102
rac1 eth1:192.168.1.101

Result: Node connectivity check passed

Check: Total memory
  Node Name    Available      Required             Comment
  ----------  --------------  -------------------  -------
  rac2  1.98GB (2074972.0KB)  1.5GB (1572864.0KB)  passed
  rac1  1.98GB (2074972.0KB)  1.5GB (1572864.0KB)  passed
Result: Total memory check passed

Check: Available memory
  Node Name    Available        Required           Comment
  ---------  -----------------  -----------------  ---------
  rac2   1.85GB (1942924.0KB)   50MB (51200.0KB)   passed
  rac1   1.8GB (1891596.0KB)    50MB (51200.0KB)   passed
Result: Available memory check passed

Check: Swap space
  Node Name    Available        Required               Comment
  ----------  ----------------  ---------------------  -------
  rac2   2.98GB (3120464.0KB)   2.97GB (3112458.0KB)   passed
  rac1   2.98GB (3120464.0KB)   2.97GB (3112458.0KB)   passed
Result: Swap space check passed

Check: Free disk space for "rac2:/tmp"
  Path  Node Name  Mount point  Available  Required  Comment
  ------- ---------  ---------  ---------  ---------  ------
  /tmp     rac2    /            13.26GB    1GB       passed
Result: Free disk space check passed for "rac2:/tmp"

Check: Free disk space for "rac1:/tmp"
```

```
  Path                 Node Name      Mount point    Available
Required      Comment
  -------  ---------  ---------  --------  --------  --------
  /tmp    rac1    /    13.25GB    1GB    passed
Result: Free disk space check passed for "rac1:/tmp"

Check: User existence for "grid"
  Node Name     Status                    Comment
  ---------  ----------------------  ----------------------
  rac2          exists                    passed
  rac1          exists                    passed
Result: User existence check passed for "grid"

Check: Group existence for "oinstall"
  Node Name     Status                    Comment
  ------------  ----------------------  --------------------
  rac2          exists                    passed
  rac1          exists                    passed
Result: Group existence check passed for "oinstall"

Check: Group existence for "dba"
  Node Name     Status                    Comment
------------------------------------------------------------
  rac2          exists                    passed
  rac1          exists                    passed
Result: Group existence check passed for "dba"

Check: Membership of user "grid" in group "oinstall" [as
Primary]
   Node Name    User Exists    Group Exists   User in Group
Primary       Comment
------------------------------------------------------------
  rac2      yes      yes      yes      yes      passed
  rac1      yes      yes      yes      yes      passed
Result: Membership check for user "grid" in group "oinstall" [as
Primary] passed

Check: Membership of user "grid" in group "dba"
```

```
  Node Name          User Exists   Group Exists  User in Group
Comment
------------------------------------------------------------
  rac2      yes      yes      yes      passed
  rac1      yes      yes      yes      passed
Result: Membership check for user "grid" in group "dba" passed

Check: Run level
  Node Name      run level      Required      Comment
------------------------------------------------------------
  rac2          5          3,5          passed
  rac1          5          3,5          passed
Result: Run level check passed

Check: Hard limits for "maximum open file descriptors"
  Node Name     Type     Available     Required     Comment
------------------------------------------------------------
  rac2      hard      65536      65536      passed
  rac1      hard      65536      65536      passed
Result: Hard limits check passed for "maximum open file
descriptors"

Check: Soft limits for "maximum open file descriptors"
  Node Name     Type     Available     Required     Comment
------------------------------------------------------------
  rac2          soft     1024          1024         passed
  rac1          soft     1024          1024         passed
Result: Soft limits check passed for "maximum open file
descriptors"

Check: Hard limits for "maximum user processes"
  Node Name     Type     Available     Required     Comment
  ----------  ----------  ----------  ----------  ------------
  rac2          hard     16384         16384        passed
  rac1          hard     16384         16384        passed
Result: Hard limits check passed for "maximum user processes"

Check: Soft limits for "maximum user processes"
```

```
  Node Name     Type      Available     Required     Comment
  ------------  ----------  ----------  ----------  -----------
  rac2          soft      2047          2047         passed
  rac1          soft      2047          2047         passed
Result: Soft limits check passed for "maximum user processes"

Check: System architecture
  Node Name      Available      Required      Comment
  ---------  ------------------  ------------------  --------
  rac2           i686           x86           passed
  rac1           i686           x86           passed
Result: System architecture check passed

Check: Kernel version
  Node Name    Available      Required      Comment
  ---------  ---------------  ------------  -------
  rac2        2.6.18-371.el5  2.6.18        passed
  rac1        2.6.18-371.el5  2.6.18        passed
Result: Kernel version check passed

Check: Kernel parameter for "semmsl"
  Node Name      Configured      Required      Comment
  ----------  ----------------  ------------  -------
  rac2           250             250           passed
  rac1           250             250           passed
Result: Kernel parameter check passed for "semmsl"

Check: Kernel parameter for "semmns"
  Node Name      Configured      Required      Comment
  ---------  -----------------  ------------  --------
  rac2           32000           32000         passed
  rac1           32000           32000         passed
Result: Kernel parameter check passed for "semmns"

Check: Kernel parameter for "semopm"
  Node Name      Configured      Required      Comment
  ---------  -----------------  ------------  -------
  rac2           100             100           passed
```

```
  rac1           100            100           passed
Result: Kernel parameter check passed for "semopm"

Check: Kernel parameter for "semmni"
  Node Name      Configured      Required      Comment
  ---------  ------------------  ------------------  --------
  rac2           128            128           passed
  rac1           128            128           passed
Result: Kernel parameter check passed for "semmni"

Check: Kernel parameter for "shmmax"
  Node Name      Configured      Required      Comment
  --------  --------------------  ------------------  --------
  rac2           536870912      536870912     passed
  rac1           536870912      536870912     passed
Result: Kernel parameter check passed for "shmmax"

Check: Kernel parameter for "shmmni"
  Node Name      Configured      Required      Comment
  ---------  --------------------  ------------------  -------
  rac2           4096           4096          passed
  rac1           4096           4096          passed
Result: Kernel parameter check passed for "shmmni"

Check: Kernel parameter for "shmall"
  Node Name      Configured      Required      Comment
  ---------  --------------------  ------------------  -------
  rac2           2097152        2097152       passed
  rac1           2097152        2097152       passed
Result: Kernel parameter check passed for "shmall"

Check: Kernel parameter for "file-max"
  Node Name      Configured      Required      Comment
  --------  --------------------  ------------------  -------
  rac2           6815744        6815744       passed
  rac1           6815744        6815744       passed
Result: Kernel parameter check passed for "file-max"
```

```
Check: Kernel parameter for "ip_local_port_range"
  Node Name      Configured      Required      Comment
  ---------  -------------------  -------------------  -------
  rac2   between 9000 & 65500    between 9000 & 65500   passed
  rac1   between 9000 & 65500    between 9000 & 65500   passed
Result: Kernel parameter check passed for "ip_local_port_
range"

Check: Kernel parameter for "rmem_default"
  Node Name      Configured      Required      Comment
  ---------  -------------------  -------------------  --------
  rac2           262144         262144        passed
  rac1           262144         262144        passed
Result: Kernel parameter check passed for "rmem_default"

Check: Kernel parameter for "rmem_max"
  Node Name      Configured      Required      Comment
  --------  -------------------  -------------------  -------
  rac2           4194304        4194304       passed
  rac1           4194304        4194304       passed
Result: Kernel parameter check passed for "rmem_max"

Check: Kernel parameter for "wmem_default"
  Node Name      Configured      Required      Comment
  ---------  -------------------  -------------------  -------
  rac2           262144         262144        passed
  rac1           262144         262144        passed
Result: Kernel parameter check passed for "wmem_default"

Check: Kernel parameter for "wmem_max"
  Node Name      Configured      Required      Comment
  --------  -------------------  -------------------  --------
  rac2           1048586        1048576       passed
  rac1           1048586        1048576       passed
Result: Kernel parameter check passed for "wmem_max"

Check: Kernel parameter for "aio-max-nr"
  Node Name      Configured      Required      Comment
```

```
  ---------  --------------------  -------------------  -------
  rac2           1048576        1048576       passed
  rac1           1048576        1048576       passed
Result: Kernel parameter check passed for "aio-max-nr"

Check: Package existence for "make-3.81"
  Node Name      Available        Required      Comment
  ---------  -------------------  --------------------  -------
  rac2           make-3.81-3.el5  make-3.81     passed
  rac1           make-3.81-3.el5  make-3.81     passed
Result: Package existence check passed for "make-3.81"

Check: Package existence for "binutils-2.17.50.0.6"
  Node Name      Available      Required      Comment
  ---------  -----------------  -------------------  --------
rac2  binutils-2.17.50.0.6-26.el5  binutils-2.17.50.0.6  passed
rac1  binutils-2.17.50.0.6-26.el5  binutils-2.17.50.0.6  passed
Result: Package existence check passed for
"binutils-2.17.50.0.6"

Check: Package existence for "gcc-4.1.2"
  Node Name      Available      Required      Comment
  --------  --------------------  -------------------  -------
  rac2      gcc-4.1.2-54.el5      gcc-4.1.2      passed
  rac1      gcc-4.1.2-54.el5      gcc-4.1.2      passed
Result: Package existence check passed for "gcc-4.1.2"

Check: Package existence for "gcc-c++-4.1.2"
  Node Name      Available      Required      Comment
  --------  -------------------  --------------------  -------
  rac2    gcc-c++-4.1.2-54.el5     gcc-c++-4.1.2     passed
  rac1    gcc-c++-4.1.2-54.el5     gcc-c++-4.1.2     passed
Result: Package existence check passed for "gcc-c++-4.1.2"

Check: Package existence for "libgomp-4.1.2"
  Node Name      Available      Required      Comment
  ---------  ------------------  --------------------  -------
  rac2    libgomp-4.4.7-1.el5    libgomp-4.1.2    passed
```

```
  rac1    libgomp-4.4.7-1.el5    libgomp-4.1.2    passed
Result: Package existence check passed for "libgomp-4.1.2"

Check: Package existence for "libaio-0.3.106"
  Node Name      Available      Required      Comment
  --------  ------------------  ------------------  --------
  rac2      libaio-0.3.106-5      libaio-0.3.106      passed
  rac1      libaio-0.3.106-5      libaio-0.3.106      passed
Result: Package existence check passed for "libaio-0.3.106"

Check: Package existence for "glibc-2.5-24"
  Node Name      Available      Required      Comment
  ---------  ------------------  ------------------  --------
  rac2      glibc-2.5-118      glibc-2.5-24      passed
  rac1      glibc-2.5-118      glibc-2.5-24      passed
Result: Package existence check passed for "glibc-2.5-24"

Check: Package existence for "compat-libstdc++-33-3.2.3"
  Node Name      Available      Required      Comment
  --------  -----------------  ------------------  ----------
   rac2        compat-libstdc++-33-3.2.3-61        compat-
libstdc++-33-3.2.3  passed
   rac1        compat-libstdc++-33-3.2.3-61        compat-
libstdc++-33-3.2.3  passed
Result: Package existence check passed for "compat-
libstdc++-33-3.2.3"

Check: Package existence for "elfutils-libelf-0.125"
  Node Name      Available      Required      Comment
  ----------  ----------------  ------------------  ----------
rac2  elfutils-libelf-0.137-3.el5  elfutils-libelf-0.125  passed
rac1  elfutils-libelf-0.137-3.el5  elfutils-libelf-0.125  passed
Result: Package existence check passed for "elfutils-
libelf-0.125"

Check: Package existence for "elfutils-libelf-devel-0.125"
  Node Name      Available      Required      Comment
  --------  ------------------  ------------------  ----------
```

```
   rac2   elfutils-libelf-devel-0.137-3.el5   elfutils-libelf-
devel-0.125  passed
   rac1        elfutils-libelf-devel-0.137-3.el5   elfutils-
libelf-devel-0.125  passed
Result: Package existence check passed for "elfutils-libelf-
devel-0.125"

Check: Package existence for "glibc-common-2.5"
  Node Name      Available      Required      Comment
  --------  --------------------  --------------------  -------
  rac2  glibc-common-2.5-118   glibc-common-2.5   passed
  rac1  glibc-common-2.5-118   glibc-common-2.5   passed
Result: Package existence check passed for "glibc-common-2.5"

Check: Package existence for "glibc-devel-2.5"
  Node Name      Available      Required      Comment
  --------  -------------------  ------------------  ----------
  rac2   glibc-devel-2.5-118   glibc-devel-2.5      passed
  rac1   glibc-devel-2.5-118   glibc-devel-2.5      passed
Result: Package existence check passed for "glibc-devel-2.5"

Check: Package existence for "glibc-headers-2.5"
   Node  Name       Available                          Required
Comment
  --------  --------------------  --------------------  -------
  rac2   glibc-headers-2.5-118   glibc-headers-2.5    passed
  rac1   glibc-headers-2.5-118   glibc-headers-2.5    passed
Result: Package existence check passed for "glibc-headers-2.5"

Check: Package existence for "libaio-devel-0.3.106"
  Node Name      Available      Required      Comment
  --------  ------------------  --------------------  ---------
rac2  libaio-devel-0.3.106-5  libaio-devel-0.3.106   passed
rac1  libaio-devel-0.3.106-5  libaio-devel-0.3.106   passed
Result: Package existence check passed for "libaio-
devel-0.3.106"

Check: Package existence for "libgcc-4.1.2"
```

```
  Node Name      Available      Required      Comment
  --------  --------------------  --------------------  ------
  rac2      libgcc-4.1.2-54.el5      libgcc-4.1.2      passed
  rac1      libgcc-4.1.2-54.el5      libgcc-4.1.2      passed
Result: Package existence check passed for "libgcc-4.1.2"

Check: Package existence for "libstdc++-4.1.2"
  Node Name      Available      Required      Comment
  ---------  ------------------  ------------------  ---------
  rac2   libstdc++-4.1.2-54.el5   libstdc++-4.1.2   passed
  rac1   libstdc++-4.1.2-54.el5   libstdc++-4.1.2   passed
Result: Package existence check passed for "libstdc++-4.1.2"

Check: Package existence for "libstdc++-devel-4.1.2"
  Node Name      Available      Required      Comment
  --------  ----------------  ------------------  ----------
rac2  libstdc++-devel-4.1.2-54.el5  libstdc++-devel-4.1.2     passed
rac1  libstdc++-devel-4.1.2-54.el5  libstdc++-devel-4.1.2  passed
Result: Package existence check passed for "libstdc++-
devel-4.1.2"

Check: Package existence for "sysstat-7.0.2"
  Node Name      Available      Required      Comment
  ----------  ------------------  ------------------  -------
  rac2   sysstat-7.0.2-12.0.1.el5   sysstat-7.0.2   passed
  rac1   sysstat-7.0.2-12.0.1.el5   sysstat-7.0.2   passed
Result: Package existence check passed for "sysstat-7.0.2"

Check: Package existence for "unixODBC-2.2.11"
  Node Name      Available      Required      Comment
  --------  -------------------  ------------------  ---------
  rac2   unixODBC-2.2.11-10.el5   unixODBC-2.2.11      passed
  rac1   unixODBC-2.2.11-10.el5   unixODBC-2.2.11      passed
Result: Package existence check passed for "unixODBC-2.2.11"

Check: Package existence for "unixODBC-devel-2.2.11"
  Node Name      Available      Required      Comment
  --------  ------------------  ------------------ ----------
```

```
rac2  unixODBC-devel-2.2.11-10.el5  unixODBC-devel-2.2.11  passed
rac1  unixODBC-devel-2.2.11-10.el5  unixODBC-devel-2.2.11  passed
Result: Package existence check passed for "unixODBC-
devel-2.2.11"

Check: Package existence for "ksh-20060214"
  Node Name      Available      Required      Comment
  ---------  -----------------  ------------------  ---------
  rac2      ksh-20100621-18.el5      ksh-20060214      passed
  rac1      ksh-20100621-18.el5      ksh-20060214      passed
Result: Package existence check passed for "ksh-20060214"

Checking for multiple users with UID value 0
Result: Check for multiple users with UID value 0 passed

Check: Current group ID
Result: Current group ID check passed
Checking Core file name pattern consistency...
Core file name pattern consistency check passed.

Checking to make sure user "grid" is not in "root" group
  Node Name     Status                    Comment
  ------------  ------------------------  --------------------
  rac2          does not exist            passed
  rac1          does not exist            passed
Result: User "grid" is not part of "root" group. Check passed

Check default user file creation mask
  Node Name      Available      Required      Comment
  ---------  -------------------  ------------------  -------
  rac2          0022            0022          passed
  rac1          0022            0022          passed
Result: Default user file creation mask check passed

Starting Clock synchronization checks using Network Time
Protocol(NTP)...

NTP Configuration file check started...
```

```
Network Time Protocol(NTP) configuration file not found on any of the
nodes. Oracle Cluster Time Synchronization Service(CTSS) can be
used instead of NTP for time synchronization on the cluster nodes

Result: Clock synchronization check using Network Time
Protocol(NTP) passed

Pre-check for cluster services setup was successful.
```

15.3.2 ASMLib 驱动与 ASM 磁盘

拟构造的集群环境使用 ASM 作为共享存储。ASM 使用磁盘组来存储数据，而磁盘组是由 ASM 磁盘构成的。使用 ASM 磁盘需要专门的驱动，在 Linux 平台下，这个驱动就是 ASMLib。使用 ASMLib 需要安装与操作系统内核完全一致的 rpm 软件包。对于测试环境，使用的主机操作系统内核是 Kernel 2.6.18-371.el5，对应地，需要如下 3 个 rpm 软件包：

※ oracleasm-support-2.1.8-1.el5.i386.rpm——管理工具包。

※ oracleasmlib-2.0.4-1.el5.i386.rpm——库文件包。

※ oracleasm-2.6.18-371.el5-2.0.5-1.el5.i686.rpm——内核文件包。

下面是安装配置过程示例。该过程需要 3 个步骤：安装 ASMLib、配置 ASM 驱动、创建 ASM 磁盘，安装和配置需要在每一台主机上完成，创建 ASM 磁盘仅需在一台主机上执行即可，因为磁盘是共享的。

（1）安装 ASMLib，代码如下：

```
[root@rac1 share]# rpm -ivh oracleasm-support-2.1.8-1.el5.
i386.rpm
warning: oracleasm-support-2.1.8-1.el5.i386.rpm: Header V3 DSA
signature: NOKEY, key ID 1e5e0159
Preparing...
########################################### [100%]
   1:oracleasm-support
########################################### [100%]
[root@rac1 share]# rpm -ivh oracleasmlib-2.0.4-1.el5.i386.rpm
warning: oracleasmlib-2.0.4-1.el5.i386.rpm: Header V3 DSA
signature: NOKEY, key ID 1e5e0159
Preparing...
########################################### [100%]
```

```
   1:oracleasmlib
########################################################## [100%]
```

（2）配置 ASM 驱动。

```
[root@rac1 share]#
# rpm -ivh oracleasm-2.6.18-371.el5-2.0.5-1.el5.i686.rpm
warning: oracleasm-2.6.18-371.el5-2.0.5-1.el5.i686.rpm: Header
V3 DSA signature: NOKEY, key ID 1e5e0159
Preparing...
########################################################## [100%]
   1:oracleasm-2.6.18-371
########################################################## [100%]

[root@rac1 share]# /etc/init.d/oracleasm configure
Configuring the Oracle ASM library driver.

This will configure the on-boot properties of the Oracle ASM
library
driver.  The following questions will determine whether the
driver is
loaded on boot and what permissions it will have.  The current
values
will be shown in brackets ('[]').  Hitting <ENTER> without
typing an
answer will keep that current value.  Ctrl-C will abort.

Default user to own the driver interface [grid]:
Default group to own the driver interface [asmadmin]:
Start Oracle ASM library driver on boot (y/n) [y]:
Scan for Oracle ASM disks on boot (y/n) [y]:
Writing Oracle ASM library driver configuration: done
Initializing the Oracle ASMLib driver:
[  OK  ]
Scanning the system for Oracle ASMLib disks:
[  OK  ]
```

（3）创建 ASM 磁盘。

```
[root@rac1 ~]# /etc/init.d/oracleasm createdisk asmdsk1 /dev/
sdb1
```

```
Marking disk "asmdsk1" as an ASM disk:
[  OK  ]
[root@rac1 ~]# /etc/init.d/oracleasm scandisks
Scanning the system for Oracle ASMLib disks:               [
OK  ]
[root@rac1 ~]# /etc/init.d/oracleasm listdisks
ASMDSK1
```

15.3.3 网格基础架构（Grid Infrastructure）

通过前面运行的 runcluvfy.sh stage -pre crsinst -n rac1,rac2 检查后，即可在其中的一台主机上以 grid 用户身份运行 runInstaller（位于安装介质根目录下）启动 Grid Infrastructure 的安装过程，在此从略。此处列出后续检查情况。

（1）检查集群服务在各节点的运行情况，代码如下：

```
[root@rac1 ~]# su - grid
[grid@rac1 ~]$ crsctl check crs
CRS-4638: Oracle High Availability Services is online
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
[grid@rac1 ~]$ crsctl check cluster -all
**************************************************************
rac1:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
rac2:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
```

（2）检查 OCR 和 Voting Disk，代码如下：

```
[grid@rac1 ~]$ ocrcheck
Status of Oracle Cluster Registry is as follows :
         Version                  :          3
```

```
         Total space (kbytes)     :     262120
         Used space (kbytes)      :       2596
         Available space (kbytes) :     259524
         ID                       :  622159630
         Device/File Name         :    +OCRGRP
                                        Device/File integrity
check succeeded
                                    Device/File not configured
                                    Device/File not configured
                                    Device/File not configured
                                    Device/File not configured
         Cluster registry integrity check succeeded
            Logical corruption check bypassed due to non-
privileged user

[root@rac1 bin]# ./ocrcheck -local
Status of Oracle Local Registry is as follows :
         Version                  :          3
         Total space (kbytes)     :     262120
         Used space (kbytes)      :       2068
         Available space (kbytes) :     260052
         ID                       :  607715664
          Device/File Name          : /oracle/grid/cdata/rac1.
olr
                                          Device/File integrity
check succeeded
         Local registry integrity check succeeded
         Logical corruption check succeeded

[grid@rac1 ~]$ crsctl query css votedisk
##  STATE    File Universal Id           File Name Disk group
--  -----    -----------------           --------- ---------
 1. ONLINE  6d76ab7a146a4fabbf7661c40fabcb3a  (ORCL:ASMDSK1)
[OCRGRP]
Located 1 voting disk(s).
```

此处仅将 OCR 和 Voting Disk 放置于外部（External）冗余的 ASM 组 OCRGRP 上，因此，OCR 和 Voting Disk 都没有镜像。接下来可以利用集群件工具 crsctl、ocrconfig 等将其转移至具有常规冗余或高冗余的 ASM 磁盘组上，以实现对 OCR 和 Voting Disk 的镜像保护。

（3）检查集群资源的运行情况。这里的资源是实现上层应用（如 RAC 数据库）的基础。结合测试环境，必要的 Clusterware 资源列举如下：

※ 虚拟 IP 资源：

```
ora.scan1.vip
ora.rac1.vip
ora.rac2.vip
```

※ 网络与监听资源：

```
ora.LISTENER.lsnr
ora.rac1.LISTENER_RAC1.lsnr
ora.rac2.LISTENER_RAC2.lsnr
ora.net1.network
ora.LISTENER_SCAN1.lsnr
```

※ 节点应用资源：

```
ora.ons
ora.eons
ora.rac1.ons
ora.rac2.ons
```

※ ASM 与磁盘组资源：

```
ora.asm
ora.rac1.ASM1.asm
ora.rac2.ASM2.asm
ora.OCRGRP.dg
```

```
[grid@rac1 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora....ER.lsnr     ora....er.type      ONLINE    ONLINE    rac1
ora....N1.lsnr     ora....er.type      ONLINE    ONLINE    rac2
ora.OCRGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
```

```
ora.asm              ora.asm.type          ONLINE    ONLINE    rac1
ora.eons             ora.eons.type         ONLINE    ONLINE    rac1
ora.gsd              ora.gsd.type          OFFLINE   OFFLINE
ora....network       ora....rk.type        ONLINE    ONLINE    rac1
ora.oc4j             ora.oc4j.type         OFFLINE   OFFLINE
ora.ons              ora.ons.type          ONLINE    ONLINE    rac1
ora....SM1.asm       application           ONLINE    ONLINE    rac1
ora....C1.lsnr       application           ONLINE    ONLINE    rac1
ora.rac1.gsd         application           OFFLINE   OFFLINE
ora.rac1.ons         application           ONLINE    ONLINE    rac1
ora.rac1.vip         ora....t1.type        ONLINE    ONLINE    rac1
ora....SM2.asm       application           ONLINE    ONLINE    rac2
ora....C2.lsnr       application           ONLINE    ONLINE    rac2
ora.rac2.gsd         application           OFFLINE   OFFLINE
ora.rac2.ons         application           ONLINE    ONLINE    rac2
ora.rac2.vip         ora....t1.type        ONLINE    ONLINE    rac2
ora.scan1.vip        ora....ip.type        ONLINE    ONLINE    rac2
```

另外，还可以使用 crsctl stat res -t、crsctl stat res -init -t 查看资源运行情况，也可使用 srvctl 对 Clusterware 管理的资源进行配置和管理。

这里有两类资源：gsd 和 oc4j，在测试环境中不需要，默认处于 offline 状态。

```
ora.gsd  is OFFLINE by default if there is no 9i database in
the cluster.
ora.oc4j is OFFLINE in 11.2.0.1 as Database Workload
Management(DBWLM) is unavailable. These can be ignored in
11gR2 RAC.
```

15.4　创建 RAC 集群数据库

在由集群件 Clusterware（包含于 Grid Infrastracture）构建的集群平台基础上可以构建 RAC 集群数据库。RAC 数据库必须有集群件 Clusterware 的支持，即使使用了第三方的集群件平台。

15.4.1　选择 DBMS 软件

虽然 Oracle 的 Clusterware 是一款通用的集群件产品，不仅可以构建数据库集群，同时也可以构建 Web 集群、应用服务器集群等，但 Oracle 数据库管理系统软件与集群件软件高度集成，两者在软件发行的版本上也高度一致，因此，为了最大限度地利用 Oracle 一体化产品的便利，在构建 RAC 集群数据库时，建议选择

与 Clusterware 版本一致的数据库系统软件。当然这不排除在某些情况下利用同一版本的 Clusterware 构建具有不同版本的 RAC 数据库。

在安装 Oracle DBMS 软件之前，可以利用 cluvfy 工具对 DBMS 运行的集群环境进行检查，以确认满足 RAC 数据库的需求。

以 Oracle 用户身份运行下列指令：

```
$ ./runcluvfy.sh stage -pre dbinst -n rac1,rac2 -verbose
...
Pre-check for database installation was successful.
```

接下来，在集群的一个节点上以 Oracle 用户身份运行如下指令启动 OUI 安装 Oracle DBMS 软件即可。

```
$ ./runInstaller
...
```

启动交互式安装过程后，有几个重要的选项需要选择，建议如下：

（1）选择 Install database software only。

（2）选择 Oracle Real Application Cluster Database Installation。

最后，根据安装程序 OUI 的提示分别在两个节点上以 root 用户身份运行 root.sh 脚本，在节点 rac1 上运行的结果示例如下：

```
[root@rac1 ~]# /oracle/dbms/db_1/root.sh
Running Oracle 11g root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /oracle/dbms/db_1

Enter the full pathname of the local bin directory: [/usr/
local/bin]:
The file "dbhome" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying dbhome to /usr/local/bin ...
The file "oraenv" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying oraenv to /usr/local/bin ...
The file "coraenv" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
```

```
    Copying coraenv to /usr/local/bin ...

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created.
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
Finished product-specific root actions.
```

15.4.2　集群数据库的特有设置

集群数据库与传统的单实例数据库相比，本质上并没有什么不同，只是运行方式（实例）有所不同。单实例数据库运行过程中，实例以独占方式访问数据库，而集群数据库的物理存储被多个实例共享，运行过程中多个实例并发地访问数据库的物理存储。

从用户的角度，创建集群数据库是从创建传统的单实例数据库开始的，然后根据集群的需要调整必要的数据库设置和实例设置。

与单实例数据库相比，集群数据库必要的数据库设置有如下几个方面：

（1）集群数据库需要额外的数据字典的支持，因此，在创建数据库的过程中，需要额外运行一个 PL/SQL 脚本创建此部分的数据字典。

```
SQL> @$ORACLE_HOME/RDBMS/ADMIN/catclust.sql
```

（2）集群数据库运行的每个实例需要有自己的回滚表空间（UNDO_TABLESPACE）。一个具有 3 个节点的 RAC 数据库，在创建数据库时需要分别创建 3 个不同的回滚表空间，而单实例数据库仅需一个即可。

（3）集群数据库运行的每个实例需要有自己的日志线程（Thread）及其日志组（不少于两个组）。一个具有 3 节点的 RAC 数据库，需要 3 个日志线程，其日志组的数量不少于 6 个组。

（4）Oracle 数据库的控制文件中有一个参数限制 MAXINSTANCES，该参数限制数据库最大运行的实例数，很显然，集群数据库控制文件中的参数必须大于将来运行的实例数。如果将该参数设置为 1，则必须重新创建控制文件，以便修改该参数的设置值。

与单实例数据库相比，集群数据库必要的初始化参数的设置有如下几个方面：

（1）集群数据库参数 CLUSTER_DATABASE 的默认值为 false，该参数必须调整为 true，指示实例打开的是一个集群数据库。需要注意的是，在集群数据库执行转换日志归档模式（如由非归档调整为归档）、启动数据库闪回、执行表空间的介质恢复、数据库升级等操作时需要将该参数重新设置为 false。

（2）集群数据库运行的所有实例必须有唯一的实例名和对应的实例编号，实例名 INSTANCE_NAME 及其实例号 INSTANCE_NUMBER。

（3）集群中的每个实例必须有自己的日志线程（Thread），单实例中该参数默认为 1，不同的实例必须有自己的线程号及其对应的联机日志组。

（4）集群数据库有多个回滚表空间，不同的实例通过参数 UNDO_TABLESPACE 设置使用不同的回滚表空间。

（5）集群中的每个实例必须设置远程监听器（REMOTE_LISTENER），即每个运行的数据库实例必须动态地向远程监听器注册。从 11gR2 开始，远程监听器参数需要指向 SCAN 监听器。集群的 SCAN 地址与 SCAN 监听器是对应的，如有 3 个 SCAN 地址，则集群中就应该有 3 个 SCAN 监听器，SCAN 监听器接收客户端通过 SCAN 地址发来的集群数据库连接请求，并将该连接请求转向集群中的某个本地监听，从而实现负载均衡。

（6）与单实例数据库相比，集群数据库的运行其 SGA 需要增加缓存融合的开销，一般情况下这个缓存融合的开销是单实例 SGA 的 5% 左右。因此，集群数据库实例的 SGA 设置要适当扩大。

（7）设置用于执行实例间缓存融合的后台进程 LMS（锁管理服务器）数量的参数 GCS_SERVER_PROCESSES。该参数的默认值与系统的 CPU 数量有关，1 ~ 3 个 CPU，其默认值为 1，4 ~ 16 个 CPU，其默认值为 2，该参数可设置的范围是 1 ~ 20。该参数的设置值需要在所有的实例中保持相同，以获得均衡的缓存融合效率。

15.4.3 RAC 数据库的形成过程

下面介绍 RAC 数据库的主要创建过程。

（1）在集群的一个节点上创建一个单实例数据库，包括设置环境变量、创建口令文件、初始化参数文件、所有的数据库文件等，并将数据库、Oracle 主目录等信息以如下方式加入本地的 /etc/oratab 文件：

```
ORCL:/oracle/product/11.2.0/db_1:Y
```

此处的 ORCL 是数据库名称，/oracle/product/11.2.0/db_1 是 DBMS 软件安装的 $ORACLE_HOME 主目录。

创建 ASM 存储的数据库，并运行必要的脚本，代码如下：

```
SQL> startup nomount pfile = 'local_init_parameter_file';
SQL> create database ORCL
datafile '+DATGRP' size 100m autoextend on
sysaux datafile '+DATGRP' size 50m autoextend on
logfile
```

```
group 1 ('+LOGGRP') size 50m, group 2 ('+LOGGRP') size 50m
character set zhs16gbk;

SQL> @ $ORACLE_HOME/rdbms/admin/catalog.sql
SQL> @ $ORACLE_HOME/rdbms/admin/catproc.sql
SQL> @ $ORACLE_HOME/rdbms/admin/catclust.sql
```

（2）创建必要的表空间。按照以上方式创建的数据库仅包含两个系统表空间（SYSTEM 和 SYSAUX），在此基础上创建必要的表空间，如默认的临时表空间、默认的用户表空间、每个实例对应的回滚表空间，并做必要的设置。示例如下：

```
SQL> create temporary tablespace TEMPTBS
tempfile '+DATGRP' size 50m autoextend on;
SQL> alter database default temporary tablespace TEMPTBS;

SQL> create tablespace USERDATA
datafile '+DATGRP' size 50m autoextend on;
SQL> alter database default tablespace USERDATA;

SQL> create tablespace UNDOTBS1
datafile '+DATGRP' size 50m autoextend on;
SQL> create tablespace UNDOTBS2
datafile '+DATGRP' size 50m autoextend on;
```

（3）为多实例创建必要的日志线程和联机日志组。

为第二个实例创建日志组，并启动日志线程，示例如下：

```
SQL> alter database add logfile thread 2
group 3 ('+LOGGRP') size 50m, group 4 ('+LOGGRP') size 50m;
SQL> alter database enable thread 2;
```

（4）调整必要的初始化参数设置，为集群数据库的启动做准备。

根据前面的介绍，数据库名称为 ORCL，集群启动的两个实例分别为 ORCL1、ORCL2，调整的参数示例如下：

```
*.db_name=ORCL
*.cluster_database=true
ORCL1.instance_number=1
ORCL2.instance_number=2
ORCL1.thread=1
ORCL2.thread=2
```

```
ORCL1.undo_management=auto
ORCL2.undo_management=auto
ORCL1.undo_tablespace=UNDOTBS1
ORCL2.undo_tablespace=UNDOTBS2
remote_listener=win-scan:1521
......
```

（5）在另一节点创建与启动第二个实例。

在节点二创建第二个实例，包括设置环境变量、创建口令文件、初始化参数文件，并将数据库、ORACLE 主目录等信息以如下方式加入本地的 /etc/oratab 文件：

```
ORCL:/oracle/product/11.2.0/db_1:Y
```

备注：节点二的口令文件与初始化参数文件都可以从节点一复制而来，注意变更口令文件的默认文件名。

在第一个节点的实例 1 启动的情况下，尝试启动第二个节点的实例 2，若启动成功，则说明集群数据库配置正确。

（6）创建共享的初始化参数文件 SPFILE。

集群数据库运行在多个节点上，为了统一控制多个数据库实例，通常将使用统一的初始化参数文件，并将其放置于共享存储中。

在其中的一个节点根据本地的 pfile 创建生成 spfile，并将其放置在共享的 ASM 磁盘组中。示例如下：

```
SQL> create spfile='+DATGRP' from pfile='local_init_parameter_
file';
```

在 ASMCMD 中确认生成的 spfile 路径及其文件名，记录下来以供后续的注册使用。

集群环境使用共享 spfile 的方法如下。

※ 方法一：在本地创建默认的 pfile 文件 initsid.ora，其中仅设置 spfile=spfile_shared 一项。

※ 方法二：删除本地默认的参数文件 pfile（initsid.ora）和 spfile（spfilesid.ora），Oracle 会使用注册数据库中的 spfile，使用 srvctl config database -d ORCL 查看。

（7）向集群中注册数据库及其实例。

将数据库及其实例作为集群资源向集群注册，以接受集群件 Clusterware 的管理。本质上这一步的目的是将数据库及其实例的信息及其配置写入集群注册表

OCR，数据库和实例就是集群管理的高可用性资源。

使用服务器控制器 srvctl 注册、配置数据库及其实例。示例如下：

```
[root@rac1 ~]# su - oracle
[oracle@rac1 ~]$
$ srvctl add database -d ORCL -o /oracle/product/11.2.0/db_1
-p +datgrp/orcl/parameterfile/spfile.263.839102915
$ srvctl add instance -d ORCL -i ORCL1 -n rac1
$ srvctl add instance -d ORCL -i ORCL2 -n rac2
$ srvctl enable database -d ORCL
$ srvctl start database -d ORCL

[oracle@rac1 ~]$ srvctl config database -d ORCL
     Database unique name: ORCL
     Database name:
     Oracle home: /oracle/dbms/db_1
     Oracle user: oracle
     Spfile: +datgrp/orcl/parameterfile/spfile.263.839102915
     Domain:
     Start options: open
     Stop options: immediate
     Database role: PRIMARY
     Management policy: AUTOMATIC
     Server pools: ORCL
     Database instances: ORCL1,ORCL2
     Disk Groups:
     Services:
     Database is administrator managed

[oracle@rac1 ~]$ srvctl status database -d ORCL
Instance ORCL1 is running on node rac1
Instance ORCL2 is running on node rac2
```

15.5　RAC 的缓存融合技术

RAC 数据库与单实例数据库相比，它的核心技术就是缓存融合（Cache Fusion），它通过集群中的互连网络（Private Net）在不同节点的实例间交换数据，使得 RAC 数据库在运行期间避免了一个实例必须将数据写回磁盘再由另外一个实例读取的磁盘 I/O 过程，数据直接在不同的 SGA 之间传输，形成逻辑上的集群 SGA。

缓存融合技术是通过如下两个方面实现的，一是 SGA 中增加的 GRD（Global Resource Directory），即全局资源目录，这是 RAC 数据库实例 SGA 中特有的内存区域；二是增加了两类集群服务：全局缓存服务（Global Cache Service，GCS）和全局队列服务（Global Enqueue Service，GES），当然，这两类集群服务都是通过 RAC 数据库实例所额外增加的后台进程实现的。

15.5.1 全局资源目录（GRD）

RAC 集群中的全局资源有两类：缓存资源（即缓存中的数据块资源）和队列资源（即全局锁资源）。

GRD 负责记录缓存资源和队列资源在集群范围内的分布及其状态信息，当一个缓存数据块由一个实例传输到另外一个实例时，GRD 的内容会发生更新。一个节点加入或离开集群都会导致 GRD 被重建。节点关系的任何改动都将导致集群信息的重构。GRD 是一个分布式的结构，每个实例 SGA 的 GRD 区域仅存储与本实例有关的 GRD 信息。因此，GRD 相当于一个分布式的内部共享数据库，这个共享数据库的信息对于集群中的所有实例都是可访问的，如果这个信息是在当前实例中，显然可以直接访问；如果存在于其他节点上的实例中，可以通过和远程节点的后台进程通信来获取需要的信息。

15.5.2 GCS 和 GES

RAC 的缓存融合是通过 GCS 和 GES 实现的，两类集群服务分别用于实现对全局缓存资源和全局队列资源的维护和管理，包括资源在不同节点间传输、资源在集群范围内的重构等。

GCS 是针对缓存数据块的，它负责维护数据块在全局缓存中的数据一致性。实施 GCS 的关键后台进程是 LMS（Lock Manager Server）。

GES 是针对字典缓存和库缓存的，它负责维护非缓存融合资源（即全局队列资源）在整个集群范围内的协调。实施 GES 需要多个后台进程，如 LCK（Lock Process）、LMD（Lock Manager Daemon）、LMON（Lock Monitor）的协同作业。

这里讲的锁（Lock），指的是全局资源锁，它不同于在单实例环境下由于事务处理产生的 TX 锁、TM 锁，它负责维护资源在整个集群范围内的数据一致性。

为了使得集群中的不同实例都能访问同一个数据块，一个数据块不得不在不同的实例中维护多个不同版本的副本，这些副本处于两种基本的模式：要么处于当前（Cur）模式，要么处于读一致（CR）模式，但只有一个实例拥有这个数据块的 Exclusive Current 模式（XCUR），只有拥有这个模式的实例才可以将数据写入磁盘。一个缓存块在实例中所处的模式可以由视图 V$BH 中的 Status 字段指示。当一个实例试图修改某个数据块时，必须获得该数据块上的一种全局锁资源，从而避免其他实例同时修改该数据块的可能性。因此，每个缓存块都有对应的锁资源

队列存在，拥有这个锁资源队列的实例只有一个，这个实例就是这个缓存块资源的 Master 实例或 Master 节点。当一个实例需要修改一个缓存块中的内容时，必须访问该缓存块的 Master 节点，以排队获得这个缓存块上的锁资源。一个缓存块的 Master 节点在某个时间点是确定的，但在集群运行过程中是动态的。集群中用户的访问、一个节点的加入或离开、节点配置的变更、节点故障等都可能导致 Master 节点的动态调整（Dynamic Resource Remastering）。

要获得一个缓存块的 Master 节点，可以执行如下查询：

```
select b.dbablk, r.kjblmaster master_node
from x$lel, x$kjblr, x$bhb
where b.obj = <DataObjectId>
and b.le_addr= l.le_addr
and l.le_kjbl= r.kjbllockp;
```

15.5.3　RAC 后台进程

根据前面对缓存融合的介绍，下面总结一下 RAC 数据库实例所特有的主要后台进程，正是这些后台进程的作用，实现了 RAC 数据库的关键功能。这些主要的后台进程都与全局锁（Lock）有关，当然，这里的锁指的是全局资源锁。

GRD 配合 GCS 和 GES，三者共同完成 RAC 的缓存融合任务，GCS 负责多实例间的内容（缓存块）的协调，GES 负责多实例间的秩序（全局锁）的协调，而 GRD 就是这两者协调的信息中心。Oracle 通过下面的后台进程完成 GCS 和 GES。

LMS 进程：即 Lock Manager Server 进程，又称 Global Cache Service 进程，它是实现 GCS 的关键进程。LMS 进程负责缓存块的内容在集群的不同实例间的传输与控制，负责协调一个缓存块在不同实例中的副本及其状态，即该进程负责缓存融合资源的管理与协调。LMS 进程同时负责更新与维护 GRD 中的内容。初始化参数 gcs_server_processes 设置实例中启动的 LMS 进程数量。

LCK 进程：即 Lock Process 进程，又称 Instance Enqueue Process，它是实现 GES 的主要进程。LCK 进程负责非缓存融合（non-Cache Fusion）资源的请求，如库缓存内容的实例间请求、数据字典内容的实例间请求。每个实例只需要一个 LCK 进程负责全局队列的协调。

LMD 进程：即 Lock Manager Daemon 进程，又称 Global Enqueue Service Daemon，它在实例中负责管理来自远程节点的资源请求，同时，还负责监测全局队列中可能出现的死锁、监控全局队列中的锁状态及其状态转换。

LMON 进程：即 Lock Monitor 进程，又称 Global Enqueue Service Monitor 进程，它负责监控集群中的全局资源与队列，并在实例加入或退出集群时负责全局资源与队列的重新配置。

第16章

RAC环境下的Data Guard

通过前面的章节，我们对Oracle Dataguard系统和RAC系统有了一个比较全面的了解。事实上，Dataguard和RAC是Oracle数据库最大可用性架构（Maximum Availability Architecture，MAA）的两个重要组成部分，两者分别从数据库和实例的角度为服务器提供了冗余，避免了系统因数据库或实例出现的单点故障（Single Points of Failure），由此增强系统的可用性（Availability）和可扩展性（Scalability），并提高系统的吞吐量（Throughput）。

本章更进一步，将Dataguard和RAC这两种高可用解决方案结合起来，取长补短，充分利用这两种解决方案的优势，为数据库服务器提供更高级别的可用性、可扩展性及其吞吐量，由此极大地提高服务器系统的容错、容灾、防灾能力，有效避免因人为错误、系统故障、不可抗因素等给服务器带来的风险。

16.1 RAC主机环境描述

拟构建的高可用系统是基于RAC的Dataguard环境，主站点（Primary）和备用站点（Standby）都采用RAC架构。至于节点的数量，主备站点都采用两节点，因为更多节点并不会给主备用数据库运行和管理带来本质上的变化。

16.1.1 基于RAC的Dataguard框架

由于如下两个主要原因，在此仅以物理备用数据库为例说明：①在构建Dataguard环境过程中，逻辑备用数据库是由物理备用数据库转换而来的；②从11g版本开始，物理备用数据库可以以只读打开方式运行（不停止Redo Apply），即Active Dataguard，工程应用中物理备用数据库的价值大大增加，在很多场合省去了创建逻辑备用数据库的必要。

基于RAC的Dataguard框架结构如图16-1所示。

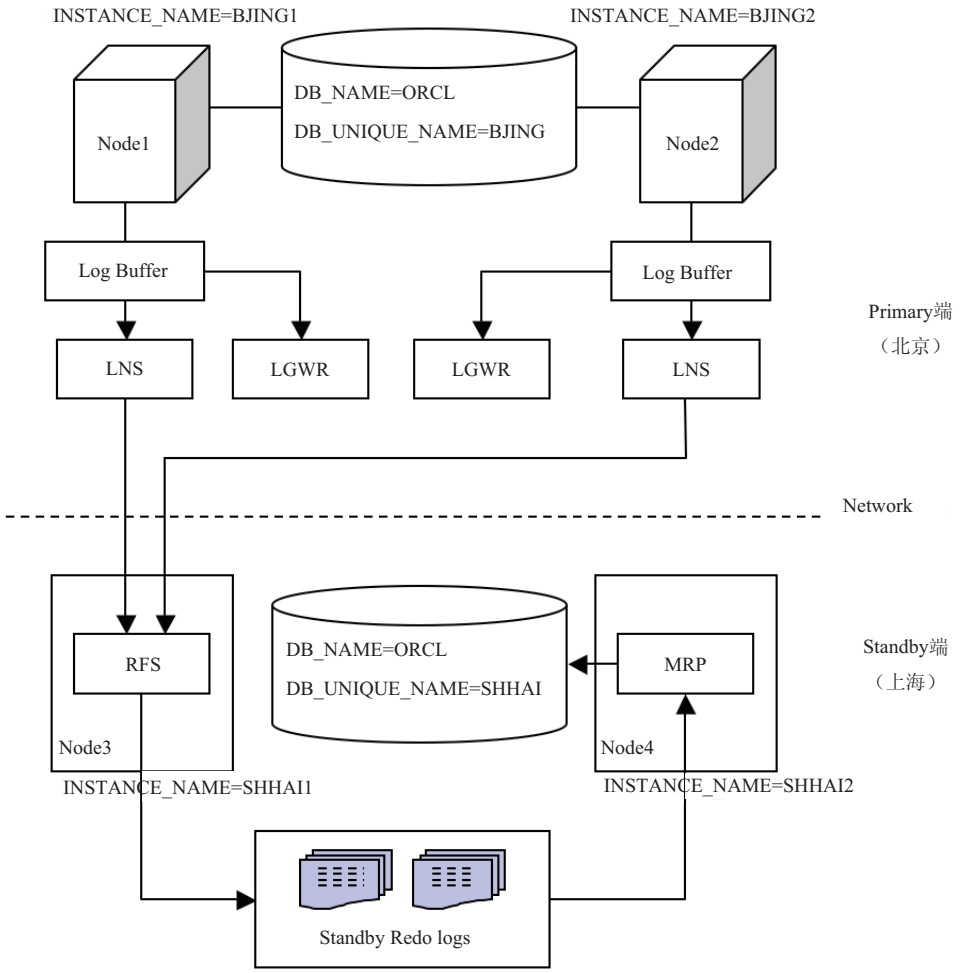


图 16-1　基于 RAC 的 Dataguard 框架结构

16.1.2　主备数据库及其实例配置

主备数据库及其实例配置如表 16-1 所示。

表 16-1　主备数据库及其实例配置

配置项目	Primary 端	Standby 端
Clusterware	Grid Infrastructure 11g R2 (11.2.0.1.0)	Grid Infrastructure 11g R2 (11.2.0.1.0)
Cluster Nodes	rac1, rac2	rac3, rac4

续表

配置项目	Primary 端	Standby 端
GRID_HOME	/oracle/grid	/oracle/grid
SCAN	bjing-scan	shhai-scan
SCAN Listener	SCAN VIPs, Port 1535	SCAN VIPs, Port 1535
VIPs	rac1-vip, rac2-vip	rac3-vip, rac4-vip
ASM Instance	+ASM1, +ASM2	+ASM1, +ASM2
Diskgroup for GI	+OCRGRP	+OCRGRP
DB_NAME	ORCL	ORCL
DB_UNIQUE_NAME	BJING	SHHAI
INSTANCE_NAME	BJING1, BJING2	SHHAI1, SHHAI2
DB Listener	rac1-vip, rac2-vip (port 1521)	rac3-vip, rac4-vip (port 1521)
DB Storage	ASM	ASM
File Management	Oracle Managed File	Oracle Managed File
Diskgroup for DB	+DATGRP	+DATGRP
Diskgroup for FRA	+FRAGRP	+FRAGRP
ORACLE_HOME	/oracle/dbms/db_1	/oracle/dbms/db_1
DBMS version	11g R2 (11.2.0.0.1)	11g R2 (11.2.0.0.1)
Operation System	Oracle Enterprise Linux 5.10 (32 bit)	Oracle Enterprise Linux 5.10 (32 bit)

16.1.3 建立 Dataguard 前的基础条件

建立一个 RAC 对 RAC 的 Dataguard 环境，涉及 Primary 端和 Standby 端的软硬件平台较复杂。为了简化叙述，假设如下基础条件已经具备：

1. Primary 端

※ 两节点的 Grid Infrastructure 集群环境已经安装配置完成。

※ 共享存储：ASM 实例及其磁盘组 OCRGRP、DATGRP、FRAGRP。

※ 两节点的 DBMS 集群环境已经安装完成。

※ RAC 数据库 ORCL 已经创建，两个实例 BJING1、BJING2 分别运行于两节点。

※ 节点的 Local Listener 和集群的 SCAN Listener 已经配置。

※ 主数据库端的主机地址信息如下：

```
#######################################
# public ip & host name
188.168.0.101 host1
188.168.0.102 host2

# private ip & host name
192.168.1.101 host1-priv
192.168.1.102 host2-priv

# virtual ip & host name
188.168.0.111 host1-vip
188.168.0.112 host2-vip

# scan vip & scan name
188.168.0.121 bjing-scan
188.168.0.122 bjing-scan
188.168.0.123 bjing-scan
#######################################
```

2. Standby 端

※ 两节点的 Grid Infrastructure 集群环境已经安装配置完成。

※ 共享存储：ASM 实例及其磁盘组 OCRGRP、DATGRP、FRAGRP。

※ 两节点的 DBMS 集群环境已经安装完成。

※ 节点的 Local Listener 和集群的 SCAN Listener 已经配置。

※ 备用端的主机地址信息如下：

```
#######################################
# public ip & host name
188.168.0.103 host3
188.168.0.104 host4

# private ip & host name
192.168.1.103 host3-priv
192.168.1.104 host4-priv

# virtual ip & host name
188.168.0.113 host3-vip
```

```
188.168.0.114 host4-vip

# scan vip & scan name
188.168.0.124 shhai-scan
188.168.0.125 shhai-scan
188.168.0.126 shhai-scan
#######################################
```

16.2 Primary 端 RAC 数据库的准备

为构造基于 RAC 的 Dataguard 环境，在 RAC 主数据库中需要做必要的准备工作，主要包括如下几个方面：

※ 数据库启动强制日志记录（Enable Force Logging）。

※ 启动归档模式（Enable Archivelog Mode）。

※ 可选地，创建备用重做日志（Standby Redo Log）。

※ 对数据库执行一次物理备份。

※ 创建备用数据库的控制文件（Controlfile for Standby）。

※ 针对备用数据库配置网络服务。

※ 为 Dataguard 设置主数据库的初始化参数。

※ 为备用数据库准备初始化参数文件。

16.2.1 检查和调整主数据库的运行

Dataguard 环境下的主数据库需要启动强制日志记录和归档日志模式。首先，检查主数据库的日志记录模式与归档模式；然后，登录 RAC 数据库的其中一个节点，执行如下操作：

```
SYS@BJING1>select force_logging from v$database;

FOR
---
YES

SYS@BJING1>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
```

```
Oldest online log sequence     6
Next log sequence to archive   7
Current log sequence           7
```

如果主数据库没有处于上述强制日志记录状态和归档日志模式，就需要对其进行调整。RAC 集群数据库在调整时需要关闭其他所有实例，像单实例数据库那样只有一个实例运行，并建议调整参数 cluster_database=false。

```
$ srvctl stop database -d BJING
$ sqlplus / as sysdba
Connected to an idle instance.

SYS@BJING1>startup mount
ORACLE instance started.
Total System Global Area  534462464 bytes
Fixed Size                  2177456 bytes
Variable Size             390071888 bytes
Database Buffers          134217728 bytes
Redo Buffers                7995392 bytes
Database mounted.

SYS@BJING1>alter database force logging;
Database altered.

SYS@BJING1>alter database archivelog;
Database altered.

SYS@BJING1>alter database open;
Database altered.
```

16.2.2　备份主数据库

备份主数据库需注意两个方面的内容，一是对主数据库执行一次物理备份，为在备用端还原数据库做准备；二是在主数据库中创建备用数据库的控制文件。

在主数据库端备份完成后，将形成的备份集传输到备用端节点（执行数据库还原操作的节点）的对应位置，代码如下：

```
$ mkdir /oracle/backup
$ rman target / nocatalog
Recovery Manager: Release 11.2.0.1.0 - Production on Wed Feb
26 08:56:40 2014
```

```
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: ORCL (DBID=1366860597)

RMAN> run
2>   {
3>      sql "alter system switch logfile";
4>      allocate channel ch1 type disk format
5>          '/oracle/backup/Primary_bkp_for_stdby_%U';
6>      backup database;
7>      backup current controlfile for standby;
8>      sql "alter system archive log current";
9>   }

using target database control file instead of recovery catalog
sql statement: alter system switch logfile

allocated channel: ch1
channel ch1: SID=30 instance=bjing1 device type=DISK

Starting backup at 26-FEB-14
channel ch1: starting full datafile backup set
channel ch1: specifying datafile(s) in backup set
input datafile file number=00001
name=+DATGRP/bjing/datafile/system.259.840464925
input datafile file number=00003
name=+DATGRP/bjing/datafile/undotbs1.262.840464927
input datafile file number=00002
name=+DATGRP/bjing/datafile/sysaux.261.840464929
input datafile file number=00004
name=+DATGRP/bjing/datafile/undotbs2.260.840464929
input datafile file number=00005
name=+DATGRP/bjing/datafile/example.258.840464929
channel ch1: starting piece 1 at 26-FEB-14
channel ch1: finished piece 1 at 26-FEB-14
piece handle=/ORACLE/BACKUP/PRIMARY_BKP_FOR_STDBY_04P1IVFG_1_1
tag=TAG20140226T085720 comment=NONE
channel ch1: backup set complete, elapsed time: 00:00:15
```

```
channel ch1: starting full datafile backup set
channel ch1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ch1: starting piece 1 at 26-FEB-14
channel ch1: finished piece 1 at 26-FEB-14
piece handle=/ORACLE/BACKUP/PRIMARY_BKP_FOR_STDBY_05P1IVG0_1_1
tag=TAG20140226T085720 comment=NONE
channel ch1: backup set complete, elapsed time: 00:00:01
Finished backup at 26-FEB-14

Starting backup at 26-FEB-14
channel ch1: starting full datafile backup set
channel ch1: specifying datafile(s) in backup set
including standby control file in backup set
channel ch1: starting piece 1 at 26-FEB-14
channel ch1: finished piece 1 at 26-FEB-14
piece handle=/ORACLE/BACKUP/PRIMARY_BKP_FOR_STDBY_06P1IVG5_1_1
tag=TAG20140226T085741 comment=NONE
channel ch1: backup set complete, elapsed time: 00:00:01
Finished backup at 26-FEB-14

sql statement: alter system archive log current
released channel: ch1
```

16.2.3　主数据库端的网络配置

对备用数据库端将要运行的数据库及其实例配置网络服务名（Net Service Name）。结合前面给出的预置环境，更新 Oracle Net 网络服务配置文件 tnsname.ora 的内容。备用端各节点的本地监听器端口号为 1521，SCAN 监听器的端口号为 1535，与备用端有关的网络服务设置如下：

```
SHHAI =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.124)(PORT =
1535))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.125)(PORT =
1535))
```

```
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.126)(PORT =
1535))
    )
    (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = SHHAI)
    )
  )

SHHAI1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.113)(PORT =
1521))
    )
    (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = SHHAI)
            (INSTANCE_NAME = SHHAI1)
    )
  )

SHHAI2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.114)(PORT =
1521))
    )
    (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = SHHAI)
            (INSTANCE_NAME = SHHAI2)
    )
  )
```

如果上述配置仅在 RAC 主数据库端的一个节点配置，则需要将 tnsname.ora 文件复制至集群的所有节点。

16.2.4　主数据库端的参数调整

Dataguard 环境下的主数据库角色需要设置的参数主要有：LOG_ARCHIVE_CONFIG 和 LOG_ARCHIVE_DEST_n，前者设置 Dataguard 环境中的所有 DB_UNIQUE_NAME，后者设置远程日志传输的目的地及其传输方式。与此同时，为了将来的角色转换，建议设置备用数据库角色有关的初始化参数。

```
SYS@BJING1>alter system set LOG_ARCHIVE_CONFIG = 'DG_
CONFIG=(BJING, SHHAI)' scope=spfile sid='*';
System altered.

SYS@BJING1>alter system set LOG_ARCHIVE_DEST_2 =
'SERVICE=SHHAI1 LGWR ASYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_
ROLE) DB_UNIQUE_NAME=SHHAI' scope=spfile sid='*';
System altered.

SYS@BJING1>alter system set LOG_ARCHIVE_DEST_STATE_2=
defer scope=spfile sid='*';
System altered.
```

此处将备用端的日志接收指定为备用数据库实例 SHHAI1，如果不做此限定，可以在此通过网络服务名 SHHAI 来指定主数据库远程日志的传输目的地，这样，接收日志的备用节点是 RAC1 或 RAC2。

角色转换准备的备用数据库角色参数如下：

```
########################################################
# for switchover to standby role
########################################################
FAL_SERVER = SHHAI1, SHHAI2
STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT = '+DATGRP/shhai', '+DATGRP/bjing'
LOG_FILE_NAME_CONVERT = '+FRAGRP/shhai', '+FRAGRP/bjing'
########################################################
```

16.3　Standby 端 RAC 数据库的准备

物理备用数据库的最初形态是由主数据库的物理备份还原而成的，物理备用数据库的控制文件是在主数据库中生成的。因此，备用数据库端的主要工作如下：

※　将主数据库中形成的备份集、备用控制文件传输至备用端的相应位置。

※　复制主数据库端的口令文件、参数文件至备用端，并进行必要的修改。

※ 准备备用数据库实例的运行环境。

※ 启动备用数据库实例并利用备份还原数据库。

※ 配置针对主数据库及其实例的网络服务。

※ 创建备用重做日志（Standby Redo Logs）。

16.3.1 准备备用数据库实例

需要从主数据库端将如下内容复制至备用端，为备用数据库实例的启动和运行做准备：RMAN 物理备份集、备用控制文件、口令文件、初始化参数文件。

下面的操作在主数据库端完成。

```
$ scp $ORACLE_HOME/dbs/pfile_for_standby.ora
                                rac3:$ORACLE_HOME/dbs/
initshhai1.ora
$ scp -r /oracle/backup rac3:/oracle

$ scp $ORACLE_HOME/dbs/orapwbjing1
                                rac3:$ORACLE_HOME/dbs/
orapwshhai1
$ scp $ORACLE_HOME/dbs/orapwbjing1
                                rac4:$ORACLE_HOME/dbs/
orapwshhai2
```

在上述文件传输中，复制至备用端的口令文件需要根据备用实例修改其文件名；备用数据库实例的参数文件可以根据主数据库端复制来的参数文件进行必要修改，需要修改的内容如 db_unique_name、instance_name、remote_listener 等，同时，需要添加备用数据库的必要参数，如 FAL_SERVER、STANDBY_FILE_MANAGEMENT、DB_FILE_NAME_CONVERT、LOG_FILE_NAME_CONVERT 等。

下面是根据前面的预置环境给出的备用端初始化参数文件样本。

```
##########################################################
*.db_name='ORCL'
*.db_unique_name='SHHAI'

*.memory_target=536870912
*.db_block_size=8192

*.db_create_file_dest='+DATGRP'
*.db_recovery_file_dest='+FRAGRP'
```

```
*.db_recovery_file_dest_size=1468006400
*.diagnostic_dest='/oracle/base'

*.log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'

# *.control_files='...'

*.remote_login_passwordfile='exclusive'
*.compatible='11.2.0.1.0'
*.open_cursors=300
*.processes=150
*.undo_management='auto'

#########################################################
*.cluster_database=true
*.remote_listener='shhai-scan:1535'

SHHAI1.instance_number=1
SHHAI2.instance_number=2
SHHAI1.thread=1
SHHAI2.thread=2
SHHAI1.undo_tablespace='UNDOTBS1'
SHHAI2.undo_tablespace='UNDOTBS2'

#########################################################
# for standby role
#########################################################
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'
FAL_SERVER = BJING1, BJING2
STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT = '+DATGRP/bjing', '+DATGRP/shhai'
LOG_FILE_NAME_CONVERT = '+FRAGRP/bjing', '+FRAGRP/shhai'
#########################################################
# for switchover to primary role
#########################################################
LOG_ARCHIVE_DEST_2 = 'SERVICE=BJING1 LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=BJING'
#########################################################
```

另外，为了将来角色转换的需要，备用数据库端也需要配置针对主数据库及其实例的网络服务，备用端每个节点的网络服务配置相同。示例如下（本地监听器端口为 1521，SCAN 监听器端口为 1535）：

```
BJING =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.121)(PORT =
1535))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.122)(PORT =
1535))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.123)(PORT =
1535))
    )
    (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = BJING)
    )
  )

BJING1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.111)(PORT =
1521))
    )
    (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = BJING)
            (INSTANCE_NAME = BJING1)
    )
  )

BJING2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.112)(PORT =
1521))
    )
```

```
    (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = BJING)
            (INSTANCE_NAME = BJING2)
    )
  )
```

16.3.2　还原并启动备用数据库

首先，根据上述初始化参数文件启动实例 SHHAI1 至 nomount 状态后，还原备用控制文件；然后，加载数据库，通过主数据库中的备份还原备用数据库的数据文件。

```
$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Wed Feb 26 10:25:32
2014
Copyright (c) 1982, 2010, Oracle.  All rights reserved.
Connected to an idle instance.

SYS@SHHAI1>startup pfile='/oracle/initshhai.ora' nomount
ORACLE instance started.

Total System Global Area  534462464 bytes
Fixed Size                  2177456 bytes
Variable Size             381683280 bytes
Database Buffers          142606336 bytes
Redo Buffers                7995392 bytes
SYS@SHHAI1>host rman target /

Recovery Manager: Release 11.2.0.1.0 - Production on Wed Feb
26 10:40:03 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: ORCL (not mounted)

RMAN> restore standby controlfile
2> from '/oracle/backup/PRIMARY_BKP_FOR_STDBY_06P1IVG5_1_1';

Starting restore at 26-FEB-14
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
```

```
channel ORA_DISK_1: SID=139 instance=shhai1 device type=DISK

channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:07
output file name=+DATGRP/shhai/controlfile/current.256.840537703
output file name=+FRAGRP/shhai/controlfile/current.256.840537705
Finished restore at 26-FEB-14
```

根据上面还原出的备用控制文件，修改初始化参数文件中的 control_files 参数的指向，重新启动实例至 mount 状态。

```
SYS@SHHAI1>startup pfile='/oracle/initshhai.ora' mount
ORACLE instance started.

Total System Global Area  534462464 bytes
Fixed Size                  2177456 bytes
Variable Size             385877584 bytes
Database Buffers          138412032 bytes
Redo Buffers                7995392 bytes
Database mounted.
SYS@SHHAI1>
SYS@SHHAI1>select database_role from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY

SYS@SHHAI1>host rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Wed Feb
26 10:48:03 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: ORCL (DBID=1366860597, not open)

RMAN> restore database;

Starting restore at 26-FEB-14
Starting implicit crosscheck backup at 26-FEB-14
allocated channel: ORA_DISK_1
Crosschecked 2 objects
```

```
Finished implicit crosscheck backup at 26-FEB-14

Starting implicit crosscheck copy at 26-FEB-14
using channel ORA_DISK_1
Finished implicit crosscheck copy at 26-FEB-14

using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
+DATGRP/shhai/datafile/system.259.840464925
channel ORA_DISK_1: restoring datafile 00002 to
+DATGRP/shhai/datafile/sysaux.261.840464929
channel ORA_DISK_1: restoring datafile 00003 to
+DATGRP/shhai/datafile/undotbs1.262.840464927
channel ORA_DISK_1: restoring datafile 00004 to
+DATGRP/shhai/datafile/undotbs2.260.840464929
channel ORA_DISK_1: restoring datafile 00005 to
+DATGRP/shhai/datafile/example.258.840464929
channel ORA_DISK_1: reading from backup piece
/ORACLE/BACKUP/PRIMARY_BKP_FOR_STDBY_04P1IVFG_1_1
channel ORA_DISK_1: piece
handle=/ORACLE/BACKUP/PRIMARY_BKP_FOR_STDBY_04P1IVFG_1_1
tag=TAG20140226T085720
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 26-FEB-14
```

16.3.3　创建备用重做日志

为将要启动的备用数据库实例（不同的线程）创建备用重做日志。若原主数据库中每个日志线程有两个日志组，则备用数据库中的每个日志线程需要三组备用重做日志。

```
SYS@SHHAI1>alter database add standby logfile thread 1 group 5
'+FRAGRP' size 16M;
SYS@SHHAI1>alter database add standby logfile thread 1 group 6
'+FRAGRP' size 16M;
```

```
SYS@SHHAI1>alter database add standby logfile thread 1 group 7
'+FRAGRP' size 16M;
SYS@SHHAI1>alter database add standby logfile thread 2 group 8
'+FRAGRP' size 16M;
SYS@SHHAI1>alter database add standby logfile thread 2 group 9
'+FRAGRP' size 16M;
SYS@SHHAI1>alter database add standby logfile thread 2 group 10
'+FRAGRP' size 16M;
```

16.3.4 向集群中注册 RAC 备用数据库

备用数据库要以 RAC 方式运行，必须将数据库及其实例向集群件 Clusterware 注册，过程如下：

```
$ srvctl add database -d SHHAI -o /oracle/dbms
            -p +DATGRP/shhai/parameterfile/spfile.262.840550383
$ srvctl add instance -d SHHAI -i SHHAI1 -n rac3
$ srvctl add instance -d SHHAI -i SHHAI2 -n rac4
```

此处的参数 -p 指定的 spfile 文件是在实例 SHHAI1 中根据文本参数文件创建而来的，具体文件由 OMF 特性自动生成，类似如下：

```
SYS@SHHAI1>create spfile='+DATGRP' from pfile='/oracle/init.
ora';
```

接下来，可以对 RAC 集群数据库 SHHAI 做一次停止和启动，并在本地节点验证实例的状态。

```
$ srvctl stop database -d SHHAI
$ srvctl start database -d SHHAI -o mount
$ srvctl status database -d SHHAI
```

实例 SHHAI1 正在节点 rac3 上运行

实例 SHHAI2 正在节点 rac4 上运行

```
$ sqlplus / as sysdba
SYS@SHHAI1>select inst_id,instance_name,status from
gv$instance;

   INST_ID INSTANCE_NAME    STATUS
---------- ---------------- ------------
         1 shhai1           MOUNTED
         2 shhai2           MOUNTED
```

16.4　启动备用实例的托管恢复

经过前面的一系列准备，现在可以测试 RAC 主数据库 BJING 到 RAC 物理备用数据库 SHHAI 的日志传输和托管恢复了。

16.4.1　检查日志传输

首先，在 RAC 主数据库 BJING 的实例 BJING1 中测试日志传输，并在备用数据库 SHHAI 的实例 SHHAI1 中检查日志的接收情况。然后，同样检查实例 BJING2 的日志传输情况。

根据前面的参数配置，这里将主数据库实例 BJING1 和实例 BJING2 的日志都传输向备用节点 rac3（实例 SHHAI1）。

```
SYS@BJING1>alter system set log_archive_dest_state_2=enable;
System altered.

SYS@BJING1>alter system set log_archive_dest_state_2=enable
scope=spfile sid='*';
System altered.

SYS@BJING1>alter system switch logfile;
System altered.

SYS@BJING1>select inst_id,dest_id,error,status from gv$archive_
dest where dest_id=2 and inst_id=1;

   INST_ID         DEST_ID      ERROR
     STATUS
----------------------------------------------------------
         1         2
                 VALID

SYS@BJING1>archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     25
Next log sequence to archive   26
Current log sequence           26
```

```
SYS@SHHAI1>select  group#,thread#,sequence#,used,status,archi
ved
from v$standby_log;

    GROUP#    THREAD#  SEQUENCE#       USED STATUS     ARC
---------- ---------- ---------- ---------- ---------- ---
         5          1         26     169984  ACTIVE       YES
         6          1          0        512  UNASSIGNED   NO
         7          1          0        512  UNASSIGNED   YES
         8          2          0        512  UNASSIGNED   NO
         9          2         19    2042368  ACTIVE       YES
        10          2          0        512  UNASSIGNED   NO
```

进一步，在备用端实例 SHHAI1 中检查与物理备用有关的进程运行状况。

```
SYS@SHHAI1>select  client_process,process,thread#,sequence#,st
atus
  2  from v$managed_standby;

CLIENT_P PROCESS      THREAD#  SEQUENCE# STATUS
-------- --------- ---------- ---------- ------------
ARCH         ARCH             2         18  CLOSING
ARCH         ARCH             0          0  CONNECTED
ARCH         ARCH             1         25  CLOSING
ARCH         ARCH             1         24  CLOSING
UNKNOWN      RFS              0          0  IDLE
UNKNOWN      RFS              0          0  IDLE
LGWR         RFS              2         19  IDLE
LGWR         RFS              1         26  IDLE
UNKNOWN      RFS              0          0  IDLE
```

此处可以清楚地看到实例 SHHAI1 中启动了两个 RFS 进程，这两个进程分别用于接收来自主数据库端实例 BJING1（Thread 1）和实例 BJING2（Thread 2）传输过来的重做日志。

16.4.2 启动日志应用

根据图 16-1，并为了证实备用端的 RAC 架构，我们将备用数据库端的日志接收和重做应用分别置于实例 SHHAI1 和实例 SHHAI2，接下来，在实例 SHHAI2 中启动重做日志的应用。

```
$ sqlplus / as sysdba
SYS@SHHAI2>alter database recover managed standby database
disconnect;
Database altered.

SYS@SHHAI2>select client_process,process,thread#,sequence#,st
atus
  2  from v$managed_standby;

CLIENT_P PROCESS      THREAD#  SEQUENCE# STATUS
-------- --------- ---------- ---------- ------------
ARCH     ARCH               0          0            CONNECTED
ARCH     ARCH               0          0            CONNECTED
ARCH     ARCH               0          0            CONNECTED
ARCH     ARCH               0          0            CONNECTED
N/A      MRP0               2         19            WAIT_FOR_
LOG

......
SYS@SHHAI2>select client_process,process,thread#,sequence#,st
atus
  2  from v$managed_standby;

CLIENT_P PROCESS      THREAD#  SEQUENCE# STATUS
-------- --------- ---------- ---------- ------------
ARCH     ARCH               0          0 CONNECTED
ARCH     ARCH               0          0 CONNECTED
ARCH     ARCH               0          0 CONNECTED
ARCH     ARCH               0          0 CONNECTED
N/A      MRP0               1         27 WAIT_FOR_LOG
```

此处可以看出，实例 SHHAI2 中并没有启动 RFS 进程，而是启动了 MRP 进程（托管恢复进程）用于实际的重做日志应用。从上面的查询可以了解到，来自主数据库 BJING 不同实例（线程）的重做日志都是由同一个 MRP0 进程负责在实例 SHHAI2 中应用，在不同的时间执行上述查询可以看出 MRP0 进程应用不同线程日志的情况。

16.5 RAC 备用数据库的应用

Dataguard 的关键之处是通过数据库的冗余实现防灾和容灾，保障数据库的数据安全。因此，在大多数工程案例中，备用数据库采用单实例数据库运行即可满足应用需要。然而，如果备用数据库采用 RAC 形式，在实践中也会为用户带来额外的优势。

16.5.1 RAC 备用冷集群

如果主数据库是 RAC 形式，为了在角色转换（Switchover 或 Failover）时备用数据库提供对等的服务，可以将备用数据库也配置为对等的 RAC 形式，甚至备用端集群的节点数也与主数据库相同。如果备用端在正常运行的情况下没有特别的需求，我们只保留备用端 RAC 集群的其中的一个节点运行，用于接收从主数据库端传输过来的日志并启动日志应用，而集群的其他节点（实例）保持完全的关闭状态。这种对备用数据库实施了 RAC 配置但只保持单实例运行的形态，称为 RAC 备用冷集群。

这种 RAC 备用冷集群形式最大限度地利用了 RAC 集群数据库和 Dataguard 的优点。首先，备用数据库保持和主数据库同样的集群配置，这为将来可能的角色转换提供了完全对等的服务器环境和服务资源，不至于应用系统切换至备用环境而导致性能下降；其次，备用端处于备用状态下并没有启动集群服务和数据库实例的节点服务器可以用于其他服务，这样可以最大限度地利用有限的服务器硬件资源。

16.5.2 RAC 活动备用数据库

从 11g 开始，Oracle 的物理备用数据库在应用重做日志的同时，以只读的方式打开，这一特性被称为活动备用数据库（Active Dataguard）。活动备用数据库极大地增强了物理备用数据库的应用价值，而 RAC 物理备用数据库与 Active Dataguard 的结合，将进一步扩展备用数据库的适用范围并提升其应用价值。

在那些查询、只读访问负荷较大的系统中，通过 Active Data Guard 备用数据库实现分布式的只读、实时访问，活动备库以多节点 RAC 方式运行，通过活动备用数据库的多个实例实现分布式访问，主数据库主要负责事务处理的负荷，而备用数据库的 RAC 架构可以显著提高系统能够承载的只读访问负荷，如图 16-2 所示。

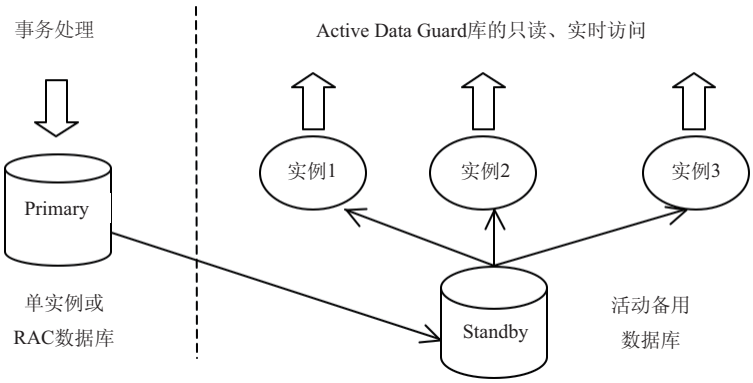


图 16-2　集群形式下的活动备用数据库

第17章

由 ASM 到 RAC+DG 的高可用之路

从前面的章节中，我们已经了解到，Oracle 数据库的高可用（High Availability）解决方案包括 DG（Data Guard）和 RAC（Real Application Cluster）两大策略，它们分别从数据库（Database）冗余和实例（Instance）冗余的角度给出了高可用方案的实施途径，RAC 和 DG 的有机结合可以为 Oracle 数据库在所有应用环境下提供完美的高可用方案。善于利用 DG 和 RAC，不仅可以实现数据的高可用性，同时，还可以实现扩展服务器容量、提高服务器性能、增强服务可靠性和数据安全性的目的。其中，ASM 技术是 Oracle 的综合存储方案，它可以看作一切 Oracle 数据库技术的存储基础。

本章将根据第 16 章的阐述，带领读者在 Oracle Linux 企业版的环境下进行 Oracle 数据库高可用环境的实践，作为全书的回顾与总结。

17.1 GI 独立服务器基础架构及其 ASM 数据库

实验环境的操作系统环境为 Oracle Linux Enterprise Edition 5.0，内核版本为 2.6.39-400.209.1.el5uek，安装了 GI 和 DBMS 所必需的软件包，安装和配置了 ASMLib 驱动。

系统中创建了用户组 oinstall、dba、asmadmin、asmdba 及其 grid、oracle 用户，并配置了用户环境，代码如下：

```
oinstall    grid, oracle        Oracle Inventory and Software
Owner
dba                oracle       Database Administrator
asmadmin    grid, oracle        Oracle Automatic Storage
Management Group
asmdba      grid, oracle        ASM Database Administrator
Group
```

系统通过了 GI 安装前的检查，代码如下：

```
[grid@host3]$ ./runcluvfy.sh stage -pre crsinst -n host3 -
verbose
...
Pre-check for cluster services setup was successful.
```

17.1.1　安装独立服务器网格基础架构

准备磁盘，并对磁盘执行必要的分区。

1. 准备 ASM 候选磁盘（裸设备，磁盘分区）

```
[root@host3 ~]# /etc/init.d/oracleasm createdisk ASMDSK1 /dev/
sdb1
Marking disk "ASMDSK1" as an ASM disk:                [  OK  ]
[root@host3 ~]#
[root@host3 ~]# /etc/init.d/oracleasm createdisk ASMDSK2 /dev/
sdb2
Marking disk "ASMDSK2" as an ASM disk:                [  OK  ]
```

2. 安装 Oracle Grid Infrastructure 软件

以 grid 用户安装 Oracle Grid Infrastructure 软件，选择“Install Grid Infrastructure Software Only”选项。

安装 GI 软件的最后，提示以 root 用户运行必要的脚本，示例如下。

```
============================================================
```

以 root 用户依次执行如下脚本：

```
/oracle/base/oraInventory/orainstRoot.sh
/oracle/grid/root.sh

[root@host3 ~]# /oracle/base/oraInventory/orainstRoot.sh
Changing permissions of /oracle/base/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.

Changing groupname of /oracle/base/oraInventory to oinstall.
The execution of the script is complete.
```

```
[root@host3 ~]#
[root@host3 ~]# /oracle/grid/root.sh
Running Oracle 11g root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= grid
    ORACLE_HOME=  /oracle/grid

Enter the full pathname of the local bin directory: [/usr/
local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...

Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.

To configure Grid Infrastructure for a Stand-Alone Server run
the following command as the root user:
/oracle/grid/perl/bin/perl -I/oracle/grid/perl/lib -I/oracle/
grid/crs/install /oracle/grid/crs/install/roothas.pl

To configure Grid Infrastructure for a Cluster perform the
following steps:
1. Provide values for Grid Infrastructure configuration parameters
in the file - /oracle/grid/crs/install/crsconfig_params. For
details on how to do this, see the installation guide.
2. Run the following command as the root user:
 /oracle/grid/perl/bin/perl -I/oracle/grid/perl/lib -I/oracle/
grid/crs/install /oracle/grid/crs/install/rootcrs.pl
To update inventory properties for Grid Infrastructure,
perform the following steps.
If a pre-11.2 home is already configured, execute the
following:
```

```
11.2_Home/oui/bin/runInstaller -updateNodeList -silent -local
CRS=false ORACLE_HOME=pre-11.2_Home
Always execute the following to register the current home:
11.2_Home/oui/bin/runInstaller -updateNodeList -silent -local
CRS=true ORACLE_HOME=11.2_Home.
If either home is shared, provide the additional argument
-cfs.

=====================================================
[root@host3 ~]# /oracle/grid/perl/bin/perl -I/oracle/grid/
perl/lib -I/oracle/grid/crs/install /oracle/grid/crs/install/
roothas.pl
2014-04-29 11:01:56: Checking for super user privileges
2014-04-29 11:01:56: User has super user privileges
2014-04-29 11:01:56: Parsing the host name
Using configuration parameter file:
/oracle/grid/crs/install/crsconfig_params
Creating trace directory
LOCAL ADD MODE
Creating OCR keys for user 'grid', privgrp 'oinstall'..
Operation successful.
CRS-4664: Node host3 successfully pinned.
Adding daemon to inittab
CRS-4123: Oracle High Availability Services has been started.
ohasd is starting
ADVM/ACFS is not supported on oraclelinux-release-5-10.0.2

host3     2014/04/29 11:02:28
     /oracle/grid/cdata/host3/backup_20140429_110228.olr
Successfully configured Oracle Grid Infrastructure for a
Standalone Server
=====================================================
```

安装 GI 时如果选择“安装和配置独立服务器网络基础设施”选项，则最后运行的脚本如下：

```
[root@host3 ~]# /oracle/base/oraInventory/orainstRoot.sh
Changing permissions of /oracle/base/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.
```

```
Changing groupname of /oracle/base/oraInventory to oinstall.
The execution of the script is complete.
[root@host3 ~]# /oracle/grid/root.sh
Running Oracle 11g root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= grid
    ORACLE_HOME=  /oracle/grid

Enter the full pathname of the local bin directory: [/usr/
local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...


Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
2014-05-02 16:15:59: Checking for super user privileges
2014-05-02 16:15:59: User has super user privileges
2014-05-02 16:15:59: Parsing the host name
Using configuration parameter file:
/oracle/grid/crs/install/crsconfig_params
Creating trace directory
LOCAL ADD MODE
Creating OCR keys for user 'grid', privgrp 'oinstall'..
Operation successful.
CRS-4664: Node host3 successfully pinned.
Adding daemon to inittab
CRS-4123: Oracle High Availability Services has been started.
ohasd is starting
ADVM/ACFS is not supported on oraclelinux-release-5-10.0.2

host3     2014/05/02 16:16:21
```

```
     /oracle/grid/cdata/host3/backup_20140502_161621.olr
Successfully configured Oracle Grid Infrastructure for a
Standalone Server
Updating inventory properties for clusterware
Starting Oracle Universal Installer...

Checking swap space: must be greater than 500 MB.    Actual
3047 MB    Passed
The inventory pointer is located at /etc/oraInst.loc
The inventory is located at /oracle/base/oraInventory
'UpdateNodeList' was successful.
=====================================================
```

3．检查 GI 的高可用服务及其资源（集群同步服务）

```
[root@host3 bin]# ./crsctl check has
CRS-4638: Oracle High Availability Services is online
[root@host3 bin]#
[root@host3 bin]# ./crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.cssd       ora.cssd.type    OFFLINE   OFFLINE
ora.diskmon    ora....on.type   OFFLINE   OFFLINE
[root@host3 bin]#
[root@host3 bin]# ./crsctl start resource ora.cssd
CRS-2672: Attempting to start 'ora.cssd' on 'host3'
CRS-2672: Attempting to start 'ora.diskmon' on 'host3'
CRS-2676: Start of 'ora.diskmon' on 'host3' succeeded
CRS-2676: Start of 'ora.cssd' on 'host3' succeeded
[root@host3 bin]#
[root@host3 bin]# ./crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.cssd       ora.cssd.type    ONLINE    ONLINE    host3
ora.diskmon    ora....on.type   ONLINE    ONLINE    host3
```

4．配置并启动集群同步服务 CSS

```
[grid@host3 bin]$ crs_stat -p ora.cssd
NAME=ora.cssd
```

```
TYPE=ora.cssd.type
ACTION_SCRIPT=
ACTIVE_PLACEMENT=0
AUTO_START=never
CHECK_INTERVAL=30
DESCRIPTION="Resource type for CSSD"
...

[grid@host3 bin]$ crsctl modify resource ora.cssd -attr AUTO_
START=1
[grid@host3 bin]$
[grid@host3 bin]$ crs_stat -p ora.cssd
NAME=ora.cssd
TYPE=ora.cssd.type
ACTION_SCRIPT=
ACTIVE_PLACEMENT=0
AUTO_START=1
CHECK_INTERVAL=30
DESCRIPTION="Resource type for CSSD"
...
```

#AUTO_START 的值也可以用 0、1、2 来表示，其中，0 等同 always，1 等同 restore，2 等同 never。

```
[grid@host3 bin]$ crsctl stop has
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'host3'
CRS-2673: Attempting to stop 'ora.cssd' on 'host3'
CRS-2677: Stop of 'ora.cssd' on 'host3' succeeded
CRS-2673: Attempting to stop 'ora.diskmon' on 'host3'
CRS-2677: Stop of 'ora.diskmon' on 'host3' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-
managed resources on 'host3' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[grid@host3 bin]$ crsctl start has
CRS-4123: Oracle High Availability Services has been started.
[grid@host3 bin]$ crsctl check has
CRS-4638: Oracle High Availability Services is online
[grid@host3 bin]$
[grid@host3 bin]$ crs_stat -t
```

```
Name           Type           Target    State     Host
------------------------------------------------------------
ora.cssd       ora.cssd.type  ONLINE          ONLINE    host3
ora.diskmon    ora....on.type ONLINE          ONLINE    host3
```

5．创建监听服务

使用 netca 或 lsnrctl 工具创建监听服务，代码如下：

```
[grid@host3 ~]$ lsnrctl status

LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 12-DEC-
2014 09:56:59

Copyright (c) 1991, 2009, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)
(KEY=EXTPROC1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                         TNSLSNR for Linux: Version
11.2.0.1.0 - Production
Start Date                12-DEC-2014 09:14:21
Uptime                    0 days 0 hr. 42 min. 37 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /oracle/grid/network/admin/listener.
ora
Listener Log File
/oracle/base/diag/tnslsnr/host3/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=host3)
(PORT=1521)))
Services Summary...
...
The command completed successfully
```

6. 创建并启动 ASM 实例

以 grid 用户创建 ASM 实例的口令文件，代码如下：

```
[grid@host3 ~]$ orapwd
file=/oracle/grid/dbs/orapw+ASM password=internal
```

在 /etc/oratab 文件中添加一行（可选）：

```
+ASM:/oracle/grid:Y
```

7. 启动 ASM 实例至 nomount 状态

（1）准备 ASM 实例参数，代码如下：

```
########################################################
*.instance_type='asm'
*.instance_name='+ASM'
*.memory_target=128m
*.large_pool_size=12M
*.asm_power_limit=1
*.remote_login_passwordfile='EXCLUSIVE'
*.diagnostic_dest='/oracle/base'
*.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'
# *.asm_diskgroups='DATGRP','FRAGRP'
########################################################

[grid@host3 ~]$ vi /oracle/grid/dbs/init+ASM.ora
[grid@host3 ~]$
[grid@host3 ~]$ sqlplus / as sysasm

SQL*Plus: Release 11.2.0.1.0 Production on Tue Apr 29 11:32:35
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to an idle instance.

SQL> startup nomount
ASM instance started
Total System Global Area  267825152 bytes
Fixed Size                  1335924 bytes
Variable Size             241323404 bytes
```

```
ASM Cache                  25165824 bytes
```

（2）创建 ASM 实例的 spfile。注意：只有注册 ASM 实例，通过 srvctl 启动 ASM 后才可在 SQL*Plus 中创建 spfile 至 ASM 磁盘组，此时，Oracle 会自动修改 ASM 在 HA 中的配置。

```
SQL> create spfile='+DATGRP' from pfile;
File created.

[grid@host3 ~]$ srvctl config asm
ASM home: /oracle/grid
ASM listener: LISTENER
Spfile: +DATGRP/asm/asmparameterfile/registry.253.846157807
ASM diskgroup discovery string:
```

8．创建并加载 ASM 磁盘组

（1）以 grid 用户启动 ASM 配置助理 asmca。

（2）创建 ASM 磁盘组 create diskgroup ...

（3）加载磁盘组 alter diskgroup ... mount;

此处，创建两个磁盘组：DATGRP、FRAGRP，分别用于数据库及其闪回恢复区（Flash Recovery Area）的物理存储。

```
SQL> select name,state,type from v$asm_diskgroup;

NAME                 STATE                TYPE
-------------------- -------------------- ------------
DATGRP               MOUNTED              EXTERN
FRAGRP               MOUNTED              EXTERN
```

9．注册监听和 ASM 实例

向本地 OCR 注册监听、ASM 实例。注册 ASM 实例后，通过 srvctl 启动 ASM 实例后，磁盘组会自动注册。

```
[grid@host3 ~]$ srvctl add listener
[grid@host3 ~]$ srvctl add asm

[grid@host3 ~]$ srvctl stop listener
[grid@host3 ~]$ srvctl start listener
```

```
[grid@host3 ~]$ srvctl stop asm
[grid@host3 ~]$ srvctl start asm
```

配置 OCR 资源的自动启动，代码如下：

```
[grid@host3 ~]$ srvctl enable asm

[grid@host3 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      ONLINE       ONLINE
host3
ora.FRAGRP.dg      ora....up.type      ONLINE       ONLINE
host3
ora....ER.lsnr     ora....er.type      ONLINE       ONLINE
host3
ora.asm            ora.asm.type        ONLINE       ONLINE
host3
ora.cssd           ora.cssd.type       ONLINE       ONLINE
host3
ora.diskmon        ora....on.type      ONLINE       ONLINE
host3
```

10. 检查与备份本地 OCR

```
[grid@host3 ~]$ ocrcheck -local
Status of Oracle Local Registry is as follows :
         Version:                          3
         Total space (kbytes):         262120
         Used space (kbytes):       2116
         Available space (kbytes):     260004
         ID:
563197068
         Device/File Name:
                                          /oracle/grid/
cdata/localhost/host3.olr
Device/File integrity check succeeded
         Local registry integrity check succeeded
            Logical corruption check bypassed due to non-
privileged user.
```

此处，请比较 ocrcheck 的输出结果。

```
[grid@host3 ~]$ ocrconfig -local -manualbackup
PROTL-23: Message 23 not found;  product=srvm; facility=PROTL

[grid@host3 ~]$ su - root
[root@host3 ~]# cd /oracle/grid/bin
[root@host3 bin]# ./ocrconfig -local -manualbackup

host3     2014/04/29 11:55:02
     /oracle/grid/cdata/host3/backup_20140429_115502.olr
host3     2014/04/29 11:02:28
     /oracle/grid/cdata/host3/backup_20140429_110228.olr
```

11. GI 独立服务器架构总结

资源依赖关系：HAS → CSS → LISTENER → ASM → DISKGROUP。

```
[grid@host3 ~]$ crsctl stop res -all
...
[grid@host3 ~]$ crsctl stop has
...
[grid@host3 ~]$ crsctl start has
...
[grid@host3 ~]$ crsctl start res -all
...
[grid@host3 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      ONLINE       ONLINE
host3
ora.FRAGRP.dg      ora....up.type      ONLINE       ONLINE
host3
ora....ER.lsnr     ora....er.type      ONLINE       ONLINE
host3
ora.asm            ora.asm.type        ONLINE       ONLINE
host3
ora.cssd           ora.cssd.type       ONLINE       ONLINE
host3
ora.diskmon        ora....on.type      ONLINE       ONLINE
host3
```

17.1.2 安装配置 DBMS 软件和数据库

1. 安装 DBMS 软件

以 Oracle 用户安装 Oracle DBMS 软件，并选择“仅安装数据库软件”。

2. 创建基于文件存储的数据库

（1）以 Oracle 用户（管理员身份）创建数据库实例口令文件，代码如下：

```
orapwd file=/oracle/dbms/dbs/orapwBJING password=internal
```

（2）编辑初始化参数文件。

（3）设置 ORACLE_SID 环境变量，启动实例（nomount）。

（4）创建数据库 CREATE DATABASE BJING。

3. 向 OCR 注册数据库

```
[oracle@host3 ~]srvctl add database -d BJING -o /oracle/dbms
-p  /oracle/dbms/dbs/spfileBJING.ora

[oracle@host3 ~]$ srvctl config database -d BJING
Database unique name: BJING
Database name:
Oracle home: /oracle/dbms
Oracle user: grid
Spfile: /oracle/dbms/dbs/spfileBJING.ora
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Disk Groups:
Services:

[grid@host3 ~]$ crs_stat -p ora.bjing.db
NAME=ora.bjing.db
TYPE=ora.database.type
ACTION_SCRIPT=
ACTIVE_PLACEMENT=1
AUTO_START=restore
```

```
CHECK_INTERVAL=1
DESCRIPTION=Oracle Database resource
......

[grid@host3 ~]$ srvctl enable database -d BJING
PRCC-1010 : BJING was already enabled

[grid@host3 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      ONLINE       ONLINE
host3
ora.FRAGRP.dg      ora....up.type      ONLINE       ONLINE
host3
ora....ER.lsnr     ora....er.type      ONLINE       ONLINE
host3
ora.asm            ora.asm.type        ONLINE       ONLINE
host3
ora.bjing.db       ora....se.type      ONLINE       ONLINE
host3
ora.cssd           ora.cssd.type       ONLINE       ONLINE
host3
ora.diskmon        ora....on.type      ONLINE       ONLINE
host3
```

17.1.3　迁移数据库至 ASM 存储

1. 数据库准备

（1）如果当前数据库正在使用跟踪功能（参见视图 v$block_change_tracking），则关闭其跟踪功能，代码如下：

```
SQL>alter database disable block change tracking;
```

（2）关闭数据库。

（3）修改初始化参数，代码如下：

```
db_create_file_dest='+DATGRP'
db_recovery_file_dest='+FRAGRP'
# control_files=...
```

（4）将实例启动到 nomount 状态。

2. 迁移控制文件至 ASM

```
SQL> show parameters control_files

NAME                         TYPE        VALUE
------------------------------------------------------------
control_files       string      /oracle/dat/BJING/bjingctl.ora
SQL> shutdown
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup nomount
ORACLE instance started.

Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             327157384 bytes
Database Buffers          201326592 bytes
Redo Buffers                5840896 bytes
SQL> host rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Fri Dec
12 19:09:08 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (not mounted)

RMAN> restore controlfile from '/oracle/dat/BJING/bjingctl.
ora';
Starting restore at 2014-12-12 19:09:16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=20 device type=DISK
channel ORA_DISK_1: copied control file copy
output file name=+DATGRP/bjing/controlfile/current.270.866142561
output file name=+FRAGRP/bjing/controlfile/current.272.866142563
Finished restore at 2014-12-12 19:09:25
```

3．迁移数据文件

```
SQL> show parameters control_files

NAME                                 TYPE                 VALUE
------------------------------------------------------
control_files                        string
                           +DATGRP/bjing/controlfile/
current.270.866142561,
+FRAGRP/bjing/controlfile/current.272.866142563
SQL> alter database mount;
Database altered.

SQL> host rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Fri Dec
12 21:32:06 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (DBID=3050489608, not
open)

RMAN> backup as copy database format '+DATGRP';
Starting backup at 2014-12-12 21:32:26
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=25 device type=DISK
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001 name=/oracle/dat/BJING/
bjingsys.ora
output file name=+DATGRP/bjing/datafile/system.267.866151151
tag=TAG20141212T213229 RECID=1 STAMP=866151155
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00002 name=/oracle/dat/BJING/
bjingaux.ora
output file name=+DATGRP/bjing/datafile/sysaux.269.866151157
tag=TAG20141212T213229 RECID=2 STAMP=866151161
```

```
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00004 name=/oracle/dat/BJING/
bjingud1.ora
output file name=+DATGRP/bjing/datafile/undotbs1.268.866151165
tag=TAG20141212T213229 RECID=3 STAMP=866151165
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003 name=/oracle/dat/BJING/
bjingexp.dbf
output file name=+DATGRP/bjing/datafile/example.266.866151167
tag=TAG20141212T213229 RECID=4 STAMP=866151167
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:03
channel ORA_DISK_1: starting datafile copy
copying current control file
output file name=+DATGRP/bjing/controlfile/backup.265.866151169
tag=TAG20141212T213229 RECID=5 STAMP=866151172
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:07
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 2014-12-12 21:32:56
channel ORA_DISK_1: finished piece 1 at 2014-12-12 21:32:57
piece handle=+DATGRP/bjing/backupset/2014_12_12/nnsnf0_ta
g20141212t213229_0.263.866151177 tag=TAG20141212T213229
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 2014-12-12 21:32:57

RMAN> switch database to copy;
datafile 1 switched to datafile copy
"+DATGRP/bjing/datafile/system.267.866151151"
datafile 2 switched to datafile copy
```

```
"+DATGRP/bjing/datafile/sysaux.269.866151157"
datafile 3 switched to datafile copy
"+DATGRP/bjing/datafile/example.266.866151167"
datafile 4 switched to datafile copy
"+DATGRP/bjing/datafile/undotbs1.268.866151165"

RMAN> alter database open;
database opened
```

接下来迁移临时数据文件，代码如下：

```
SQL> select name from v$tempfile;
NAME
--------------------------------------------------
/oracle/dat/BJING/bjingtmp.ora

SQL> alter tablespace temptbs add tempfile '+DATGRP';
Tablespace altered.

SQL> alter tablespace temptbs
  2  drop tempfile '/oracle/dat/BJING/bjingtmp.ora';
Tablespace altered.

SQL> select name from v$tempfile;
NAME
--------------------------------------------------
+DATGRP/bjing/tempfile/temptbs.264.866151613
```

4．迁移日志文件

```
SQL> alter database add logfile group 3 size 10m;
Database altered.
SQL> alter database add logfile  group 4 size 10m;
Database altered.

SQL> alter system switch logfile;
System altered.
SQL> alter system checkpoint;
System altered.
```

```
SQL> alter database drop logfile group 1;
Database altered.
SQL> alter database drop logfile group 2;
Database altered.

SQL> select group#,members,sequence#,status from v$log;

    GROUP#    MEMBERS  SEQUENCE#       STATUS
---------- ---------- ---------- ----------------
         3          2         79              INACTIVE
         4          2         80              CURRENT
```

至此，数据库迁移完毕。

5. 迁移 spfile 至 ASM

```
SQL> show parameters spfile

NAME         TYPE         VALUE
------------------------------------------------------
Spfile       string       /oracle/dbms/dbs/spfileBJING.ora
SQL> create spfile='+DATGRP' from memory;
File created.
```

查询在 ASM 磁盘组上生成的参数文件，例如：

```
+datgrp/BJING/parameterfile/spfile.262.866154879
```

6. 注册数据库

```
[oracle@host3 ~]$ srvctl add database -d BJING -o /oracle/dbms
-p +datgrp/BJING/parameterfile/spfile.262.866154879
[oracle@host3 ~]$
[oracle@host3 ~]$ srvctl config database -d BJING
Database unique name: BJING
Database name:
Oracle home: /oracle/dbms
Oracle user: grid
Spfile: +datgrp/BJING/parameterfile/spfile.262.866154879
Domain:
Start options: open
```

```
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Disk Groups:
Services:
==================================================
```

17.2　Data Guard 及其备用数据库诞生记

实验环境的操作系统环境为 Oracle Linux Enterprise Edition 5.0，内核版本为 2.6.39-400.209.1.el5uek，安装了 GI 和 DBMS 必需的软件包，安装和配置了 ASMLib 驱动。

系统中创建了用户组 oinstall、dba、asmadmin、asmdba 及其 grid、oracle 用户，并配置了用户环境，代码如下：

```
oinstall    grid, oracle        Oracle Inventory and Software
Owner
dba                oracle                    D a t a b a s e
Administrator
asmadmin    grid                        Oracle Automatic Storage
Management Group
asmdba      grid, oracle        ASM Database Administrator
Group
```

系统通过了 GI 安装前的检查，代码如下：

```
[grid@host3]$ ./runcluvfy.sh stage -pre crsinst -n host3 -
verbose
...
Pre-check for cluster services setup was successful.
```

17.2.1　创建物理备用数据库的基本过程

此部分我们要建立一个典型的 Data Guard 环境，这里的主机采用“ASM 到 DG+RAC 的 HA 之路（一）”中建立的主机环境（unxs）。

1．主机及其数据库信息

```
==================================================
```

主机 host3，IP 地址：188.168.0.103，默认监听

```
==================================================
```

数据库唯一标识 BJING

```
==========================================================
    *.db_name='jiadp'
    *.instance_name=BJING
    *.db_unique_name=BJING
    *.memory_target=512M
    *.db_block_size=8192
    *.control_files='+DATGRP/bjing/controlfile/
    current.270.866142561','+FRAGRP/bjing/controlfile/
    current.272.866142563'
    *.db_create_file_dest='+DATGRP'
    *.db_recovery_file_dest='+FRAGRP'
    *.db_recovery_file_dest_size=1G
    *.log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
    *.diagnostic_dest='/oracle/base'
    *.open_cursors=300
    *.processes=150
    *.remote_login_passwordfile='EXCLUSIVE'
    *.compatible='11.2.0.1.0'
    *.undo_management='AUTO'
    *.undo_tablespace='UNDOTBS1'
    *.local_listener
    ='(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'
    ========================================================
    SQL> archive log list;
    Database log mode               No Archive Mode
    Automatic archival              Disabled
    Archive destination             USE_DB_RECOVERY_FILE_DEST
    Oldest online log sequence      81
    Current log sequence            82
    ========================================================
    SQL> select dbid,name,log_mode,force_logging from v$database;
          DBID         NAME      LOG_MODE         FORCE_LOGGING
    ---------- --------- ------------ --------------------
    3050489608  JIADP     NOARCHIVELOG     NO
    ========================================================
```

2．主数据库的准备

主数据库的准备包括三个方面：数据库配置为归档模式、强制日志、必须使用口令文件。

```
SQL> startup mount
ORACLE instance started.

Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             331351688 bytes
Database Buffers          197132288 bytes
Redo Buffers                5840896 bytes
Database mounted.
SQL> alter database archivelog;
Database altered.

SQL> alter database force logging;
Database altered.

SQL> alter database open;
Database altered.

SQL> select dbid,name,log_mode,force_logging from v$database;
      DBID         NAME      LOG_MODE     FORCE_LOGGING
---------- --------- ------------ ---------------
3050489608  JIADP     ARCHIVELOG   YES

SQL> archive log list;
Database log mode               Archive Mode
Automatic archival              Enabled
Archive destination             USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     81
Next log sequence to archive    82
Current log sequence            82
=====================================================
[oracle@host3 ~]$ orapwd file=/oracle/dbms/dbs/orapwBJING
password=internal
[oracle@host3 ~]$ ls -l /oracle/dbms/dbs/orapwBJING
```

```
-rw-r----- 1 oracle oinstall 1536 Dec 13 14:32
/oracle/dbms/dbs/orapwBJING
```

3．备用主机的状况

主机 host1，IP 地址：188.168.0.101，已安装 DBMS 软件，默认监听

```
================================================
```

数据库唯一标识 SHHAI

从主数据库端复制初始化参数文件、口令文件，根据备用数据库唯一标识，修改参数文件内容、修改口令文件名、修改 Oracle 用户的环境变量 ORACLE_SID

```
================================================
[root@host1 ~]# ls -l /oracle
[root@host1 ~]# cd /oracle
[root@host1 oracle]# mkdir dat
[root@host1 oracle]# mkdir -p dat/SHHAI
[root@host1 oracle]# mkdir bak
[root@host1 oracle]# mkdir fra
[root@host1 oracle]# chown -R oracle:oinstall dat
[root@host1 oracle]# chown -R oracle:oinstall bak
[root@host1 oracle]# chown -R oracle:oinstall fra
[root@host1 oracle]# chmod -R 775 dat
[root@host1 oracle]# chmod -R 775 bak
[root@host1 oracle]# chmod -R 775 fra

====================================================
[oracle@host3 ~]$ ls -l /oracle/dbms/dbs/orapwBJING
-rw-r----- 1 oracle oinstall 1536 Dec 13 14:32 /oracle/dbms/
dbs/orapwBJING
[oracle@host3 ~]$ cd /oracle/dbms/dbs
[oracle@host3 dbs]$ scp orapwBJING 188.168.0.101:/oracle/dbms/
dbs
The authenticity of host '188.168.0.102 (188.168.0.101)' can't
be established.
RSA key fingerprint is
a1:af:16:4b:82:b6:da:bf:a7:12:30:37:45:eb:08:11.
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added '188.168.0.102' (RSA) to the list
of known hosts.
orapwBJING                100% 1536     1.5KB/s   00:00
[oracle@host3 dbs]$ cd /oracle/dat
[oracle@host3 dat]$ scp initBJING.asm 188.168.0.101:/oracle/
dat
initBJING.asm             100%  984     1.0KB/s   00:00
=====================================================
[oracle@host1 dat]$ cd /oracle/dbms/dbs
[oracle@host1 dbs]$ ls
init.ora  orapwBJING
[oracle@host1 dbs]$ mv orapwBJING orapwSHHAI
[oracle@host1 ~]$ cd /oracle/dat
[oracle@host1 dat]$ ls
initBJING.asm  SHHAI
[oracle@host1 dat]$ mv initBJING.asm initSHHAI.ora
=====================================================
*.db_name='jiadp'
*.instance_name=SHHAI
*.db_unique_name=SHHAI
*.memory_target=512M
*.db_block_size=8192
*.control_files='/oracle/dat/SHHAI/shhaictl.ora'
*.db_create_file_dest='/oracle/dat'
*.db_recovery_file_dest='/oracle/fra'
*.db_recovery_file_dest_size=1G
*.log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
*.diagnostic_dest='/oracle/base'
*.open_cursors=300
*.processes=150
*.remote_login_passwordfile='EXCLUSIVE'
*.compatible='11.2.0.1.0'
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS1'
*.local_listener
='(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'
=====================================================
```

4．备份主数据库

```
[oracle@host3 bak]$ rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Sat Dec
13 15:28:47 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (DBID=3050489608)

RMAN> backup format '/oracle/bak/%U' database;
Starting backup at 2014-12-13 15:29:06
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=41 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=+DATGRP/bjing/datafile/system.267.866151151
input datafile file number=00002
name=+DATGRP/bjing/datafile/sysaux.269.866151157
input datafile file number=00004
name=+DATGRP/bjing/datafile/undotbs1.268.866151165
input datafile file number=00003
name=+DATGRP/bjing/datafile/example.266.866151167
channel ORA_DISK_1: starting piece 1 at 2014-12-13 15:29:08
channel ORA_DISK_1: finished piece 1 at 2014-12-13 15:29:23
piece handle=/oracle/bak/09pq2pq4_1_1 tag=TAG20141213T152908
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:15
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 2014-12-13 15:29:24
channel ORA_DISK_1: finished piece 1 at 2014-12-13 15:29:25
piece handle=/oracle/bak/0apq2pqj_1_1 tag=TAG20141213T152908
comment=NONE
```

```
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
Finished backup at 2014-12-13 15:29:25
```

将备份结果传输到备用数据库端，代码如下：

```
[oracle@host3 bak]$ scp * 188.168.0.101:/oracle/bak
09pq2pq4_1_1              100%  227MB   3.9MB/s   00:59
0apq2pqj_1_1              100% 7744KB   3.8MB/s   00:02
[oracle@host3 bak]$
```

5．创建备用数据库的控制文件

```
RMAN> backup as copy format '/oracle/bak/shhaictl.ora' current
controlfile for standby;

Starting backup at 2014-12-13 15:36:57
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=47 device type=DISK
channel ORA_DISK_1: starting datafile copy
copying standby control file
output file name=/oracle/bak/shhaictl.ora tag=TAG20141213T153658
RECID=10 STAMP=866216219
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:01
Finished backup at 2014-12-13 15:37:00
Recovery Manager complete.
```

备注，创建备用数据库的控制文件也可使用如下方式：

```
SQL> alter database create standby controlfile
as '/oracle/bak/shhaictl.ora';
=====================================================
[oracle@host3 bak]$ scp shhaictl.ora 188.168.0.101:/oracle/
dat/SHHAI
shhaictl.ora                                  100%  7664KB
7.5MB/s   00:01
```

6．启动备用数据库实例

```
SQL> create spfile from pfile='/oracle/dat/initSHHAI.ora';
File created.
```

```
SQL> startup mount
ORACLE instance started.
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             327157384 bytes
Database Buffers          201326592 bytes
Redo Buffers                5840896 bytes
Database mounted.

SQL> select dbid,name,database_role from v$database;

      DBID         NAME      DATABASE_ROLE
---------- --------- ----------------
3050489608  JIADP     PHYSICAL STANDBY
```

7. 配置主备用数据库之间的网络服务

主数据库端的网络配置如下：

```
=====================================================
SHHAI =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.101)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = SHHAI)
    )
  )
=====================================================
```

备用数据库端的网络配置如下：

```
=====================================================
BJING =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.103)(PORT =
1521))
```

```
    )
    (CONNECT_DATA =
      (SERVICE_NAME = BJING)
    )
  )
==========================================================
```

8．设置主备用数据库参数

主数据库端的参数设置如下：

```
==========================================================
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'
LOG_ARCHIVE_DEST_2 = 'SERVICE=SHHAI LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=SHHAI'

############For Switch to Physical Standby###########
FAL_SERVER = SHHAI
STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT =     '/oracle/dat/SHHAI',
'+DATGRP/bjing/datafile'
LOG_FILE_NAME_CONVERT =    '/oracle/dat/SHHAI',
'+FRAGRP/bjing/onlinelog/'
==========================================================
```

备用数据库端的网络配置如下：

```
==========================================================
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'

FAL_SERVER = BJING
STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT =    '+DATGRP/bjing/datafile',
'/oracle/dat/SHHAI',
                                                '+DATGRP/bjing/
tempfile',
'/oracle/dat/SHHAI'
```

```
LOG_FILE_NAME_CONVERT =   '+FRAGRP/bjing/onlinelog/',
'/oracle/dat/SHHAI'
################For Switch to Primary#########################
LOG_ARCHIVE_DEST_2 = 'SERVICE=BJING LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=BJING'
=====================================================
```

9．在备用端还原数据库

```
SQL> host rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Sat Dec
13 17:11:02 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (DBID=3050489608, not
open)

RMAN> list backup;
using target database control file instead of recovery catalog
List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------------------------------------------------------------
4       Full    227.20M    DISK        00:00:10     2014-12-13
15:29:18
         BP Key: 4   Status: AVAILABLE  Compressed: NO  Tag:
TAG20141213T152908
        Piece Name: /oracle/bak/09pq2pq4_1_1
  List of Datafiles in backup set 4
  File LV Type Ckp SCN    Ckp Time            Name
  ---- -- ---- ---------- ------------------- ----
  1       Full 314106     2014-12-13 15:29:08
/oracle/dat/SHHAI/system.267.866151151
  2       Full 314106     2014-12-13 15:29:08
/oracle/dat/SHHAI/sysaux.269.866151157
  3       Full 314106     2014-12-13 15:29:08
```

```
/oracle/dat/SHHAI/example.266.866151167
  4       Full 314106     2014-12-13 15:29:08
/oracle/dat/SHHAI/undotbs1.268.866151165

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------------------------------------------------------------
5       Full    7.55M      DISK        00:00:02     2014-12-13
15:29:25
         BP Key: 5    Status: AVAILABLE  Compressed: NO  Tag:
TAG20141213T152908
        Piece Name: /oracle/bak/0apq2pqj_1_1
  SPFILE Included: Modification time: 2014-12-13 14:46:23
  SPFILE db_unique_name: BJING
  Control File Included: Ckp SCN: 314113       Ckp time: 2014-
12-13 15:29:23

RMAN> restore database;

Starting restore at 2014-12-13 17:11:27
Starting implicit crosscheck backup at 2014-12-13 17:11:27
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=27 device type=DISK
Crosschecked 2 objects
Finished implicit crosscheck backup at 2014-12-13 17:11:28

Starting implicit crosscheck copy at 2014-12-13 17:11:28
using channel ORA_DISK_1
Crosschecked 8 objects
Finished implicit crosscheck copy at 2014-12-13 17:11:29

searching for all files in the recovery area
cataloging files...
cataloging done

List of Cataloged Files
=======================
File Name:
```

```
/oracle/fra/SHHAI/archivelog/2014_12_13/o1_mf_1_83_b8qyql0k_.
arc
File Name:
/oracle/fra/SHHAI/archivelog/2014_12_13/o1_mf_1_84_b8qyqkys_.
arc
File Name:
/oracle/fra/SHHAI/archivelog/2014_12_13/o1_mf_1_86_b8qyqp5h_.
arc
File Name:
/oracle/fra/SHHAI/archivelog/2014_12_13/o1_mf_1_82_b8qyqkt2_.
arc
File Name:
/oracle/fra/SHHAI/archivelog/2014_12_13/o1_mf_1_85_b8qyqlos_.
arc

using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
/oracle/dat/SHHAI/system.267.866151151
channel ORA_DISK_1: restoring datafile 00002 to
/oracle/dat/SHHAI/sysaux.269.866151157
channel ORA_DISK_1: restoring datafile 00003 to
/oracle/dat/SHHAI/example.266.866151167
channel ORA_DISK_1: restoring datafile 00004 to
/oracle/dat/SHHAI/undotbs1.268.866151165
channel ORA_DISK_1: reading from backup piece
/oracle/bak/09pq2pq4_1_1
channel ORA_DISK_1: piece handle=/oracle/bak/09pq2pq4_1_1
tag=TAG20141213T152908
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:08
Finished restore at 2014-12-13 17:11:38
```

10．检查日志传输

启动主数据库后，检查日志传输。

```
SQL> alter system switch logfile;
System altered.

SQL> select status, error from v$archive_dest where dest_id=2;
STATUS    ERROR
-----------------------------------------------------
VALID
```

备用端检查日志的接收和应用情况如下：

```
SQL> select dest_id,sequence#,archived,applied,status,fal
from v$archived_log;
......
```

11．备用端启动托管恢复

```
SQL> alter database recover managed standby database
disconnect;
Database altered.
```

稍等片刻，再次检查备用端检查日志的接收和应用情况，代码如下：

```
SQL> select dest_id,sequence#,archived,applied,status,fal
from v$archived_log;
......
```

12．数据更新测试

主数据库端的数据更新：

（1）运行脚本 scott_main.sql 创建 scott 模式。

（2）手工切换日志。

备用端的数据更新检查：

（1）停止托管恢复。

（2）以只读方式打开数据库。

（3）检查数据更新情况：

```
SYS@SHHAI>alter database recover managed standby database
cancel;
Database altered.
```

```
SYS@SHHAI>alter database open read only;
Database altered.

SYS@SHHAI>select * from scott.dept;
    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES                CHICAGO
        40 OPERATIONS     BOSTON
==================================================
```

17.2.2 创建逻辑备用数据库的基本过程

本节针对同一主数据库 BJING 创建一个名为 GZHOU 逻辑备用数据库。

1. 为逻辑备用检查主数据库

在物理备用要求的基础上检查如下 3 项：

（1）不受支持的模式。

```
select owner,statement_opt from dba_logstdby_skip
where statement_opt='INTERNAL SCHEMA';
```

（2）不支持的表（由于数据类型）。

```
select distinct owner, table_name
from dba_logstdby_unsupported order by owner,table_name;
```

（3）存在唯一性问题的表。

```
select * from dba_logstdby_not_unique;
```

若 bad_column='Y'，则表示该表中还有不支持的数据类型字段，调整后可支持，例如：

```
ALTER TABLE scott.bonus add CONSTRAINT pk_bonus
PRIMARY KEY (ename, job) RELY DISABLE;
```

2. 为主数据库添加补充日志（由于 ROWID 问题）

```
SQL> select
  2  supplemental_log_data_min min,
```

```
  3  supplemental_log_data_pk pk,
  4  supplemental_log_data_ui ui from v$database;

MIN      PK  UI
-------- --- ---
NO       NO  NO

alter database add supplemental log
  2  data(primary key,unique index) columns;
Database altered.

SQL> select
  2  supplemental_log_data_min min,
  3  supplemental_log_data_pk pk,
  4  supplemental_log_data_ui ui from v$database;

MIN      PK  UI
-------- --- ---
IMPLICIT YES YES
```

3．构造物理备用数据库

本节在物理备用数据库的主机上构造逻辑备用数据库。

（1）设置环境变量。

```
[oracle@host1 ~]$ export ORACLE_SID=GZHOU
```

（2）复制并重命名口令文件

```
[oracle@host1 ~]$ cd /oracle/dbms/dbs
[oracle@host1 dbs]$ ls
[oracle@host1 dbs]$ cp orapwSHHAI orapwGZHOU
```

（3）复制并重命名控制文件。

```
[oracle@host1 ~]$ cd /oracle/dat
[oracle@host1 dat]$ mkdir GZHOU
[oracle@host1 dat]$ chmod -R 775 GZHOU
[oracle@host1 dat]$ cd /oracle/dat/SHHAI
[oracle@host1 SHHAI]$ ls
example.266.866151167   group_4.259.866152253   shhaictl.ora
system.267.866151151   undotbs1.268.866151165
```

```
group_3.256.866152231   onlinelog   sysaux.269.866151157
temptbs.264.866151613
[oracle@host1 SHHAI]$ cp shhaictl.ora ../GZHOU/gzhouctl.ora
```

（4）复制并修改参数文件。

```
[oracle@host1 dat]$ cp initSHHAI.ora initGZHOU.ora
[oracle@host1 dat]$ gedit initGZHOU.ora
########################################################
*.db_name='jiadp'
*.instance_name=GZHOU
*.db_unique_name=GZHOU
*.memory_target=512M
*.db_block_size=8192
*.control_files='/oracle/dat/GZHOU/gzhouctl.ora'
*.db_create_file_dest='/oracle/dat'
*.db_recovery_file_dest='/oracle/fra'
*.db_recovery_file_dest_size=1G
*.log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
*.diagnostic_dest='/oracle/base'
*.open_cursors=300
*.processes=150
*.remote_login_passwordfile='EXCLUSIVE'
*.compatible='11.2.0.1.0'
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS1'
*.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'

########################################################
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI, GZHOU)'

FAL_SERVER = BJING, SHHAI
STANDBY_FILE_MANAGEMENT = AUTO
DB_FILE_NAME_CONVERT =   '+DATGRP/bjing/datafile/', '/oracle/
dat/GZHOU/','+DATGRP/bjing/tempfile/','/oracle/dat/GZHOU/'
LOG_FILE_NAME_CONVERT =  '+FRAGRP/bjing/onlinelog/','/oracle/
dat/GZHOU/'
########################################################
```

（5）启动实例至 mount 状态。

```
[oracle@host1 dat]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Sun Dec 14 09:16:32
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to an idle instance.

SQL> create spfile from pfile='/oracle/dat/initGZHOU.ora';
File created.

SQL> startup mount
ORACLE instance started.
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             327157384 bytes
Database Buffers          201326592 bytes
Redo Buffers                5840896 bytes
Database mounted.
SQL> select name from v$datafile;

NAME
--------------------------------------------------------
/oracle/dat/GZHOU/system.267.866151151
/oracle/dat/GZHOU/sysaux.269.866151157
/oracle/dat/GZHOU/example.266.866151167
/oracle/dat/GZHOU/undotbs1.268.866151165
...
```

（6）配置主备用数据库的网络环境。

主数据库端的网络配置如下：

```
=====================================================
GZHOU =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.101)(PORT =
1521))
    )
    (CONNECT_DATA =
```

```
      (SERVICE_NAME = GZHOU)
    )
  )
==========================================================
```

主数据库增加向逻辑备用数据库端归档的参数，代码如下：

```
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI, GZHOU)' ...
LOG_ARCHIVE_DEST_3 = 'SERVICE=GZHOU LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=GZHOU'
==========================================================
```

备用数据库端的网络配置如下：

```
==========================================================
SHHAI =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.103)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = SHHAI)
    )
  )
==========================================================
```

调整逻辑备用数据库的初始化参数，代码如下：

```
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI, GZHOU)'
FAL_SERVER = BJING, SHHAI
==========================================================
```

（7）通过备份还原数据库。

```
[oracle@host1 ~]$ export ORACLE_SID=GZHOU
[oracle@host1 ~]$ rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Sun Dec
14 13:55:18 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
```

```
connected to target database: JIADP (DBID=3050489608, not
open)

RMAN> restore database;
Starting restore at 2014-12-14 13:55:43
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=29 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
/oracle/dat/GZHOU/system.267.866151151
channel ORA_DISK_1: restoring datafile 00002 to
/oracle/dat/GZHOU/sysaux.269.866151157
channel ORA_DISK_1: restoring datafile 00003 to
/oracle/dat/GZHOU/example.266.866151167
channel ORA_DISK_1: restoring datafile 00004 to
/oracle/dat/GZHOU/undotbs1.268.866151165
channel ORA_DISK_1: reading from backup piece
/oracle/bak/09pq2pq4_1_1
channel ORA_DISK_1: piece handle=/oracle/bak/09pq2pq4_1_1
tag=TAG20141213T152908
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 2014-12-14 13:56:00
```

（8）检查日志传输，并同步物理备用。

```
SQL> alter database recover managed standby database
disconnect;
Database altered.
...
SQL> alter database recover managed standby database cancel;
Database altered.
```

4．转换物理备用为逻辑备用

（1）在主数据库中构造 Logminer 字典，并及时通过日志流传向备用端。

```
SQL> exec dbms_logstdby.build;
```

```
PL/SQL procedure successfully completed.

SQL> alter system switch logfile;
System altered.
```

（2）转换物理备用库至逻辑备用库（可重新指定数据库名 DB_NAME）。

```
SQL> select dbid, database_role from v$database;

      DBID  DATABASE_ROLE
---------- ----------------
3050489608  PHYSICAL STANDBY

SQL> alter database recover to logical standby GZHOU;
Database altered.

SQL> shutdown immediate;
ORA-01507: database not mounted
ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.

Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             327157384 bytes
Database Buffers          201326592 bytes
Redo Buffers                5840896 bytes
Database mounted.
SQL> alter database open resetlogs;
Database altered.

SQL> select dbid, database_role from v$database;

      DBID  DATABASE_ROLE
---------- ------------------
1448506242  LOGICAL STANDBY
```

5．根据需要忽略部分主数据库的操作

```
SQL> exec dbms_logstdby.skip('ALTER TABLESPACE');
```

```
PL/SQL procedure successfully completed.

SQL> exec dbms_logstdby.skip('CREATE TABLESPACE');
PL/SQL procedure successfully completed.
```

注意：此处参数的赋值需要大写。利用 skip 还可以略过特定模式下的操作，代码如下：

```
exec dbms_logstdby.skip(stmt => 'DML', schema_name => 'SCOTT',
object_name => 'TEST/_%', ESC=>'/');
exec dbms_logstdby.skip(stmt => 'SCHEMA_DDL', schema_name =>
'SCOTT');
```

6. 启动逻辑备用端的 SQL Apply

```
=====================================================
SQL> alter database start logical standby apply;
```

检查验证主数据库与逻辑备用数据库的数据更新（有延迟，因日志挖掘需要时间）。

```
=====================================================
```

特别说明，由于创建了逻辑备用数据库 GZHOU，原物理备用数据库的初始化参数也应相应调整，代码如下：

```
LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI, GZHOU)'
LOG_ARCHIVE_DEST_3 = 'SERVICE=GZHOU LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=GZHOU'
=====================================================
```

对应地，物理备用数据库端也需要配置网络服务名 GZHOU 指向逻辑备用端，代码如下：

```
=====================================================
GZHOU =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = GZHOU)
```

```
    )
  )
==================================================
```

至此，对主数据库 BJING 分别创建了物理备用数据库 SHHAI 和逻辑备用数据库 GZHOU，且 3 个数据库可以同时运行，两个备用数据库对主数据库的保护模式是默认的 Maximize Performance 模式。

17.3 RAC 集群数据库的构造过程

实验环境的操作系统环境为 Oracle Linux Enterprise Edition 5.0，内核版本为 2.6.39-400.209.1.el5uek，安装了 GI 和 DBMS 所必需的软件包，安装和配置了 ASMLib 驱动。

系统中创建了用户组 oinstall、dba、asmadmin、asmdba 及其 grid、oracle 用户，并配置了用户环境，代码如下：

```
oinstall    grid, oracle        Oracle Inventory and Software
Owner
dba         oracle              Database Administrator
asmadmin    grid                Oracle Automatic Storage
Management Group
asmdba      grid, oracle        ASM Database Administrator
Group
```

系统通过了 GI 安装前的检查，代码如下：

```
[grid@rac1]$ ./runcluvfy.sh stage -pre crsinst -n rac1 -
verbose
......
Pre-check for cluster services setup was successful.
```

17.3.1 单节点 GI for Cluster 的安装和配置

本节介绍一种特殊的构造 RAC 集群数据库之路，首先，安装和配置单节点 RAC 集群，然后，由该节点扩展到其他节点。

1. 单节点主机的准备

（1）根据需要修改主机及其网络设置（主机名和 IP 地址），网络接口如下：

※ eth0（public 接口）。

※ eth1（private 接口）。

```
############################
127.0.0.1 localhost

# public
188.168.0.101 rac1

# private
192.168.1.101 rac1-prv

# vip
188.168.0.111 rac1-vip

# scan
188.168.0.123 rac-scan
############################
```

（2）分别修改 Grid 用户和 Oracle 用户的环境变量 ORACLE_SID。

※ Grid 用户：ORACLE_SID=+ASM1。

※ Oracle 用户：ORACLE_SID=BJING1。

（3）配置 SSH 用户等价。分别对 Grid 用户和 Oracle 用户配置 SSH 等价性，即使是本机也需要等价性测试。

```
$ ssh-keygen -t dsa
$ cd ~/.ssh
$ cat id_dsa.pub >> authorized_keys
$ ssh rac1
```

（4）GI 安装前的检查。

```
[grid@rac1 linux32-grid]$ ./runcluvfy.sh stage -pre crsinst -n
rac1 -verbose
  ......
  Pre-check for cluster services setup was successful.
```

2．配置共享存储

（1）准备两块独立的物理磁盘

※ 选择 SATA 或 SCSI 磁盘控制器接口，固定大小，可共享。

※ 磁盘设备 /dev/sda（unxocr.vdi），1GB，用于存储 OCR 和 Votedisk。

※ 磁盘设备 /dev/sdb（unxdat.vdi），3GB，两个分区（各 1.5GB），分别用于 database 和快速恢复区。

```
[root@rac1 ~]# fdisk /dev/sda
Command (m for help): p
Disk /dev/sda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         130     1044193+  83  Linux

[root@rac1 ~]# fdisk /dev/sdb
Command (m for help): p
Disk /dev/sdb: 3221 MB, 3221225472 bytes
255 heads, 63 sectors/track, 391 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         195     1566306   83  Linux
/dev/sdb2             196         391     1574370   83  Linux
```

（2）创建 ASM 磁盘。

```
[root@rac1 ~]# which oracleasm
/usr/sbin/oracleasm
[root@rac1 ~]# /etc/init.d/oracleasm createdisk ASMDSK1 /dev/
sda1
Marking disk "ASMDSK1" as an ASM disk:                [  OK  ]
[root@rac1 ~]# /etc/init.d/oracleasm createdisk ASMDSK2 /dev/
sdb1
Marking disk "ASMDSK2" as an ASM disk:                [  OK  ]
[root@rac1 ~]# /etc/init.d/oracleasm createdisk ASMDSK3 /dev/
sdb2
Marking disk "ASMDSK3" as an ASM disk:                [  OK  ]
[root@rac1 ~]# /etc/init.d/oracleasm listdisks
ASMDSK1
ASMDSK2
ASMDSK3
```

3. 安装 Grid Infrastructure

以 Grid 用户安装 GI，选择“Install and Configure grid infrastructure for cluser”选项，进行典型安装，注意设置 SCAN 名，选择 Cluster Registry Storage Type 为 ASM 类型：

```
[grid@rac1 linux32-grid]$ ./runInstaller
  ...
```

安装过程选择 ASM 存储 OCR 和 Votedisk，并使用 ASMDSK1 创建第一个 ASM 磁盘组 OCRGRP（冗余方式 external）。

以 root 用户运行脚本，代码如下：

```
  # /oracle/base/oraInventory/orainstRoot.sh
  # /oracle/grid/root.sh
=====================================================
[root@rac1 ~]# /oracle/base/oraInventory/orainstRoot.sh
Changing permissions of /oracle/base/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.
Changing groupname of /oracle/base/oraInventory to oinstall.
The execution of the script is complete.

[root@rac1 ~]# /oracle/grid/root.sh
Running Oracle 11g root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= grid
    ORACLE_HOME=  /oracle/grid

Enter the full pathname of the local bin directory: [/usr/
local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...

Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
```

```
2014-12-15 09:43:16: Parsing the host name
2014-12-15 09:43:16: Checking for super user privileges
2014-12-15 09:43:16: User has super user privileges
Using configuration parameter file:
/oracle/grid/crs/install/crsconfig_params
Creating trace directory
LOCAL ADD MODE
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
  root wallet
  root wallet cert
  root cert export
  peer wallet
  profile reader wallet
  pa wallet
  peer wallet keys
  pa wallet keys
  peer cert request
  pa cert request
  peer cert
  pa cert
  peer root cert TP
  profile reader root cert TP
  pa root cert TP
  peer pa cert TP
  pa peer cert TP
  profile reader pa cert TP
  profile reader peer cert TP
  peer user cert
  pa user cert
Adding daemon to inittab
CRS-4123: Oracle High Availability Services has been started.
ohasd is starting
ADVM/ACFS is not supported on oraclelinux-release-5-10.0.2

CRS-2672: Attempting to start 'ora.gipcd' on 'rac1'
CRS-2672: Attempting to start 'ora.mdnsd' on 'rac1'
CRS-2676: Start of 'ora.gipcd' on 'rac1' succeeded
```

```
CRS-2676: Start of 'ora.mdnsd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.gpnpd' on 'rac1'
CRS-2676: Start of 'ora.gpnpd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'rac1'
CRS-2676: Start of 'ora.cssdmonitor' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.cssd' on 'rac1'
CRS-2672: Attempting to start 'ora.diskmon' on 'rac1'
CRS-2676: Start of 'ora.diskmon' on 'rac1' succeeded
CRS-2676: Start of 'ora.cssd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.ctssd' on 'rac1'
CRS-2676: Start of 'ora.ctssd' on 'rac1' succeeded

ASM created and started successfully.

DiskGroup OCRGRP created successfully.

clscfg: -install mode specified
Successfully accumulated necessary OCR keys.
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
CRS-2672: Attempting to start 'ora.crsd' on 'rac1'
CRS-2676: Start of 'ora.crsd' on 'rac1' succeeded
CRS-4256: Updating the profile
Successful addition of voting disk 8bbe34e3d55b4f6ebf4dbe918b1
cf7a7.
Successfully replaced voting disk group with +OCRGRP.
CRS-4256: Updating the profile
CRS-4266: Voting file(s) successfully replaced
##  STATE    File Universal Id                  File Name Disk
group
--  -----    -----------------           --------- ---------
 1. ONLINE    8bbe34e3d55b4f6ebf4dbe918b1cf7a7 (ORCL:ASMDSK1)
[OCRGRP]
Located 1 voting disk(s).
CRS-2673: Attempting to stop 'ora.crsd' on 'rac1'
CRS-2677: Stop of 'ora.crsd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.asm' on 'rac1'
CRS-2677: Stop of 'ora.asm' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.ctssd' on 'rac1'
```

```
CRS-2677: Stop of 'ora.ctssd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.cssdmonitor' on 'rac1'
CRS-2677: Stop of 'ora.cssdmonitor' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'rac1'
CRS-2677: Stop of 'ora.cssd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.gpnpd' on 'rac1'
CRS-2677: Stop of 'ora.gpnpd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.gipcd' on 'rac1'
CRS-2677: Stop of 'ora.gipcd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.mdnsd' on 'rac1'
CRS-2677: Stop of 'ora.mdnsd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.mdnsd' on 'rac1'
CRS-2676: Start of 'ora.mdnsd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.gipcd' on 'rac1'
CRS-2676: Start of 'ora.gipcd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.gpnpd' on 'rac1'
CRS-2676: Start of 'ora.gpnpd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'rac1'
CRS-2676: Start of 'ora.cssdmonitor' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.cssd' on 'rac1'
CRS-2672: Attempting to start 'ora.diskmon' on 'rac1'
CRS-2676: Start of 'ora.diskmon' on 'rac1' succeeded
CRS-2676: Start of 'ora.cssd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.ctssd' on 'rac1'
CRS-2676: Start of 'ora.ctssd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.asm' on 'rac1'
CRS-2676: Start of 'ora.asm' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'rac1'
CRS-2676: Start of 'ora.crsd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.evmd' on 'rac1'
CRS-2676: Start of 'ora.evmd' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.asm' on 'rac1'
CRS-2676: Start of 'ora.asm' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.OCRGRP.dg' on 'rac1'
CRS-2676: Start of 'ora.OCRGRP.dg' on 'rac1' succeeded

rac1      2014/12/15 09:48:39      /oracle/grid/cdata/rac1/
backup_20141215_094839.olr
```

```
Preparing packages for installation...
cvuqdisk-1.0.7-1
Configure Oracle Grid Infrastructure for a Cluster ...
succeeded
Updating inventory properties for clusterware
Starting Oracle Universal Installer...

Checking swap space: must be greater than 500 MB.    Actual
3046 MB    Passed
The inventory pointer is located at /etc/oraInst.loc
The inventory is located at /oracle/base/oraInventory
'UpdateNodeList' was successful.
=====================================================
```

【备注】此处存在 SCAN 名称解析问题，若 SCAN　IP 地址有效，可忽略。安装日志中的提示信息类似如下：

```
INFO: Checking Single Client Access Name (SCAN)...
INFO: Checking name resolution setup for "rac-scan"...
INFO: ERROR:
INFO: PRVF-4664 : Found inconsistent name resolution entries
for SCAN name "rac-scan"
INFO: ERROR:
INFO: PRVF-4657 : Name resolution setup check for "rac-scan" (IP
address: 188.168.0.123) failed
INFO: ERROR:
INFO: PRVF-4664 : Found inconsistent name resolution entries
for SCAN name "rac-scan"
INFO: Verification of SCAN VIP and Listener setup failed
=====================================================
```

4. GI 安装后的检查

```
[grid@rac1 ~]$ cluvfy stage -post crsinst -n rac1 -verbose
Performing post-checks for cluster services setup
...
```

此处同样出现 PRVF-4664 错误提示，原因同上，忽略。

```
=====================================================
```

（1）检查基础服务，代码如下：

```
[grid@rac1 ~]$ crsctl check cluster -n rac1
**************************************************************
rac1:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
[grid@rac1 ~]$ crsctl check crs
CRS-4638: Oracle High Availability Services is online
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
[grid@rac1 ~]$ crsctl stat res -init
......
[grid@rac1 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.LISTENER.lsnr  ora.listener.type  ONLINE  ONLINE  rac1
ora.LISTENER_SCAN1.lsnr   ora.scan_listener.type   ONLINE
ONLINE  rac1
ora.OCRGRP.dg  ora.diskgroup.type  ONLINE  ONLINE  rac1
ora.asm    ora.asm.type    ONLINE    ONLINE    rac1
ora.eons   ora.eons.type   ONLINE    ONLINE    rac1
ora.gsd    ora.gsd.type    OFFLINE   OFFLINE
ora.net1.network   ora.network.type   ONLINE   ONLINE  rac1
ora.oc4j   ora.oc4j.type   OFFLINE   OFFLINE
ora.ons    ora.ons.type    ONLINE    ONLINE    rac1
ora.rac1.ASM1.asm    application   ONLINE   ONLINE   rac1
ora.rac1.LISTENER_RAC1.lsnr   application  ONLINE  ONLINE  rac1
ora.rac1.gsd   application   OFFLINE   OFFLINE
ora.rac1.ons   application   ONLINE    ONLINE    rac1
ora.rac1.vip   ora.cluster_vip_net1.type  ONLINE  ONLINE  rac1
ora.scan1.vip  ora.scan_vip.type   ONLINE  ONLINE  rac1
==============================================================
```

（2）检查 Votedisk 和 OCR 信息，代码如下：

```
[grid@rac1 ~]$ crsctl query css votedisk
##  STATE    File Universal Id          File Name Disk group
```

```
--  -----    ----------------           --------- ---------
 1. ONLINE  8bbe34e3d55b4f6ebf4dbe918b1cf7a7 (ORCL:ASMDSK1)
[OCRGRP]
[grid@rac1 ~]$ ocrcheck
Status of Oracle Cluster Registry is as follows :
         Version                  :          3
         Total space (kbytes)     :     262120
         Used space (kbytes)      :       2120
         Available space (kbytes) :     260000
         ID                       : 2088774543
         Device/File Name         :    +OCRGRP
                                        Device/File integrity
check succeeded
                                    Device/File not configured
                                    Device/File not configured
                                    Device/File not configured
                                    Device/File not configured
         Cluster registry integrity check succeeded
            Logical corruption check bypassed due to non-
privileged user
[grid@rac1 ~]$ cat /etc/oracle/ocr.loc
ocrconfig_loc=+OCRGRP
local_only=FALSE

[grid@rac1 ~]$ asmcmd
ASMCMD> cd ...
ASMCMD> pwd
+ocrgrp/rac-scan/ocrfile
ASMCMD> ls
REGISTRY.255.866367943
```

（3）检查 ASM 实例及磁盘组，代码如下：

```
[grid@rac1 ~]$ sqlplus / as sysasm
SQL*Plus: Release 11.2.0.1.0 Production on Mon Dec 15 10:54:27
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
Production
```

```
With the Real Application Clusters and Automatic Storage
Management options

SQL> select name,state,type,total_mb,free_mb,voting_files
2  from v$asm_diskgroup;

NAME    TATE    TYPE    TOTAL_MB    FREE_MB    VOTE_FILES
----------------- -------------------- ------------ ----------
OCRGRP    MOUNTED    EXTERN    1019    666     N

=====================================================
```

17.3.2 Oracle 数据库软件的安装

```
[oracle@rac1 bin]$ ./cluvfy stage -pre dbinst -n rac1 -verbose

Performing pre-checks for database installation
...
Pre-check for database installation was successful.
=====================================================
[oracle@rac1 linux32-dbms]$ ./runInstaller
Starting Oracle Universal Installer...
```

选择 Install database software only。

选择 Real Application Clusters database installation。

```
...
[root@rac1 ~]# /oracle/dbms/root.sh
Running Oracle 11g root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /oracle/dbms

Enter the full pathname of the local bin directory: [/usr/
local/bin]:
The file "dbhome" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying dbhome to /usr/local/bin ...
```

```
The file "oraenv" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying oraenv to /usr/local/bin ...
The file "coraenv" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying coraenv to /usr/local/bin ...

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
Finished product-specific root actions.
=====================================================
```

17.3.3　创建集群数据库

本节从一个单实例文件数据库开始，将其转变为一个集群数据库。为了节约时间，这里的单实例文件数据库直接由物理备份还原形成。

目录准备：

```
[root@rac1 oracle]# mkdir dat bak fra
[root@rac1 oracle]# ls
bak  base  dat  dbms  fra  grid
[root@rac1 oracle]# chown -R oracle:oinstall dat bak fra
[root@rac1 oracle]# chmod -R 775 dat fra bak
[root@rac1 oracle]# ls -l
total 24
drwxrwxr-x  2 oracle oinstall 4096 Dec 15 11:39 bak
drwxrwxr-x  6 grid   oinstall 4096 Dec 15 11:07 base
drwxrwxr-x  2 oracle oinstall 4096 Dec 15 11:39 dat
drwxrwxr-x 72 oracle oinstall 4096 Dec 15 11:24 dbms
drwxrwxr-x  2 oracle oinstall 4096 Dec 15 11:39 fra
drwxr-xr-x 64 root   oinstall 4096 Dec 15 09:43 grid
```

1．从单实例文件数据库开始

我们从事先准备的物理备份文件 BJING.tar.gz 开始，该文件是数据库物理存储的脱机备份（冷备份、压缩包）。

```
[oracle@rac1 share]$ cp BJING.tar.gz /oracle/bak
[oracle@rac1 share]$ cd /oracle/bak
```

```
[oracle@rac1 bak]$ tar -zxf BJING.tar.gz
[oracle@rac1 bak]$ ls
BJING  BJING.tar.gz  initBJING.ora
[oracle@rac1 bak]$ mv BJING /oracle/dat
[oracle@rac1 bak]$ ls
BJING.tar.gz  initBJING.ora
[oracle@rac1 bak]$ mv initBJING.ora /oracle/dat
[oracle@rac1 bak]$ cd /oracle/dat
[oracle@rac1 dat]$ mv initBJING.ora initBJING1.ora
[oracle@rac1 dat]$ vi initBJING1.ora

==========================================================
*.db_name='jiadp' *.instance_name=BJING1 *.db_unique_
name=BJING *.memory_target=512M *.db_block_size=8192 *.control_
files='/oracle/dat/BJING/bjingctl.ora' db_create_file_dest='/
oracle/dat' db_recovery_file_dest='/oracle/fra' db_recovery_
file_dest_size=1G log_archive_dest_1 = 'LOCATION=USE_DB_
RECOVERY_FILE_DEST' *.diagnostic_dest='/oracle/base' *.open_
cursors=300 *.processes=150
*.remote_login_passwordfile='EXCLUSIVE'
*.compatible='11.2.0.1.0' *.undo_management='AUTO' *.undo_
tablespace='UNDOTBS1' *.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=127.0.0.1)(PORT=1521))'
==========================================================

[oracle@rac1 ~]$ orapwd file=/oracle/dbms/dbs/orapwBJING1
password=internal
[oracle@rac1 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Mon Dec 15 12:01:09
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to an idle instance.

SQL> startup pfile='/oracle/dat/initBJING1.ora';
ORACLE instance started.
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
```

```
Variable Size              327157384 bytes
Database Buffers           201326592 bytes
Redo Buffers                 5840896 bytes
Database mounted.
Database opened.
```

2．为数据库准备 ASM 磁盘组

```
[grid@rac1 disks]$ pwd
/dev/oracleasm/disks
[grid@rac1 disks]$ ls -l
total 0
brw-rw---- 1 grid asmadmin 8,  1 Dec 15 08:53 ASMDSK1
brw-rw---- 1 grid asmadmin 8, 17 Dec 15 08:54 ASMDSK2
brw-rw---- 1 grid asmadmin 8, 18 Dec 15 08:54 ASMDSK3
[grid@rac1 disks]$ sqlplus / as sysasm
SQL*Plus: Release 11.2.0.1.0 Production on Mon Dec 15 14:28:31
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -
Production
With the Real Application Clusters and Automatic Storage
Management options
SQL> select group_number,name,path from v$asm_disk;

GROUP_NUMBER       NAME          PATH
------------ ---------- ------------------------------
           0                    ORCL:ASMDSK2
           0                    ORCL:ASMDSK3
           1       ASMDSK1      ORCL:ASMDSK1

SQL> create diskgroup DATGRP external redundancy
  2  disk 'ORCL:ASMDSK2';
Diskgroup created.

create diskgroup FRAGRP external redundancy
  2  disk 'ORCL:ASMDSK3';
Diskgroup created.
```

```
SQL> select name,state,type,total_mb,free_mb,voting_files
  2  from v$asm_diskgroup;

NAME    STATE    TYPE    TOTAL_MB    FREE_MB    VO
------- ------------------ ---------- ---------- ---------
OCRGRP  MOUNTED  EXTERN  1019        666        N
DATGRP  MOUNTED  EXTERN  1529        1479       N
FRAGRP  MOUNTED  EXTERN  1537        1487       N
```

3．迁移数据库至 ASM

前面已经介绍了如何将数据库迁移至 ASM，下面直接操作。

备注：如果在其他节点存在数据库，这里也可采用复制的方法快速产生新的 ASM 数据库。

（1）为新的 ASM 数据库编辑参数文件，代码如下：

```
==========================================================
*.db_name='jiadp' *.instance_name=BJING1 *.db_unique_name=BJING
*.memory_target=512M *.db_block_size=8192 # *.control_
files='......' db_create_file_dest='+DATGRP' db_recovery_file_
dest='+FRAGRP' db_recovery_file_dest_size=1G log_archive_dest_1
= 'LOCATION=USE_DB_RECOVERY_FILE_DEST' *.diagnostic_dest='/
oracle/base' *.open_cursors=300 *.processes=150 *.remote_
login_passwordfile='EXCLUSIVE' *.compatible='11.2.0.1.0'
*.undo_management='AUTO' *.undo_tablespace='UNDOTBS1' *.local_
listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=188.168.0.101)(PORT=1521))'
==========================================================
```

（2）启动实例至 nomount，代码如下：

```
[oracle@rac1 ~]$ sqlplus / as sysdba
Connected to an idle instance.

SQL> create spfile from pfile='/oracle/dat/initBJING1.ora';
File created.

SQL> startup nomount
ORACLE instance started.
```

```
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             327157384 bytes
Database Buffers          201326592 bytes
Redo Buffers                5840896 bytes
```

（3）还原控制文件至 ASM，并加载数据库，代码如下：

```
SQL> host rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Mon Dec
15 16:06:03 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (not mounted)

RMAN> restore controlfile from '/oracle/dat/BJING/bjingctl.
ora';
Starting restore at 2014-12-15 16:07:06
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=25 device type=DISK

channel ORA_DISK_1: copied control file copy
output file name=+DATGRP/bjing/controlfile/current.256.866390833
output file name=+FRAGRP/bjing/controlfile/current.256.866390835
Finished restore at 2014-12-15 16:07:16
RMAN> alter database mount;
using target database control file instead of recovery catalog
database mounted.
```

（4）映像备份数据库至 ASM，并作文件切换，代码如下：

```
RMAN> backup as copy format='+DATGRP' database;
Starting backup at 2014-12-15 16:14:16
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001 name=/oracle/dat/BJING/
bjingsys.ora
output file name=+DATGRP/bjing/datafile/system.257.866391257
tag=TAG20141215T161417 RECID=1 STAMP=866391260
```

```
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00002 name=/oracle/dat/BJING/
bjingaux.ora
output file name=+DATGRP/bjing/datafile/sysaux.258.866391261
tag=TAG20141215T161417 RECID=2 STAMP=866391263
...
Finished backup at 2014-12-15 16:14:31

RMAN> switch database to copy;
datafile 1 switched to datafile copy
"+DATGRP/bjing/datafile/system.257.866391257"
datafile 2 switched to datafile copy
"+DATGRP/bjing/datafile/sysaux.258.866391261"
datafile 3 switched to datafile copy
"+DATGRP/bjing/datafile/example.260.866391265"
datafile 4 switched to datafile copy
"+DATGRP/bjing/datafile/undotbs1.259.866391265"
```

（5）迁移临时文件，代码如下：

```
SQL> select name from v$tempfile;
NAME
-----------------------------------------------------
/oracle/dat/BJING/bjingtmp.ora

SQL> alter database open;
Database altered.

SQL> alter tablespace temptbs
  2  add tempfile '+DATGRP' size 10m autoextend on;
Tablespace altered.

SQL> alter tablespace temptbs
  2  drop tempfile '/oracle/dat/BJING/bjingtmp.ora';
Tablespace altered.
```

（6）迁移联机日志文件，代码如下：

```
SQL> alter database add logfile group 3
```

```
  2  ('+FRAGRP') size 10M;
Database altered.

SQL> alter database add logfile group 4
  2  ('+FRAGRP') size 10M;
Database altered.

SQL> alter system switch logfile;
System altered.
SQL> alter system checkpoint;
System altered.

SQL> alter database drop logfile group 1;
Database altered.

SQL> alter database drop logfile group 2;
Database altered.

SQL> select member from v$logfile;
MEMBER
------------------------------------------------------
+FRAGRP/bjing/onlinelog/group_3.257.866391871
+FRAGRP/bjing/onlinelog/group_4.258.866391881
```

4．启动集群数据库

（1）运行集群数据库的专门脚本，代码如下：

```
SQL> connect / as sysdba
SQL> @$ORACLE_HOME/rdbms/admin/catclust.sql
......
PL/SQL procedure successfully completed.
```

（2）设置集群数据库参数，代码如下：

```
SQL> select name from v$controlfile;
NAME
------------------------------------------------------
+DATGRP/bjing/controlfile/current.256.866390833
+FRAGRP/bjing/controlfile/current.256.866390835
```

```
==========================================================
*.cluster_database=true
*.db_name=jiadp
*.db_unique_name=BJING
BJING1.instance_name=BJING1
BJING1.instance_number=1
BJING1.thread=1
*.control_files=
'+DATGRP/bjing/controlfile/current.256.866390833',
'+FRAGRP/bjing/controlfile/current.256.866390835'
......
*.undo_management=auto
BJING1.undo_tablespace=UNDOTBS1

BJING1.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=188.168.0.111)(PORT=1521))'
*.remote_listener=rac-scan:1521
==========================================================
```

根据新的参数设置重新启动数据库即可。此时，BJING 以集群数据库形态运行，不过只有一个实例而已。

17.3.4 向 OCR 注册集群数据库

1. 创建位于 ASM 的 spfile

```
SQL> create spfile='+DATGRP' from memory;
File created.
```

通过 asmcmd 查询生成的 spfile：

```
+datgrp/bjing/parameterfile/spfile.264.866406681
```

可选地，修改本地初始化参数文件 initBJING1.ora，做如下设置：

```
spfile='+datgrp/bjing/parameterfile/spfile.264.866406681'
```

同时删除本地的 spfile 文件。

2. 注册数据库至集群

```
[oracle@rac1 ~]$ srvctl add database -d BJING -o /oracle/dbms
-p +datgrp/bjing/parameterfile/spfile.264.866406681
```

```
[oracle@rac1 ~]$ srvctl add instance -d BJING -i BJING1 -n
rac1
[oracle@rac1 ~]$ srvctl enable database -d BJING
PRCC-1010 : BJING was already enabled
[oracle@rac1 ~]$ srvctl config database -d BJING
Database unique name: BJING
Database name:
Oracle home: /oracle/dbms
Oracle user: oracle
Spfile: +datgrp/bjing/parameterfile/spfile.264.866406681
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: BJING
Database instances: BJING1
Disk Groups:
Services:
Database is administrator managed
=====================================================

[grid@rac1 disks]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora.FRAGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora....ER.lsnr     ora....er.type      ONLINE    ONLINE    rac1
ora....N1.lsnr     ora....er.type      ONLINE    ONLINE    rac1
ora.OCRGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora.asm            ora.asm.type        ONLINE    ONLINE    rac1
ora.bjing.db       ora....se.type      ONLINE    ONLINE    rac1
ora.eons           ora.eons.type       ONLINE    ONLINE    rac1
ora.gsd            ora.gsd.type        OFFLINE   OFFLINE
ora....network     ora....rk.type      ONLINE    ONLINE    rac1
ora.oc4j           ora.oc4j.type       OFFLINE   OFFLINE
ora.ons            ora.ons.type        ONLINE    ONLINE    rac1
ora....SM1.asm     application         ONLINE    ONLINE    rac1
```

```
ora....C1.lsnr     application         ONLINE    ONLINE    rac1
ora.rac1.gsd       application         OFFLINE   OFFLINE
ora.rac1.ons       application         ONLINE    ONLINE    rac1
ora.rac1.vip       ora....t1.type      ONLINE    ONLINE    rac1
ora.scan1.vip      ora....ip.type      ONLINE    ONLINE    rac1
```

3．通过 SCAN 地址访问集群数据库

首先，查看本地监听和 SCAN 监听的运行状态，代码如下：

```
[grid@rac1 ~]$ lsnrctl status
LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 15-DEC-
2014 21:00:11
Copyright (c) 1991, 2009, Oracle.  All rights reserved.
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)
(KEY=LISTENER)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
......
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=188.168.0.101)
(PORT=1521)))   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=188.168.0.111)(PORT=1521)))
Services Summary...
Service "+ASM" has 1 instance(s).
  Instance "+ASM1", status READY, has 1 handler(s) for this
service...
Service "BJING" has 1 instance(s).
  Instance "BJING1", status READY, has 1 handler(s) for this
service...
The command completed successfully
[grid@rac1 ~]$ lsnrctl status LISTENER_SCAN1
LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 15-DEC-
2014 21:00:18
Copyright (c) 1991, 2009, Oracle.  All rights reserved.
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN1)))
STATUS of the LISTENER
```

```
------------------------
Alias                     LISTENER_SCAN1
......
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER_SCAN1)))
   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=188.168.0.123)
(PORT=1521)))
Services Summary...
Service "BJING" has 1 instance(s).
  Instance "BJING1", status READY, has 1 handler(s) for this
service...
The command completed successfully
```

然后，在客户端配置网络服务，测试针对集群数据库的连接，代码如下：

```
=====================================================
RACDB =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.123)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = BJING)
    )
  )
=====================================================
```

4．节点的启动与关闭

关闭节点的基本环节：

（1）停止节点资源，代码如下：

```
[grid@rac1 ~]$ crsctl stop res -all -n rac1
......
[grid@rac1 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      OFFLINE   ONLINE    rac1
ora.FRAGRP.dg      ora....up.type      OFFLINE   ONLINE    rac1
```

```
ora....ER.lsnr     ora....er.type      OFFLINE   OFFLINE
ora....N1.lsnr     ora....er.type      OFFLINE   OFFLINE
ora.OCRGRP.dg      ora....up.type      OFFLINE   OFFLINE
ora.asm            ora.asm.type        OFFLINE   ONLINE    rac1
ora.bjing.db       ora....se.type      OFFLINE   OFFLINE
ora.eons           ora.eons.type       OFFLINE   OFFLINE
ora.gsd            ora.gsd.type        OFFLINE   OFFLINE
ora....network     ora....rk.type      OFFLINE   OFFLINE
ora.oc4j           ora.oc4j.type       OFFLINE   OFFLINE
ora.ons            ora.ons.type        OFFLINE   OFFLINE
ora....SM1.asm     application         OFFLINE   ONLINE    rac1
ora....C1.lsnr     application         OFFLINE   OFFLINE
ora.rac1.gsd       application         OFFLINE   OFFLINE
ora.rac1.ons       application         OFFLINE   OFFLINE
ora.rac1.vip       ora....t1.type      OFFLINE   OFFLINE
ora.scan1.vip      ora....ip.type      OFFLINE   OFFLINE
```

（2）停止节点集群服务，代码如下：

```
[grid@rac1 ~]$ crsctl stop cluster -n rac1
CRS-4563: Insufficient user privileges.
CRS-4000: Command Stop failed, or completed with errors.
[grid@rac1 ~]$ su -
Password:
[root@rac1 ~]# cd /oracle/grid/bin
[root@rac1 bin]# ./crsctl stop cluster -n rac1
CRS-2673: Attempting to stop 'ora.crsd' on 'rac1'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'rac1'
CRS-2673: Attempting to stop 'ora.DATGRP.dg' on 'rac1'
CRS-2673: Attempting to stop 'ora.FRAGRP.dg' on 'rac1'
CRS-2677: Stop of 'ora.FRAGRP.dg' on 'rac1' succeeded
CRS-2677: Stop of 'ora.DATGRP.dg' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.asm' on 'rac1'
CRS-2677: Stop of 'ora.asm' on 'rac1' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources
on 'rac1' has completed
CRS-2677: Stop of 'ora.crsd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.cssdmonitor' on 'rac1'
CRS-2673: Attempting to stop 'ora.ctssd' on 'rac1'
```

```
CRS-2673: Attempting to stop 'ora.evmd' on 'rac1'
CRS-2673: Attempting to stop 'ora.asm' on 'rac1'
CRS-2677: Stop of 'ora.cssdmonitor' on 'rac1' succeeded
CRS-2677: Stop of 'ora.evmd' on 'rac1' succeeded
CRS-2677: Stop of 'ora.ctssd' on 'rac1' succeeded
CRS-2677: Stop of 'ora.asm' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'rac1'
CRS-2677: Stop of 'ora.cssd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.diskmon' on 'rac1'
CRS-2677: Stop of 'ora.diskmon' on 'rac1' succeeded
[root@rac1 bin]# ./crs_stat -t
CRS-0184: Cannot communicate with the CRS daemon.

[root@rac1 bin]# ./crsctl check crs
CRS-4638: Oracle High Availability Services is online
CRS-4535: Cannot communicate with Cluster Ready Services
CRS-4530: Communications failure contacting Cluster
Synchronization Services daemon
CRS-4534: Cannot communicate with Event Manager
```

（3）停止节点高可用服务，代码如下：

```
[root@rac1 bin]# ./crsctl stop has
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'rac1'
CRS-2673: Attempting to stop 'ora.mdnsd' on 'rac1'
CRS-2673: Attempting to stop 'ora.gpnpd' on 'rac1'
CRS-2677: Stop of 'ora.mdnsd' on 'rac1' succeeded
CRS-2677: Stop of 'ora.gpnpd' on 'rac1' succeeded
CRS-2673: Attempting to stop 'ora.gipcd' on 'rac1'
CRS-2677: Stop of 'ora.gipcd' on 'rac1' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-
managed resources on 'rac1' has completed
CRS-4133: Oracle High Availability Services has been stopped.
```

集群节点的启动，代码如下：

```
[root@rac1 bin]# ./crsctl start has
CRS-4123: Oracle High Availability Services has been started.
......
[root@rac1 bin]# ./crsctl check crs
```

```
CRS-4638: Oracle High Availability Services is online
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
[root@rac1 bin]# ./crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      OFFLINE   OFFLINE
ora.FRAGRP.dg      ora....up.type      OFFLINE   OFFLINE
ora....ER.lsnr     ora....er.type      OFFLINE   OFFLINE
ora....N1.lsnr     ora....er.type      OFFLINE   OFFLINE
ora.OCRGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora.asm            ora.asm.type        ONLINE    ONLINE    rac1
ora.bjing.db       ora....se.type      OFFLINE   OFFLINE
ora.eons           ora.eons.type       ONLINE    ONLINE    rac1
ora.gsd            ora.gsd.type        OFFLINE   OFFLINE
ora....network     ora....rk.type      ONLINE    ONLINE    rac1
ora.oc4j           ora.oc4j.type       OFFLINE   OFFLINE
ora.ons            ora.ons.type        ONLINE    ONLINE    rac1
ora....SM1.asm     application         ONLINE    ONLINE    rac1
ora....C1.lsnr     application         OFFLINE   OFFLINE
ora.rac1.gsd       application         OFFLINE   OFFLINE
ora.rac1.ons       application         ONLINE    ONLINE    rac1
ora.rac1.vip       ora....t1.type      OFFLINE   OFFLINE
ora.scan1.vip      ora....ip.type      OFFLINE   OFFLINE
[root@rac1 bin]# su - grid
[grid@rac1 ~]$ crsctl start res -all -n rac1
CRS-5702: Resource 'ora.OCRGRP.dg' is already running on
'rac1'
CRS-5702: Resource 'ora.asm' is already running on 'rac1'
CRS-5702: Resource 'ora.eons' is already running on 'rac1'
CRS-2501: Resource 'ora.gsd' is disabled
CRS-5702: Resource 'ora.net1.network' is already running on
'rac1'
CRS-2501: Resource 'ora.oc4j' is disabled
CRS-5702: Resource 'ora.ons' is already running on 'rac1'
CRS-5702: Resource 'ora.asm' is already running on 'rac1'
CRS-2501: Resource 'ora.gsd' is disabled
```

```
CRS-5702: Resource 'ora.ons' is already running on 'rac1'
CRS-2672: Attempting to start 'ora.rac1.vip' on 'rac1'
CRS-2672: Attempting to start 'ora.scan1.vip' on 'rac1'
CRS-2672: Attempting to start 'ora.DATGRP.dg' on 'rac1'
CRS-2672: Attempting to start 'ora.FRAGRP.dg' on 'rac1'
CRS-2676: Start of 'ora.scan1.vip' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.LISTENER_SCAN1.lsnr' on
'rac1'
CRS-2676: Start of 'ora.rac1.vip' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.LISTENER.lsnr' on 'rac1'
CRS-2676: Start of 'ora.LISTENER.lsnr' on 'rac1' succeeded
CRS-5702: Resource 'ora.LISTENER.lsnr' is already running on
'rac1'
CRS-2676: Start of 'ora.LISTENER_SCAN1.lsnr' on 'rac1'
succeeded
CRS-2672: Attempting to start 'ora.bjing.db' on 'rac1'
CRS-2676: Start of 'ora.DATGRP.dg' on 'rac1' succeeded
CRS-2676: Start of 'ora.FRAGRP.dg' on 'rac1' succeeded
CRS-2676: Start of 'ora.bjing.db' on 'rac1' succeeded
CRS-4000: Command Start failed, or completed with errors.
[grid@rac1 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora.FRAGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora....ER.lsnr     ora....er.type      ONLINE    ONLINE    rac1
ora....N1.lsnr     ora....er.type      ONLINE    ONLINE    rac1
ora.OCRGRP.dg      ora....up.type      ONLINE    ONLINE    rac1
ora.asm            ora.asm.type        ONLINE    ONLINE    rac1
ora.bjing.db       ora....se.type      ONLINE    ONLINE    rac1
ora.eons           ora.eons.type       ONLINE    ONLINE    rac1
ora.gsd            ora.gsd.type        OFFLINE   OFFLINE
ora....network     ora....rk.type      ONLINE    ONLINE    rac1
ora.oc4j           ora.oc4j.type       OFFLINE   OFFLINE
ora.ons            ora.ons.type        ONLINE    ONLINE    rac1
ora....SM1.asm     application         ONLINE    ONLINE    rac1
ora....C1.lsnr     application         ONLINE    ONLINE    rac1
ora.rac1.gsd       application         OFFLINE   OFFLINE
```

```
ora.rac1.ons        application          ONLINE    ONLINE    rac1
ora.rac1.vip        ora....t1.type       ONLINE    ONLINE    rac1
ora.scan1.vip       ora....ip.type       ONLINE    ONLINE    rac1
```

17.3.5 扩展集群至新节点

有了集群节点 rac1 后，并不需要 GI 和 DBMS 安装程序即可将集群扩展至其他新节点。下面将集群扩展至 rac2 节点。

1. 主机 rac2 的准备

（1）根据需要修改主机及其网络设置（主机名和 IP 地址），网络接口如下：

※ eth0（public 接口）。

※ eth1（private 接口）。

两个节点上的 hosts 文件修改如下：

```
#############################
127.0.0.1 localhost

# public
188.168.0.101 rac1
188.168.0.102 rac2

# private
192.168.1.101 rac1-prv
192.168.1.102 rac2-prv

# vip
188.168.0.111 rac1-vip
188.168.0.112 rac2-vip

# scan
188.168.0.123 rac-scan
#############################
```

备注：原 rac1 主机的 hosts 文件也需更新，建议修改后通过 scp 复制到另一台上。

（2）分别修改 grid 用户和 oracle 用户的环境变量 ORACLE_SID。

※ grid 用户：ORACLE_SID=+ASM2。

※ oracle 用户：ORACLE_SID=BJING2。

（3）配置 ssh 用户等价。分别对 grid 用户和 oracle 用户配置 ssh 等价性，即使是本机也需要等价性测试。

```
$ ssh-keygen -t dsa
$ cd ~/.ssh
$ cat id_dsa.pub >> authorized_keys
$ ssh rac1
```

（4）加载共享存储，代码如下：

```
[root@rac2 ~]# /etc/init.d/oracleasm scandisks
Scanning the system for Oracle ASMLib disks:
[  OK  ]
[root@rac2 ~]# /etc/init.d/oracleasm listdisks
ASMDSK1
ASMDSK2
ASMDSK3
```

（5）对新增节点 rac2 进行验证。在集群的已有节点（rac1）上对节点 rac2 执行集群扩展验证，代码如下：

```
[grid@rac1 ~] cd /oracle/grid/bin
[grid@rac1 bin]$ ./cluvfy stage -pre nodeadd -n rac2 -fixup
-verbose

Performing pre-checks for node addition
...
Pre-check for node addition was successful.
```

2. 扩展 GI 及其服务至 rac2 节点

在已有节点 rac1 上执行如下操作，将 Grid Infrastructure 软件及其服务扩展到 rac2 节点上。

```
[grid@rac1 ~]$ cd /oracle/grid/oui/bin
[grid@rac1 bin]$ ./addNode.sh -silent "CLUSTER_NEW_
NODES={rac2}" "CLUSTER_NEW_VIRTUAL_HOSTNAMES={rac2-vip}"
Starting Oracle Universal Installer...
Checking swap space: must be greater than 500 MB.   Actual
3047 MB    Passed
```

```
.....
Performing tests to see whether nodes rac2 are available
............................................................
. 100% Done.
.
-----------------------------------------------------
Cluster Node Addition Summary
Global Settings
   Source: /oracle/grid
   New Nodes
Space Requirements
   New Nodes
      rac2
         /: Required 3.32GB : Available 12.30GB
Installed Products
   Product Names
      Oracle Grid Infrastructure 11.2.0.1.0
      ...
      Cluster Ready Services Files 11.2.0.1.0
      Oracle Database 11g 11.2.0.1.0
-----------------------------------------------------

Instantiating scripts for add node (Tuesday, December 16, 2014
9:43:35 AM CST)
.
1% Done.
Instantiation of add node scripts complete

Copying to remote nodes (Tuesday, December 16, 2014 9:43:42 AM
CST)
................................................
................................................
96% Done.
Home copied to new nodes

Saving inventory on nodes (Tuesday, December 16, 2014 9:51:10
AM CST)
```

```
.
100% Done.
Save inventory complete
WARNING:A new inventory has been created on one or more nodes
in this session. However, it has not yet been registered as
the central inventory of this system.
To register the new inventory please run the script at '/
oracle/base/oraInventory/orainstRoot.sh' with root privileges
on nodes 'rac2'.
If you do not register the inventory, you may not be able to
update or patch the products you installed.
The following configuration scripts need to be executed as the
"root" user in each cluster node.
/oracle/base/oraInventory/orainstRoot.sh #On nodes rac2
/oracle/grid/root.sh #On nodes rac2
To execute the configuration scripts:
    1. Open a terminal window
    2. Log in as "root"
    3. Run the scripts in each cluster node

The Cluster Node Addition of /oracle/grid was successful.
Please check '/tmp/silentInstall.log' for more details.

=====================================================
```

接下来，按照提示在节点 rac2 上分别运行脚本：

```
[root@rac2 ~]# /oracle/base/oraInventory/orainstRoot.sh
Creating the Oracle inventory pointer file (/etc/oraInst.loc)
Changing permissions of /oracle/base/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.

Changing groupname of /oracle/base/oraInventory to oinstall.
The execution of the script is complete.
[root@rac2 ~]# /oracle/grid/root.sh
Running Oracle 11g root.sh script...

The following environment variables are set as:
    ORACLE_OWNER= grid
```

```
    ORACLE_HOME=  /oracle/grid

Enter the full pathname of the local bin directory: [/usr/
local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...

Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
2014-12-16 10:09:10: Parsing the host name
2014-12-16 10:09:10: Checking for super user privileges
2014-12-16 10:09:10: User has super user privileges
Using configuration parameter file: /oracle/grid/crs/install/
crsconfig_params
Creating trace directory
LOCAL ADD MODE
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
Adding daemon to inittab
CRS-4123: Oracle High Availability Services has been started.
ohasd is starting
ADVM/ACFS is not supported on oraclelinux-release-5-10.0.2

CRS-4402: The CSS daemon was started in exclusive mode but
found an active CSS daemon on node rac1, number 1, and is
terminating
An active cluster was found during exclusive startup,
restarting to join the cluster
CRS-2672: Attempting to start 'ora.mdnsd' on 'rac2'
CRS-2676: Start of 'ora.mdnsd' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.gipcd' on 'rac2'
CRS-2676: Start of 'ora.gipcd' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.gpnpd' on 'rac2'
CRS-2676: Start of 'ora.gpnpd' on 'rac2' succeeded
```

```
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'rac2'
CRS-2676: Start of 'ora.cssdmonitor' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.cssd' on 'rac2'
CRS-2672: Attempting to start 'ora.diskmon' on 'rac2'
CRS-2676: Start of 'ora.diskmon' on 'rac2' succeeded
CRS-2676: Start of 'ora.cssd' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.ctssd' on 'rac2'
CRS-2676: Start of 'ora.ctssd' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.asm' on 'rac2'
CRS-2676: Start of 'ora.asm' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'rac2'
CRS-2676: Start of 'ora.crsd' on 'rac2' succeeded
CRS-2672: Attempting to start 'ora.evmd' on 'rac2'
CRS-2676: Start of 'ora.evmd' on 'rac2' succeeded
clscfg: EXISTING configuration version 5 detected.
clscfg: version 5 is 11g Release 2.
Successfully accumulated necessary OCR keys.
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.

rac2       2014/12/16 10:10:25       /oracle/grid/cdata/rac2/
backup_20141216_101025.olr
Preparing packages for installation...
cvuqdisk-1.0.7-1
Configure Oracle Grid Infrastructure for a Cluster ...
succeeded
Updating inventory properties for clusterware
Starting Oracle Universal Installer...

Checking swap space: must be greater than 500 MB.    Actual
3047 MB    Passed
The inventory pointer is located at /etc/oraInst.loc
The inventory is located at /oracle/base/oraInventory
'UpdateNodeList' was successful.
=====================================================
```

扩展 GI 后的检查：

```
[grid@rac1 bin]$ pwd
/oracle/grid/bin
```

```
[grid@rac1 bin]$ ./cluvfy stage -post nodeadd -n rac2 -verbose
...
=====================================================
```

3. 扩展 DBMS 至 rac2 节点

```
[oracle@rac1 ~]$ cd /oracle/dbms/oui/bin
[oracle@rac1 bin]$ ./addNode.sh -silent "CLUSTER_NEW_
NODES={rac2}"
Starting Oracle Universal Installer...
Checking swap space: must be greater than 500 MB.    Actual
3047 MB    Passed
...
Performing tests to see whether nodes rac2 are available
............................................................
. 100% Done.
..
-----------------------------------------------------
Cluster Node Addition Summary
Global Settings
   Source: /oracle/dbms
   New Nodes
Space Requirements
   New Nodes
      rac2
         /: Required 3.67GB : Available 9.25GB
Installed Products
   Product Names
      Oracle Database 11g 11.2.0.1.0
      ......
      Enterprise Edition Options 11.2.0.1.0
-----------------------------------------------------

Instantiating scripts for add node (Tuesday, December 16, 2014
10:23:48 AM CST)
.
1% Done.
Instantiation of add node scripts complete
```

```
Copying to remote nodes (Tuesday, December 16, 2014 10:24:02
AM CST)
.....................................................
.....................................................
96% Done.
Home copied to new nodes

Saving inventory on nodes (Tuesday, December 16, 2014 10:47:58
AM CST)
.
100% Done.
Save inventory complete
WARNING:
The following configuration scripts need to be executed as the
"root" user in each cluster node.
/oracle/dbms/root.sh #On nodes rac2
To execute the configuration scripts:
    1. Open a terminal window
    2. Log in as "root"
    3. Run the scripts in each cluster node

The Cluster Node Addition of /oracle/dbms was successful.
Please check '/tmp/silentInstall.log' for more details.

=====================================================
```

运行提示的脚本：

```
[root@rac2 ~]# /oracle/dbms/root.sh
Running Oracle 11g root.sh script...
The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /oracle/dbms
Enter the full pathname of the local bin directory: [/usr/
local/bin]:
The file "dbhome" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying dbhome to /usr/local/bin ...
The file "oraenv" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
```

```
   Copying oraenv to /usr/local/bin ...
The file "coraenv" already exists in /usr/local/bin.  Overwrite
it? (y/n) [n]: y
   Copying coraenv to /usr/local/bin ...

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
Finished product-specific root actions.
=====================================================
```

4. 扩展实例至 rac2 节点

（1）为启动第二个实例的数据库准备。创建回滚表空间和日志组（日志线程）：

```
SQL> create undo tablespace UNDOTBS2
  2  datafile size 10m autoextend on;
Tablespace created.

SQL> select group#,thread#,sequence#,members,status from
v$log;

    GROUP#    THREAD#  SEQUENCE#    MEMBERS STATUS
---------- ---------- ---------- ---------- ----------------
         3          1         83          1 CURRENT
         4          1         82          1 INACTIVE

SQL> alter database add logfile thread 2
  2  group 1 ('+FRAGRP') size 10m;
Database altered.

SQL> alter database add logfile thread 2
  2  group 2 ('+FRAGRP') size 10m;
Database altered.

SQL> alter database enable thread 2;
Database altered.
```

（2）调整初始化参数设置，代码如下：

```
=====================================================
*.cluster_database=true
*.db_name='jiadp' *.db_unique_name=BJING
BJING1.instance_name=BJING1
BJING2.instance_name=BJING2
BJING1.instance_number=1
BJING2.instance_number=2
BJING1.thread=1
BJING2.thread=2
*.memory_target=512M  *.db_block_size=8192
*.control_files='+DATGRP/bjing/controlfile/
current.256.866390833','+FRAGRP/bjing/controlfile/
current.256.866390835' *.db_create_file_dest='+DATGRP' *.db_
recovery_file_dest='+FRAGRP' *.db_recovery_file_dest_size=1G
*.log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
*.diagnostic_dest='/oracle/base' *.open_cursors=300
*.processes=150  *.remote_login_passwordfile='EXCLUSIVE'
*.compatible='11.2.0.1.0' *.undo_management='AUTO' BJING1.
undo_tablespace='UNDOTBS1'
BJING2.undo_tablespace='UNDOTBS2' BJING1.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=188.168.0.111)(PORT=1521))'
BJING2.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=188.168.0.112)(PORT=1521))'
*.remote_listener=rac-scan:1521
=====================================================
```

（3）在节点 rac2 上启动实例 BJING2，代码如下：

```
[oracle@rac2 ~]$ orapwd file=/oracle/dbms/dbs/orapwBJING2
password=internal
[oracle@rac2 ~]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Tue Dec 16 11:30:02
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to an idle instance.

SQL> create spfile from pfile='/oracle/dat/initBJING.ora';
```

```
File created.

SQL> startup
ORACLE instance started.
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             385877640 bytes
Database Buffers          142606336 bytes
Redo Buffers                5840896 bytes
Database Mounted.
Database Opened.
```

此时，若数据库存在多余的日志线程及其日志组，可通过如下方式删除：

```
SQL> select group#,sequence#,thread#,members,status from
v$log;

    GROUP#  SEQUENCE#    THREAD#    MEMBERS STATUS
---------- ---------- ---------- ---------- ----------------
         1          1          2          1  CURRENT
         2          0          2          1  UNUSED
         3         83          1          1  CURRENT
         4         82          1          1  INACTIVE
         5          1          3          2  CURRENT
         6          0          3          2  UNUSED

SQL> alter database disable thread 3;
Database altered.

SQL> alter database drop logfile group 5;
Database altered.

SQL> alter database drop logfile group 6;
Database altered.
```

5. 创建公共的基于 ASM 的 spfile

```
SQL> create spfile='+DATGRP' from pfile='/oracle/dat/initBJING.
ora';
File created.
```

查询创建的 spfile 完整的路径及文件名：

```
+datgrp/bjing/parameterfile/spfile.267.866461989
```

可选地，可以本地方式使用公共的 spfle，分别编辑节点本地的默认初始化参数文件 initBJING1.ora 和 initBJING2.ora，添加一行：

```
spfile='+datgrp/bjing/parameterfile/spfile.267.866461989'
```

同时，删除本地的默认 spfile 文件。

6．注册新的数据库实例

```
[oracle@rac2 ~]$ srvctl remove database -d BJING
Remove the database BJING? (y/[n]) y
[oracle@rac2 ~]$ srvctl add database -d BJING -o /oracle/dbms
-p +datgrp/bjing/parameterfile/spfile.267.866461989
[oracle@rac2 ~]$ srvctl add instance -d BJING -i BJING1 -n
rac1
[oracle@rac2 ~]$ srvctl add instance -d BJING -i BJING2 -n
rac2
[oracle@rac2 ~]$ srvctl config database -d BJING
Database unique name: BJING
Database name:
Oracle home: /oracle/dbms
Oracle user: oracle
Spfile: +datgrp/bjing/parameterfile/spfile.267.866461989
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: BJING
Database instances: BJING1,BJING2
Disk Groups:
Services:
Database is administrator managed
=====================================================
```

17.3.6　删除集群中的节点

本节删除 rac2 节点。

1．停止被删除节点上所有的资源及其集群服务

（1）停止资源，代码如下：

```
[grid@rac2 ~]$ crsctl stop res -all -n rac2
...
[grid@rac2 ~]$ crs_stat -t
Name           Type           Target    State     Host
------------------------------------------------------------
ora.DATGRP.dg  ora....up.type ONLINE    ONLINE    rac1
......
ora.scan1.vip  ora....ip.type ONLINE    ONLINE    rac1
```

查看上面的结果中是否有资源在 rac2 节点上运行。停止被删除节点 rac2 上的集群服务。

（2）停止并禁用集群服务，代码如下：

```
[root@rac2 bin]# ./crsctl stop cluster -n rac2
CRS-2673: Attempting to stop 'ora.crsd' on 'rac2'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'rac2'
......
[root@rac2 bin]# ./crsctl stop has
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'rac2'
CRS-2673: Attempting to stop 'ora.mdnsd' on 'rac2'
CRS-2673: Attempting to stop 'ora.gpnpd' on 'rac2'
CRS-2677: Stop of 'ora.mdnsd' on 'rac2' succeeded
CRS-2677: Stop of 'ora.gpnpd' on 'rac2' succeeded
CRS-2673: Attempting to stop 'ora.gipcd' on 'rac2'
CRS-2677: Stop of 'ora.gipcd' on 'rac2' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-
managed resources on 'rac2' has completed
CRS-4133: Oracle High Availability Services has been stopped.
```

上述两步操作等价于 crsctl stop crs，该指令停止本地节点的集群服务，包括 has 服务。

最后，禁用被删除节点 rac2 上集群服务的自动启动：

```
[grid@rac2 ~]$ crsctl disable crs
CRS-4563: Insufficient user privileges.
```

```
CRS-4000: Command Disable failed, or completed with errors.
[grid@rac2 ~]$ su - root
[root@rac2 ~]# cd /oracle/grid/bin
[root@rac2 bin]# ./crsctl disable crs
CRS-4621: Oracle High Availability Services autostart is
disabled.
```

2．在保留节点上执行删除节点操作

（1）从 OCR 中删除节点。此操作即从 OCR 中删除所有与被删除节点有关的信息。

```
[root@rac1 bin]# ./crsctl delete node -n rac2
CRS-4661: Node rac2 successfully deleted.
[root@rac1 bin]# ./olsnodes
rac1
```

备注，若被删除节点 rac2 上的集群服务没有停止，此处执行删除节点操作会出现如下提示：

```
[root@rac1 bin]# ./crsctl delete node -n rac2
CRS-4658: The clusterware stack on node rac2 is not completely
down.
CRS-4000: Command Delete failed, or completed with errors.
```

此时，如果通过 crs_stat 指令发现仍有属于 rac2 节点的资源，可用 crsctl 指令手工删除，例如：

```
[root@rac1 ~]# cd /oracle/grid/bin
[root@rac1 bin]# ./crsctl delete res ora.rac2.vip
```

（2）更新 Oracle 软件的 Inventory。

```
[grid@rac1 ~]$ cd /oracle/grid/oui/bin
[grid@rac1 bin]$ ./runInstaller -updateNodeList ORACLE_HOME=/
oracle/grid "CLUSTER_NODES={rac1}"
Starting Oracle Universal Installer...

Checking swap space: must be greater than 500 MB.    Actual
3031 MB    Passed
The inventory pointer is located at /etc/oraInst.loc
The inventory is located at /oracle/base/oraInventory
'UpdateNodeList' was successful.
=====================================================
```

```
=====================================================
```

关于被删除的节点再重新加入集群的特别说明：

```
=====================================================
In 11g R2 RAC, when a node is deleted, only the OCR and the
inventory of the remaining nodes is updated. Grid software
from the deleted node is not removed and inventory of the
deleted node is not updated.
(/oracle/base/oraInventory/ContentsXML/inventory.xml)

Hence, to add back a deleted node we need not copy the Grid
software on the node which was deleted earlier. Only the
following steps need to be executed:

Run addNode.sh from a currently existing  node (say rac1) with
no copy option

[grid@rac1]$ cd /oracle/grid/oui/bin
 [grid@rac1]$./addNode.sh -silent -noCopy
"CLUSTER_NEW_NODES={rac2}"
"CLUSTER_NEW_VIRTUAL_HOSTNAMES={rac2-vip}"

This step updates the inventories on the nodes and instatiates
scripts on local node.
=====================================================
```

例如：

```
[grid@rac1 bin]$ ./addNode.sh -silent -noCopy "CLUSTER_NEW_
NODES={rac2}" "CLUSTER_NEW_VIRTUAL_HOSTNAMES={rac2-vip}"
Starting Oracle Universal Installer...

Checking swap space: must be greater than 500 MB.    Actual
3045 MB    Passed
Oracle Universal Installer, Version 11.2.0.1.0 Production
Copyright (C) 1999, 2009, Oracle. All rights reserved.

Performing tests to see whether nodes rac2 are available
```

```
.....................................................
100% Done.
...
-----------------------------------------------------
Cluster Node Addition Summary
Global Settings
   Source: /oracle/grid
   New Nodes
Space Requirements
   New Nodes
      rac2
         /: Required 3.54GB : Available 5.34GB
Installed Products
   Product Names
      Oracle Grid Infrastructure 11.2.0.1.0
      ......
      Cluster Ready Services Files 11.2.0.1.0
      Oracle Database 11g 11.2.0.1.0
-----------------------------------------------------

Instantiating scripts for add node (Tuesday, December 16, 2014
9:55:24 PM CST)
.
1% Done.
Instantiation of add node scripts complete

Saving inventory on nodes (Tuesday, December 16, 2014 9:55:39
PM CST)
.
100% Done.
Save inventory complete
WARNING:
The following configuration scripts need to be executed as the
"root" user in each cluster node.
/oracle/grid/root.sh #On nodes rac2
To execute the configuration scripts:
    1. Open a terminal window
    2. Log in as "root"
```

```
    3. Run the scripts in each cluster node

The Cluster Node Addition of /oracle/grid was successful.
Please check '/tmp/silentInstall.log' for more details.
=====================================================
```

17.4 典型的 RAC+Data Guard 高可用环境 MAA

实验环境的操作系统环境为 Oracle Linux Enterprise Edition 5.0，内核版本为 2.6.39-400.209.1.el5uek，安装了 GI 和 DBMS 所必需的软件包，安装和配置了 ASMLib 驱动。

系统中创建了用户组 oinstall、dba、asmadmin、asmdba 及其 grid、oracle 用户，并配置了用户环境，代码如下：

```
oinstall    grid, oracle        Oracle Inventory and Software
Owner
dba                 oracle              Database Administrator
asmadmin    grid                        Oracle Automatic Storage
Management Group
asmdba      grid, oracle        ASM Database Administrator
Group
```

系统通过了 GI 安装前的检查，代码如下：

```
[grid@host3]$ ./runcluvfy.sh stage -pre crsinst -n host3 -
verbose
......
Pre-check for cluster services setup was successful.
```

17.4.1 RAC 主数据库端环境描述

此部分构造的是一个典型的 RAC+DG 的环境：主数据库端为 RAC，备用端为单实例数据库。

1. 主机及其网络信息

```
############################
127.0.0.1 localhost

# public
188.168.0.101 rac1
```

```
188.168.0.102 rac2

# private
192.168.1.101 rac1-prv
192.168.1.102 rac2-prv

# vip
188.168.0.111 rac1-vip
188.168.0.112 rac2-vip

# scan
188.168.0.123 rac-scan
############################
```

2．RAC 数据库信息

```
=====================================================
[oracle@rac1 ~]$ srvctl config database -d BJING
Database unique name: BJING
Database name:
Oracle home: /oracle/dbms
Oracle user: oracle
Spfile: +datgrp/bjing/parameterfile/spfile.267.866461989
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: BJING
Database instances: BJING1,BJING2
Disk Groups:
Services:
Database is administrator managed
=====================================================
[grid@rac1 ~]$ srvctl status database -d BJING
Instance BJING1 is running on node rac1
Instance BJING2 is running on node rac2
=====================================================
SQL> select inst_id,group#,thread#,sequence#,members,archived,
```

```
  2  status from gv$log;

INST_ID     GROUP#    THREAD#  SEQUENCE#    MEMBERS ARC STATUS
----------------------------------------------------------
1           1         2        1            1 NO  INACTIVE
1           2         2        2            1 NO  CURRENT
1           3         1        85           1 NO  CURRENT
1           4         1        84           1 NO  INACTIVE
2           1         2        1            1 NO  INACTIVE
2           2         2        2            1 NO  CURRENT
2           3         1        85           1 NO  CURRENT
2           4         1        84           1 NO  INACTIVE

8 rows selected.

SQL> archive log list;
Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     1
Current log sequence            2
```

3．主数据库的配置调整

调整的内容主要有 3 个方面：归档模式、强制日志、必须使用口令文件。集群数据库的调整和单实例数据库类似，仅在一个节点上执行（该节点以 exclusive 方式打开，即参数 cluster_database=false），关闭其他所有实例，否则，将出现如下提示：

```
SQL> alter database archivelog;
alter database archivelog
*
ERROR at line 1:
ORA-01126: database must be mounted in this instance and not
open in any instance.

SQL> startup mount
ORACLE instance started.
```

```
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             390071944 bytes
Database Buffers          138412032 bytes
Redo Buffers                5840896 bytes
Database mounted.
SQL> alter database archivelog;
Database altered.

SQL> alter database force logging;
Database altered.

SQL> alter database open;
Database altered.

SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     84
Next log sequence to archive   85
Current log sequence                  85
```

不同实例口令文件的处理如下：

```
[oracle@rac1 ~]$ cd /oracle/dbms/dbs
[oracle@rac1 dbs]$ scp orapwBJING1 rac2:/oracle/dbms/dbs
orapwBJING1                                   100% 1536     1.5KB/
s   00:00
[oracle@rac1 dbs]$ ssh rac2
Last login: Mon Dec 15 22:09:24 2014 from rac1
[oracle@rac2 ~]$ cd /oracle/dbms/dbs
[oracle@rac2 dbs]$ mv orapwBJING1 orapwBJING2
```

17.4.2　为创建备用数据库做准备

1．对主数据库执行备份

```
[oracle@rac1 ~]$ rman target /
```

```
Recovery Manager: Release 11.2.0.1.0 - Production on Wed Dec
17 10:15:24 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (DBID=3050489608)

RMAN> backup format '/oracle/bak/%U' database;
Starting backup at 2014-12-17 10:15:59
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=63 instance=BJING1 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=+DATGRP/bjing/datafile/system.257.866391257
input datafile file number=00002
name=+DATGRP/bjing/datafile/sysaux.258.866391261
input datafile file number=00004
name=+DATGRP/bjing/datafile/undotbs1.259.866391265
input datafile file number=00005
name=+DATGRP/bjing/datafile/undotbs2.265.866459207
input datafile file number=00003
name=+DATGRP/bjing/datafile/example.260.866391265
channel ORA_DISK_1: starting piece 1 at 2014-12-17 10:16:03
channel ORA_DISK_1: finished piece 1 at 2014-12-17 10:16:10
piece handle=/oracle/bak/09pqcov3_1_1 tag=TAG20141217T101602
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:07
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
including current control file in backup set
including current SPFILE in backup set
channel ORA_DISK_1: starting piece 1 at 2014-12-17 10:16:11
channel ORA_DISK_1: finished piece 1 at 2014-12-17 10:16:12
piece handle=/oracle/bak/0apqcova_1_1 tag=TAG20141217T101602
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time:
00:00:01
```

```
Finished backup at 2014-12-17 10:16:12
```

2．创建备用数据库的控制文件（可选）

```
RMAN> backup as copy format '/oracle/bak/shhaictl.ora' current
controlfile for standby;

Starting backup at 2014-12-17 10:21:09
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
copying standby control file
output file name=/oracle/bak/shhaictl.ora tag=TAG20141217T102110
RECID=10 STAMP=866542871
channel ORA_DISK_1: datafile copy complete, elapsed time:
00:00:01
Finished backup at 2014-12-17 10:21:12
```

3．传输文件至备用端

```
[oracle@rac1 ~]$ cd /oracle/bak
[oracle@rac1 bak]$ scp 09pqcov3_1_1 188.168.0.103:/oracle/bak
The authenticity of host '188.168.0.103 (188.168.0.103)' can't
be established.
RSA key fingerprint is
a1:af:16:4b:82:b6:da:bf:a7:12:30:37:45:eb:08:11.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '188.168.0.103' (RSA) to the list
of known hosts.
oracle@188.168.0.103's password:
09pqcov3_1_1                                  100%  237MB   3.2MB/s
01:14
[oracle@rac1 bak]$ scp 0apqcova_1_1 188.168.0.103:/oracle/bak
[oracle@rac1 bak]$ scp shhaictl.ora 188.168.0.103:/oracle/bak

=====================================================
[oracle@rac1 bak]$ cd /oracle/dbms/dbs
[oracle@rac1 dbs]$ scp orapwBJING1 188.168.0.103:/oracle/dbms/
dbs
oracle@188.168.0.103's password:
```

```
orapwBJING1                                   100% 1536     1.5KB/
s   00:00
[oracle@rac1 dbs]$ ssh host3
ssh: host3: Temporary failure in name resolution
[oracle@rac1 dbs]$ ssh 188.168.0.103
oracle@188.168.0.103's password:
Last login: Fri Mar  7 16:20:47 2014 from rac1
[oracle@unx3 ~]$ cd /oracle/dbms/dbs
[oracle@unx3 dbs]$ ls
init.ora  orapwBJING1
[oracle@unx3 dbs]$ mv orapwBJING1 orapwSHHAI
=====================================================
```

17.4.3 启动备用端实例

1. 备用端主机信息

备用端已安装 Oracle DBMS 软件（仅安装数据库软件），并创建了默认监听。

```
[root@unx3 ~]# cat /etc/hosts
############################
127.0.0.1 localhost

# public
188.168.0.103 unx3
...
############################
```

用户 Oracle 的环境变量设置：

```
ORACLE_SID=SHHAI
......
```

2. 通过备用控制文件加载实例

（1）准备参数文件，代码如下：

```
=====================================================
*.db_name='jiadp'
*.db_unique_name=SHHAI
*.instance_name=SHHAI
*.memory_target=512M
*.db_block_size=8192
```

```
*.control_files='/oracle/dat/SHHAI/shhaictl.ora'
*.db_create_file_dest='/oracle/dat'
*.db_recovery_file_dest='/oracle/fra'
*.db_recovery_file_dest_size=1G
*.log_archive_dest_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
*.diagnostic_dest='/oracle/base'
*.open_cursors=300
*.processes=150
*.remote_login_passwordfile='EXCLUSIVE'
*.compatible='11.2.0.1.0'
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS1'
*.local_listener=
'(ADDRESS=(PROTOCOL=TCP)(HOST=188.168.0.103)(PORT=1521))'
=====================================================
```

（2）启动实例 SHHAI，代码如下：

```
[oracle@unx3 SHHAI]$ sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Wed Dec 17 11:06:26
2014
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to an idle instance.

SQL> create spfile from pfile='/oracle/dat/initSHHAI.ora';
File created.

SQL> startup mount
ORACLE instance started.
Total System Global Area  535662592 bytes
Fixed Size                  1337720 bytes
Variable Size             327157384 bytes
Database Buffers          201326592 bytes
Redo Buffers                5840896 bytes
Database mounted.
SQL> select database_role from v$database;

DATABASE_ROLE
----------------
PHYSICAL STANDBY
```

3．配置主备用网络环境

主数据库端节点 rac1 和 rac2 分别配置相同的网络服务，代码如下：

```
=====================================================
SHHAI =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.103)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = SHHAI)
    )
  )
=====================================================
```

备用数据库端配置针对集群的网络服务，代码如下：

```
=====================================================
BJING =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.101)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.111)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.102)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.112)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = BJING)
    )
  )

BJING1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
```

```
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.101)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.111)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = BJING)
    )
  )

BJING2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.102)(PORT =
1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 188.168.0.112)(PORT =
1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = BJING)
    )
  )
=====================================================
```

4. 为 Data Guard 设置参数

主数据库端参数设置如下：

```
=====================================================
*.LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'
*.LOG_ARCHIVE_DEST_2 = 'SERVICE=SHHAI LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=SHHAI'
#####################################################
*.FAL_SERVER = SHHAI
*.STANDBY_FILE_MANAGEMENT = AUTO
*.DB_FILE_NAME_CONVERT = '/oracle/dat/SHHAI', '+DATGRP/bjing'
*.LOG_FILE_NAME_CONVERT = '/oracle/fra/SHHAI', '+FRAGRP/bjing'
=====================================================
```

备用端参数设置如下：

```
=====================================================
# CONTROL_FILES='......'
#####################################################
*.LOG_ARCHIVE_CONFIG = 'DG_CONFIG=(BJING, SHHAI)'
*.FAL_SERVER = BJING
*.STANDBY_FILE_MANAGEMENT = AUTO
*.DB_FILE_NAME_CONVERT = '+DATGRP/bjing', '/oracle/dat/SHHAI'
*.LOG_FILE_NAME_CONVERT = '+FRAGRP/bjing', '/oracle/fra/SHHAI'
#####################################################
*.LOG_ARCHIVE_DEST_2 = 'SERVICE=BJING1 LGWR ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=BJING'
=====================================================
```

17.4.4 在备用端还原数据库

以新的参数设置重新启动备用实例 SHHAI 至 nomount 状态。如果创建并传输了备用控制文件，则可启动至 mount 状态，然后手工还原数据库。

注意，使用复制的方法来创建备用数据库，备用实例必须在 nomount 状态：

```
RMAN-05500: the auxiliary database must be not mounted when
issuing a DUPLICATE command.
```

确认主数据库端的数据库备份已经传输到备用端，并且备用端的相关目录结构需要预先建立，并且 oracle 用户需要有读 / 写权限。

```
[oracle@unx3 bak]$ ls
0hpqe1u4_1_1  0ipqe1uk_1_1
[oracle@unx3 bak]$ rman target sys/internal@bjing auxiliary /
Recovery Manager: Release 11.2.0.1.0 - Production on Wed Dec
17 22:17:49 2014
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All
rights reserved.
connected to target database: JIADP (DBID=3050489608)
connected to auxiliary database: JIADP (not mounted)

RMAN> duplicate target database for standby;
Starting Duplicate Db at 2014-12-17 22:18:18
using target database control file instead of recovery catalog
```

```
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=20 device type=DISK

contents of Memory Script:
{
   sql clone "alter system set  control_files =
   ''/oracle/dat/SHHAI/controlfile/o1_mf_b933opq4_.ctl'', ''/
oracle/fra/SHHAI/controlfile/o1_mf_b933oq3s_.ctl'' comment=
 ''Set by RMAN'' scope=spfile";
   restore clone standby controlfile;
}
executing Memory Script

sql statement: alter system set  control_files =
''/oracle/dat/SHHAI/controlfile/o1_mf_b933opq4_.ctl'', ''/
oracle/fra/SHHAI/controlfile/o1_mf_b933oq3s_.ctl'' comment=
''Set by RMAN'' scope=spfile

Starting restore at 2014-12-17 22:18:23
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: reading from backup piece
/oracle/bak/0ipqe1uk_1_1
channel ORA_AUX_DISK_1: piece handle=/oracle/bak/0ipqe1uk_1_1
tag=TAG20141217T215515
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time:
00:00:01
output file name=/oracle/dat/SHHAI/controlfile/o1_mf_b933opq4_.
ctl
output file name=/oracle/fra/SHHAI/controlfile/o1_mf_b933oq3s_.
ctl
Finished restore at 2014-12-17 22:18:25

contents of Memory Script:
{
```

```
   sql clone 'alter database mount standby database';
}
executing Memory Script

sql statement: alter database mount standby database

contents of Memory Script:
{
   set newname for tempfile  2 to
 "/oracle/dat/SHHAI/tempfile/temptbs.263.866391695";
   switch clone tempfile all;
   set newname for datafile  1 to
 "/oracle/dat/SHHAI/datafile/system.257.866391257";
   set newname for datafile  2 to
 "/oracle/dat/SHHAI/datafile/sysaux.258.866391261";
   set newname for datafile  3 to
 "/oracle/dat/SHHAI/datafile/example.260.866391265";
   set newname for datafile  4 to
 "/oracle/dat/SHHAI/datafile/undotbs1.259.866391265";
   set newname for datafile  5 to
 "/oracle/dat/SHHAI/datafile/undotbs2.265.866459207";
   restore
   clone database
   ;
}
executing Memory Script

executing command: SET NEWNAME

renamed tempfile 2 to
/oracle/dat/SHHAI/tempfile/temptbs.263.866391695 in control file

executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
```

```
Starting restore at 2014-12-17 22:18:35
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/oracle/dat/SHHAI/datafile/system.257.866391257
channel ORA_AUX_DISK_1: restoring datafile 00002 to
/oracle/dat/SHHAI/datafile/sysaux.258.866391261
channel ORA_AUX_DISK_1: restoring datafile 00003 to
/oracle/dat/SHHAI/datafile/example.260.866391265
channel ORA_AUX_DISK_1: restoring datafile 00004 to
/oracle/dat/SHHAI/datafile/undotbs1.259.866391265
channel ORA_AUX_DISK_1: restoring datafile 00005 to
/oracle/dat/SHHAI/datafile/undotbs2.265.866459207
channel ORA_AUX_DISK_1: reading from backup piece /oracle/
bak/0hpqe1u4_1_1
channel ORA_AUX_DISK_1: piece handle=/oracle/bak/0hpqe1u4_1_1
tag=TAG20141217T215515
channel ORA_AUX_DISK_1: restored backup piece 1
channel ORA_AUX_DISK_1: restore complete, elapsed time:
00:00:15
Finished restore at 2014-12-17 22:18:51

contents of Memory Script:
{
   switch clone datafile all;
}
executing Memory Script

datafile 1 switched to datafile copy
input datafile copy RECID=12 STAMP=866585932 file
name=/oracle/dat/SHHAI/datafile/system.257.866391257
datafile 2 switched to datafile copy
input datafile copy RECID=13 STAMP=866585932 file
name=/oracle/dat/SHHAI/datafile/sysaux.258.866391261
datafile 3 switched to datafile copy
```

```
input datafile copy RECID=14 STAMP=866585932 file
name=/oracle/dat/SHHAI/datafile/example.260.866391265
datafile 4 switched to datafile copy
input datafile copy RECID=15 STAMP=866585932 file
name=/oracle/dat/SHHAI/datafile/undotbs1.259.866391265
datafile 5 switched to datafile copy
input datafile copy RECID=16 STAMP=866585932 file
name=/oracle/dat/SHHAI/datafile/undotbs2.265.866459207
Finished Duplicate Db at 2014-12-17 22:18:55
```

17.4.5 启动托管恢复

```
SQL> select name from v$controlfile;

NAME
---------------------------------------
/oracle/dat/SHHAI/controlfile/o1_mf_b933opq4_.ctl
/oracle/fra/SHHAI/controlfile/o1_mf_b933oq3s_.ctl
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      0
Next log sequence to archive    0
Current log sequence                  0
SQL> alter database recover managed standby database
disconnect;
Database altered.
......
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence      96
Next log sequence to archive    0
Current log sequence            97
```

17.4.6　主备用角色切换

这里的角色切换的基本假设如下：主数据库处于 Open 状态、物理备用数据库处于正常的托管恢复 mount 状态。

角色切换过程中，不论是主数据库端还是备用数据库端，如果是 RAC 集群，只能保留一个节点运行（切换完成后，其他节点可正常打开）。

（1）主数据库操作如下：

```
SQL> alter database commit to switchover to physical standby
[with session shutdown];
```

当主数据库有其他用户连接时，使用 with session shutdown 选项。

重新启动，并加载数据库。此时，主数据库已转换为物理备用角色。

（2）物理备用数据库操作如下：

```
SQL> alter database commit to switchover to primary;
```

重新启动实例，打开数据库即可。此时，原来的物理备用数据库已经转换为新的主数据库。

（3）启动新的托管恢复。在新的物理备用数据库上启动托管恢复即可。

上面的切换过程称为 Switchover。当主数据库故障或不可用时，可将物理备用数据库转换为主数据库，称为 Failover，此时，在将备用数据库转换为主数据库之前，需要应用已接收的重做日志，并结束其托管恢复状态，代码如下：

```
SQL> alter database recover managed standby database finish;
```

接下来，执行角色转换操作，正常打开数据库即可，代码如下：

```
SQL> alter database commit to switchover to primary;
SQL> alter database open;
=====================================================
```

参考文献

[1] Oracle High Availability Overview, 11g Release 2 (11.2), Part Number E10804-03, 2011.
[2] Oracle Real Application Clusters Administration and Deployment Guide 11g Release 2 (11.2), Part Number E10718-11, 2012.
[3] Oracle Clusterware Administration and Deployment Guide 11g Release 2 (11.2), Part Number E10717-10, 2012.
[4] Oracle Data Guard Concepts and Administration 11g Release 2 (11.2) Part Number E10700-02, 2013.
[5] Oracle Advanced Replication Management API Reference, 11g Release 2 (11.2), Part Number E10707-02, 2014.
[6] Oracle Grid Infrastructure Installation Guide, 11g Release 2 (11.2) for Linux, Part Number E10812-04, 2011.
[7] Scott Jesse, Bryan Vongray, Bill Burton. High Availability: Maximize Your Availability with Grid Infrastructure, Oracle Real Application Clusters, and Oracle Data Guard, The McGraw Hill Companies, 2012.
[8] Bob Bryla, Kevin Loney. Oracle Database 12c: The Complete Reference, Oracle Press, 2013.
[9] http://www.Oracleinaction.com/category/12c-dataguard-index/.
[10] http://www.dbauniversity.com/Oracle_rac_asm_dg_training.html.
[11] Tom Plunkett, Brian Macdonald, Bruce Nelson，许向东 等译 .Oracle 大数据解决方案 . 北京：清华大学出版社，2015.01.
[12] 李华植，郑保卫等 . 海量数据库解决方案 . 北京：电子工业出版社，2011.
[13] 贾代平，吴丽娟 .Oracle DBA 核心技术解析 . 北京：电子工业出版社，2006.
[14] 贾代平 . 基于日志的数据恢复及其在 Oracle 中的实现 . 计算机工程与设计 2004.12.
[15] 贾代平，范洪达 . 基于信源熵的残差相关性度量方法 . 计算机学报，2005.10.
[16] 贾代平，范洪达 . 基于分数阶差分模型的记忆性扩展方法 . 通信学报，2006.9.
[17] 贾代平，吴占鳌 .Oracle 网络数据库管理与应用 . 北京：石油工业出版社，2010.
[18] 贾代平，吴丽娟 .Oracle 数据存储与访问技术 . 北京：电子工业出版社，2013.06.
[19] 赵刚 . 大数据技术与应用实践指南 . 北京：电子工业出版社，2013.10.
[20] 邹恒明 . 有备无患：信息系统之灾难应对 . 北京：机械工业出版社，2009.01.
[21] 卜海兵，徐明远，杨宏桥 . 数据存储、恢复与安全应用实践 . 北京：中国铁道出版社，2012.05.
[22] 吴朱华 . 云计算核心技术剖析 . 北京：人民邮电出版社，2011.05.
[23] Jothy rosenberg, arthur mateos, 胡建 译 . 企业实施云计算的核心问题 . 北京：机械工业出版社，2012.06.