



Oracle 技术系列丛书

ORACLE



Oracle Backup & Recovery 101

Oracle

备份与恢复 培训教程

(美) Kenny Smith 著 周琦 韩岷 李渝琳 等译
Stephan Haisley

Your
Oracle
Career
Starts
Here

FOR FIRST-TIME ORACLE USERS



机械工业出版社
China Machine Press



Education

Oracle 备份与恢复 培训教程

本书以独特的类比方式开头，介绍了数据库备份与恢复的基本知识。其幽默、睿智的语言一定会使读者对此主题有全新的认识。本书分别讲解了用户管理的备份与恢复和RMAN的备份与恢复。通过各种应用示例的讲解，使读者在理论学习的同时，能进行实际的练习，为以后在实际的运行环境中工作积累经验和自信。书中提供了大量经典的SQL文件、Linux脚本、Windows脚本和RMAN脚本，Oracle DBA可以在这些脚本的基础上编写符合自己实际需求的脚本和命令。书中对Oracle提供的RMAN的详细讲解，对于提高工作效率、更好地保护数据库都具有重大意义。本书的作者具有极丰富的实践和理论经验。本书通俗易懂，实用性强，是Oracle DBA 学习数据库备份与恢复的首选资料。

主要内容摘要

- 用离线备份与日志处理用户管理的备份与恢复
- 当数据库打开时，进行联机备份
- 为开发和测试进行复制和克隆Oracle数据库
- 为提高可用性创建备用数据库
- 用可迁移表空间进行表空间备份
- 分析离线重做日志文件来发现逻辑问题
- 使用RMAN简化备份与恢复工作

适用水平：中、高级



(Oracle 9i 初学者指南)

书号：ISBN 7-111-02927
定价：45.00元



(Oracle 8i 备份与恢复手册)

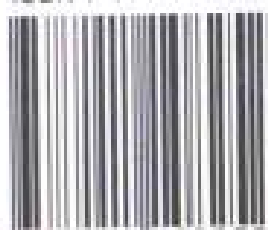
书号：ISBN 7-111-03052
定价：55.00元



(Oracle 9i DBA手册)

书号：ISBN 7-111-10145
定价：88.00元

ISBN 7-111-10825-6



9 787111 108252



华章图书

网上购书：www.china-pub.com

北京市西城区百万庄南街1号 100037

购书热线：(010)68995259、8006100280 (北京地区)

总编信箱：chiefeditor@hzbook.com

ISBN 7-111-10825-6/TP · 2573

定价：35.00 元

Mc
Graw
Hill

Oracle技术系列丛书

Oracle 备份与恢复 培训教程

(美) Kenny Smith 著
Stephan Haisley
周琦 韩岷 李渝琳 等译



机械工业出版社
China Machine Press

本书以独特的类比方式开头,对数据库备份与恢复知识进行了简要的介绍,然后分别讲解了用户管理的备份与恢复和RMAN的备份与恢复。各章主题鲜明、注重实用,通过各种应用示例,使读者在理论学习的同时进行实践,为以后在实际的运行环境中工作积累经验和自信。书中提供了许多经典的脚本和运行命令,Oracle DBA可以在这些脚本的基础上编写符合自己实际需求的脚本和命令。尤其是本书对RMAN进行了详细讲解,掌握新的工具对于提高工作效率,更好地保护数据库都具有重大意义。

本书通俗易懂、实用性强,是Oracle DBA学习数据库备份与恢复的参考书。

Kenny Smith and Stephan Haisley: Oracle Backup & Recovery 101.

Copyright © 2002 by The McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved.
No part of this publication may be reproduced or distributed in any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition jointly published by McGraw-Hill Education (Asia) Co. and China Machine Press.

本书中文简体字版由美国麦格劳-希尔教育出版公司授权机械工业出版社出版,未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有McGraw-Hill公司防伪标签,无标签者不得销售。

版权所有,侵权必究。

本书版权登记号:图字:01-2002-0805

图书在版编目(CIP)数据

Oracle备份与恢复培训教程/(美)史密思(Smith, K.)等著;周琦等译. -北京:机械工业出版社,2002.8

(Oracle技术系列丛书)

书名原文:Oracle Backup & Recovery 101

ISBN 7-111-10825-6

I. O… II. ①史… ②周… III. 关系数据库-数据库管理系统, Oracle IV. TP311.138

中国版本图书馆CIP数据核字(2002)第062689号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑:张碧霞 张鸿斌

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2002年9月第1版第1次印刷

787mm×1092mm 1/16·20.75印张

印数:0 001-4 000册

定价:35.00元

凡购本书,如有倒页、脱页、缺页,由本社发行部调换

译 者 序

本书是一本不可多见的Oracle数据库备份与恢复的好书。本书以独特的类比方式开头，作为对数据库备份与恢复知识的简要介绍。其幽默、睿智的语言一定会使读者重新认识数据库的备份与恢复。接着本书分别讲解了用户管理的备份与恢复和RMAN的备份与恢复。各章主题鲜明、注重实用。通过各种应用示例，使读者在理论学习的同时，能进行实践练习，为以后在实际的运行环境中工作积累经验和自信。书中提供了许多经典的脚本和运行命令，Oracle DBA可以在这些脚本的基础上编写符合自己实际需求的脚本和命令。尤其要注意书中对Oracle提供的RMAN的详细讲解，掌握新的工具对于提高自己的工作效率、更好地保护数据库都具有重大意义。

本书各章节的习题具有很强的代表性，使读者在回答问题的同时能增强对概念和实际操作的理解。各种技巧和注意事项将有助于读者以安全、高效地方式完成学习，从而使读者可以逐步掌握Oracle数据库备份与恢复的理论和实际操作能力。本书的作者具有极丰富的理论和实践经验。本书通俗易懂、实用性强，是Oracle DBA学习数据库备份与恢复的首选资料。

本书主要由周琦、韩岷、李渝琳等翻译。参加翻译和整理工作的还有周珩、周玮、吴伟强、高志刚、白萌、肖雄兵、王知学等。胡蓉、田震龙、李大鹏、林源、李同友、梁华等也对本书的翻译和校对付出了大量辛勤的劳动。

由于时间紧迫和译者水平有限，书中难免出现错误，敬请读者批评指正。

2002年5月

1. 5-21

前 言

嗨，新的DBA们，很高兴你们挑选了这本书！

如果你希望成为一名优秀的DBA，那么就需要学好如何优化SQL、配置数据库、编写出色的PL/SQL、实现并测试一个很好的备份与恢复策略。

如果你希望成为一个糟糕的DBA，那么就不必备份你的数据库，或者备份了之后，而不学习如何从备份中恢复。另一个糟糕的事情是：不测试你的备份是否可以用于恢复。这样系统宕机或数据丢失将会使你陷入困境。

本书只是Oracle出版社出版的其中一本。或许目前你的书架上已经放满了Oracle的书，但因为某些原因，你并没有发现更多的关于备份与恢复方面的书，而大量的书都是关于普通DBA必备技能的书，例如优化是一个重要的专题。如果你希望学习PL/SQL，那么你有许多可供选择的优秀参考书，但是备份与恢复并没有多少可供选择的资料。我曾经为了一个Oracle备份与恢复的简单指导而找遍书店的整个书架。我希望有一些简单的示例，还有很多情况下，我需要的是我所阅读内容的练习指导，但我从没有在任何Web站点或书店中找到类似的书。

于是，我决定写一本这样的书。

如何使用本书

关于这本书，可以使用下面三种阅读方式：

- 阅读本书可以学习Oracle数据库备份与恢复。本书对属于数据库保护的概念和术语给出了解释并以简单的形式进行了定义。本书包含了基本的命令和脚本，这些都可以用于保护数据库及其内容。
- 通过实践学习本书才能理解Oracle数据库备份与恢复。更好地掌握本书的一个方法是检查输入的命令和屏幕输出并与本书的输出结果进行比较。第二、三部分的各章中都有一个关于要完成任务的介绍和定义。然后提供可以在家中完成的练习，一台简单的PC机就足以操作本书的所有练习。在Linux或Windows的机器上创建一些数据库，安装Oracle 8.1.7版本就可以工作了。各章节的专项练习将提供逐步指导以完成每步练习并检验结果。当完成了所有的步骤后，就可以准备在真正的Oracle数据库系统上实际操作该功能。当你要完成一项类似的任务时，可以参考各章节的相关内容。
- 当需要执行Oracle数据库备份与恢复任务时，可以参考本书。我试着编写一些简单的“处方”，可以作为Oracle备份与恢复任务的指导。这些简单的命令和脚本可以建立在产品环境上。

专项练习主要是为了加强理解各个章节学过的概念和术语。本书并不是一个考试的辅导材料，但读者可以实际操作所学的内容以准备OCP考试。更重要的是，将所学知识运用到工作中将会获得自信。

本书分为三个部分。第一部分提供了对基本概念和一些可用于保护Oracle数据库的选项的概述，同时介绍了如何建立数据库以供第二、三部分各章中练习需要。本书的第二部分涵盖了用

户管理备份与恢复专题。第三部分则是服务器管理备份与恢复专题。第三部分的许多练习的功能类似于第二部分练习。因此，可以比较使用RMAN和不使用RMAN完成任务的情况。

第一部分——备份与恢复概述

这部分的各章节介绍了Oracle数据库备份与恢复实质性特性的概念。第1章对每日的家庭生活和Oracle数据库之间进行了一个独特的类比。第1章介绍了一些基础知识，第2、3章为学习本书后续部分提供了准备知识。

第1章：保护数据

Oracle数据库备份与恢复是确保系统成功的一项核心技术和必要方法。举一个不太确切的类比，比如个人照片集，就可以对Oracle数据库的内容有较好地理解，还可以理解数据库体系构件之间的内在联系。该章覆盖了备份与恢复重要的基本概念和Oracle体系结构。

第2章：备份与恢复选项

Oracle为备份数据和数据库结构提供了许多选项。每种方法在保护数据库方面都有各自独特的价值和用途。该章对这些不同的选项提供了一个简要介绍，并预先介绍了本书的后续内容。

第3章：专项练习

许多Oracle数据库专家都有关于恢复的理论知识，但有些人并没有在实际系统上操作恢复的经验。自己要准备挑战下一次重要的恢复任务，如何将自己置身于恢复的现场中将在本书第二、三部分讨论。该章中，读者将要创建一个作为练习场所的数据库，并会对作为Oracle恢复管理者而感到欣慰。

第二部分——用户管理的备份与恢复

数据库是由需要保护的各种文件组成的。这部分的各章提供了关于Oracle用户管理恢复方法的描述。物理备份与恢复操作涉及到使用数据库文件来保护数据库并在数据库崩溃后用于修复。在第4章中，我们将讨论在一个关闭的数据库上进行用户管理备份与恢复。在第5章中，我们描述了在一个打开的数据库上进行用户管理备份与恢复。第6章中我们将在数据库关闭和打开时克隆数据库。在第7章中，创建一个备用数据库。第8章主要描述Oracle导出/导入工具提取和创建数据对象。在第9章中，我们展示如何执行用户管理表空间时间点恢复。表空间恢复将使用Oracle导出/导入并将在数据库之间迁移表空间。最后，在第10章中，我们讨论了如何使用Log Miner来挖掘重做日志文件的SQL语句。

第4章：关闭数据库的备份与恢复

Oracle数据库其实是一系列相互交互的文件的集合。Oracle服务器与这些文件交互以便提供应用程序和Web页面需要的数据。可以在数据库关闭时通过拷贝这些文件来保护数据。本章中，我将描述如何备份数据库中的数据库文件。然后应用这些备份，在数据库某些部分损坏时进行还原与恢复数据库。通过配置数据库归档所有的重做（redo）文件，可以完全恢复数据库或恢复到某一点。

第5章：从打开的数据库备份与恢复

当数据库的可用性变得非常重要时，许多数据库维护操作必须在打开着并向用户提供服务的数据库上进行。所有的用户管理备份操作与许多恢复操作可以在数据库正提供服务时进行。本章中，将要介绍如何备份打开的数据库，然后介绍数据库打开时如何进行完全恢复。最后，

介绍使用打开的数据库备份进行不完全的恢复。

第6章：复制数据库

复制数据库是完全数据库恢复的严格测试。可以将数据库复制到同一台机器或复制到其他机器上。通过重新生成控制文件或使用控制文件的拷贝，用一个备份替换文件就可以确定是否可以恢复备份。本章我将描述如何在同一台机器上复制关闭的和打开的数据库，还将介绍如何将复制数据库中所学的知识应用到其他场景中。

第7章：备用数据库

如果源数据库不稳定时，可以采用备用数据库。备用数据库就是原始的源数据库的一个相同拷贝。备用数据库经常通过应用重做日志文件来接替源数据库工作。在本章中，将介绍应用用户管理备份与恢复技术来配置备用数据库。以只读方式打开一个备用数据库并激活它。

第8章：导出与导入

使用导出可以从数据库中析取数据，使用导入工具可将这些数据插入到同一数据库或其他数据库中。使用Oracle的导出生成的SQL命令可以被用于重新生成相同的逻辑数据拷贝到一个二进制文件中。使用Oracle导入可以从二进制文件中重新生成部分或全部数据。在该章中，将要导出数据到一个文件中并从这个文件中导入数据。示例操作将演示这些便捷工具的不同方法。

第9章：表空间时间点恢复

Oracle数据库可以被部分恢复。应用一种被称为表空间时间点恢复的技术可以将非系统表空间恢复到以前的状态。本章将详细介绍如何操作这种有用的方法以及在什么情况下可以应用这种备份与恢复技术。表空间恢复将要使用到数据文件备份、重做应用、导出与导入等。

第10章：LogMiner

所有数据库事务都被存储到日志文件中。可以挖掘这些文件以得到包含在其中的SQL命令。LogMiner提供了这种从日志文件中提取SQL命令的方法。可以从重做日志文件中重新获得事务性语句，而这些语句可以被重新应用到数据库上。该章将分析重做日志的内容。使用重新获得的事务，可以将SQL语句应用到数据库上并探究如何使用SQL语句。

第三部分——服务器管理恢复

服务器管理恢复涉及到使用恢复管理器（RMAN）来执行备份与恢复操作。因为备份与恢复数据库系统非常重要，所以Oracle为数据库管理员提供了这种处理各种细节的工具。本书用了七章的篇幅来讲解这种保护数据库的新工具。这些章节将会帮助一名新的DBA获得基本的RMAN配置与应用方面的经验。这七章也将帮助老DBA从传统的尝试-成功的用户管理技术升迁到新的技术。在第二部分讨论的大多数任务都可以用RMAN简单迅速地完成任务。因此，可以容易地比较执行一个特殊的任务采用的用户管理方法和具有同样功能的服务器管理方法。

第11章：RMAN配置

恢复管理器将部分或所有数据库文件分级备份。可以使用恢复目录来跟踪RMAN备份以及目标数据库的结构。在磁带上备份需要几个步骤来准备磁带驱动器以接收RMAN备份。该章将讨论如何使用恢复目录来配置RMAN用磁盘或磁带备份。

第12章：RMAN备份

用RMAN完成备份工作只需要几个简单命令。可以备份数据文件、归档日志和控制文件。

可以备份所有数据块或自前次备份以来发生变化的数据块。可以用存储脚本来执行备份操作。RMAN提供了许多特性，可以最大程度缩短备份数据库或还原恢复的时间。我们还将讨论监视与并行备份。这里还有一个基本的RMAN备份策略，可以用于自己的RMAN部署。

第13章：RMAN目录（catalog）维护

当用RMAN部署了一个备份策略后，那么这个部署将需要一些事务性维护。对于快速和完全的恢复还需要校验数据库是否有足够的备份。RMAN备份可以累计花费的时间。将要删除不需要的备份文件。在本章中，将介绍RMAN目录（catalog）的维护。

第14章：RMAN恢复

“恢复管理器”的功能是修复一个损坏的数据库。假设已经有了很完善的备份，那么恢复操作就是小菜一碟。可以恢复部分或全部数据库。当然也可以恢复到过去某一点的位置。可以并行恢复。只需要几个命令就可以完成恢复工作。RMAN将会从不同类别的数据库失效情况下执行恢复操作。该章将讨论如何用RMAN执行不同的恢复。

第15章：RMAN复制数据库

恢复管理器还可以从RMAN备份来复制数据库。该章将介绍RMAN向同一台机器上复制数据库，还将介绍RMAN复制操作的一些技巧和技术。该章用RMAN完成与第6章中同样的任务。

第16章：RMAN备用数据库

RMAN能创建备用数据库。该章将学习创建的方法。该章用RMAN完成与第7章中同样的任务。

第17章：RMAN表空间恢复

RMAN可以执行表空间时间点恢复。该章将学习执行的方法。该章用RMAN完成与第9章中同样的任务。

第四部分——附录

这两个附录将对理解本书有帮助：

- 附录A提供了重要的备份与恢复术语集。
- 附录B提供了在第三部分介绍的操作RMAN的快速示例和讲解，并说明了如何在Oracle 9i数据库中应用RMAN。

本书包括了大量的SQL文件、Linux脚本、Windows脚本和RMAN脚本。我们将这些脚本放在Oracle出版社的Web站点上（www.oraclepressbooks.com），可以在做这些练习时自由下载。我们力图写出一本实用、易学而又精确的书。如果读者发现什么问题，或有任何建议、评论，请给我们发送e-mail（OracleBackupRecovery@yahoo.com）。Oracle出版社的Web站点的Errata部分将定期公布错误或澄清某些问题。

目 录

译者序

前言

第一部分 备份与恢复概述

第1章 保护数据	1
1.1 介绍Sid的家	2
1.1.1 Sid的家	3
1.1.2 Debbie的家事	3
1.1.3 Logan的家事	4
1.1.4 Archie的家事	5
1.1.5 Chuck的家事	5
1.1.6 控制笔记本	6
1.1.7 告警日志	6
1.1.8 清晨程序	6
1.1.9 夜间程序	7
1.1.10 Sid的问题	8
1.2 Oracle服务器	8
1.2.1 数据库的内容	8
1.2.2 内存的体系结构	9
1.2.3 数据库的体系结构	9
1.3 Oracle数据库文件	11
1.3.1 数据文件	11
1.3.2 联机重做日志文件	11
1.3.3 归档重做日志文件	12
1.3.4 控制文件	12
1.3.5 初始化参数文件	13
1.3.6 告警日志文件和跟踪文件	13
1.3.7 口令文件	13
1.4 Oracle数据库进程	14
1.5 Oracle数据库操作	16
1.5.1 事务	16
1.5.2 启动	17

1.5.3 日志切换	18
1.5.4 检查点	18
1.5.5 关闭	18
1.6 小结	19
习题	19
答案	20
第2章 备份与恢复选项	21
2.1 用户管理的备份与恢复	22
2.1.1 物理备份选项	22
2.1.2 物理恢复方法	24
2.1.3 复制数据库	26
2.1.4 备用数据库	27
2.1.5 逻辑操作	27
2.2 服务器管理备份与恢复	28
2.2.1 恢复管理器的特征	28
2.2.2 恢复管理器的功能	29
2.3 Oracle备份与恢复	30
2.3.1 Oracle 9i的增强特性	30
2.3.2 Oracle 企业管理器	30
2.3.3 供应商的解决方案	31
2.4 小结	32
习题	32
答案	33
第3章 专项练习	35
3.1 开放疗法	36
3.2 操作所需的机器	37
3.2.1 环境变量和Oracle软件	38
3.2.2 Linux环境下的Oracle服务器	38
3.2.3 Windows NT环境下的Oracle服务器	38
3.2.4 UNIX环境下的Oracle服务器	39
3.3 安装Oracle软件	39
3.4 练习所需的数据库	40

3.5 疑难解答	52
3.5.1 Connect命令	52
3.5.2 Select命令	52
3.5.3 Shutdown命令	53
3.6 小结	53
习题	53
答案	54

第二部分 用户管理的备份与恢复

第4章 关闭数据库的备份与恢复	55
4.1 关闭数据库的备份与完全还原	56
4.2 完全与不完全恢复	61
4.3 疑难解答	71
4.3.1 Alter Database Open	71
4.3.2 Startup Mount	72
4.3.3 Alter System Switch Log File	72
4.4 小结	72
习题	72
答案	73
第5章 从打开的数据库备份与恢复	75
5.1 打开的数据库的整体备份	77
5.2 完全和不完全恢复	85
5.3 疑难解答	91
5.3.1 归档命令	91
5.3.2 启动或关闭命令	92
5.4 小结	92
习题	93
答案	93
第6章 复制数据库	95
6.1 数据库复制	98
6.2 复制数据库场景	104
6.2.1 克隆到其他机器上	104
6.2.2 从磁带备份克隆	105
6.2.3 向其他机器拷贝数据库	105
6.2.4 复制部分数据库	106
6.3 疑难解答	106

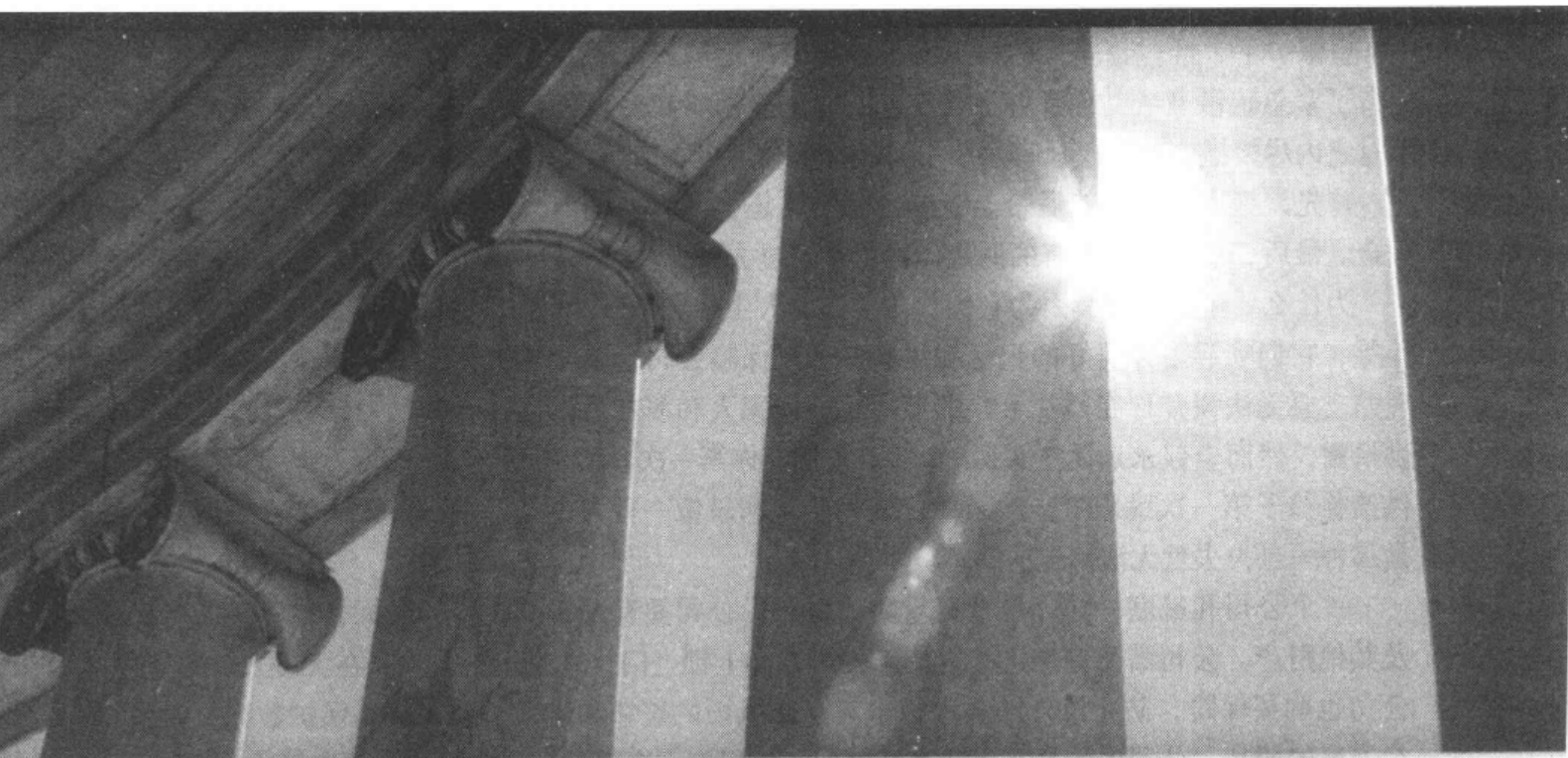
6.3.1 Connect命令	107
6.3.2 CREATE CONTROLFILE命令	107
6.3.3 RECOVER DATABASE命令	107
6.3.4 ALTER DATABASE OPEN RESETLOGS命令	107
6.4 小结	108
习题	108
答案	109
第7章 备用数据库	111
7.1 Oracle的备用数据库	113
7.2 几点建议	122
7.3 疑难解答	122
7.4 小结	122
习题	123
答案	123
第8章 导出与导入	125
8.1 数据库模式的导出和导入	129
8.2 用户模式的导出/导入	134
8.3 表模式的导出和导入	136
8.4 Oracle 9i的新特性	138
8.5 疑难解答	139
8.6 小结	139
习题	139
答案	140
第9章 表空间时间点恢复	141
9.1 表空间恢复研究	143
9.2 辅助数据库上的表空间恢复	147
9.3 可迁移的表空间	150
9.4 可迁移表空间的注意事项	153
9.5 疑难解答	154
9.6 小结	154
习题	155
答案	156
第10章 LogMiner	157
10.1 使用LogMiner分析重做	160
10.2 疑难解答	169

10.3 小结	169	13.1.1 LIST	221
习题	170	13.1.2 REPORT	221
答案	170	13.1.3 CHANGE	221
第三部分 服务器管理恢复			
第11章 RMAN配置	173	13.1.4 CROSSCHECK	222
11.1 RMAN的体系结构	174	13.1.5 VALIDATE	222
11.2 RMAN的重要特征	175	13.2 恢复管理器脚本	222
11.3 恢复管理器的目录	176	13.3 恢复管理器备份确认	226
11.4 介质管理层	182	13.4 恢复目录的清理	237
11.4.1 配置介质管理层	183	13.5 疑难解答	240
11.4.2 咨询磁带管理供应商	184	13.6 小结	240
11.5 Oracle 9i 的新特性	184	习题	240
11.6 疑难解答	185	答案	241
11.6.1 RMAN的信息	186	第14章 RMAN恢复	243
11.6.2 介质管理层	186	14.1 RMAN还原	244
11.7 小结	186	14.2 RMAN的还原与恢复	247
习题	187	14.3 恢复的技巧	261
答案	187	14.4 疑难解答	263
第12章 RMAN备份	189	14.5 小结	264
12.1 恢复管理器的备份选项	190	习题	264
12.1.1 备份	190	答案	265
12.1.2 RMAN 备份压缩	193	第15章 RMAN复制数据库	267
12.1.3 映像拷贝	197	15.1 恢复管理器复制	270
12.1.4 代理拷贝	198	15.2 在不同的服务器上复制数据库	277
12.1.5 备份类型比较	198	15.3 疑难解答	278
12.2 RMAN的部署	210	15.4 小结	279
12.3 RMAN的备份的性能	213	习题	279
12.3.1 监视备份与拷贝工作	213	答案	280
12.3.2 调整性能的技巧	214	第16章 RMAN备用数据库	281
12.4 疑难解答	214	16.1 恢复管理器的备用	284
12.5 小结	216	16.2 恢复管理器备用数据库的备份	290
习题	216	16.3 疑难解答	292
答案	217	16.4 小结	293
第13章 RMAN目录维护	219	习题	294
13.1 RMAN的命令	221	答案	294
		第17章 RMAN表空间恢复	297
		17.1 恢复管理器的TSPITR	299

17.2 疑难解答	305
17.3 小结	306
习题	306
答案	307

第四部分 附 录

附录A 术语表	309
附录B Oracle 9i中的RMAN	315



第一部分 | 备份与恢复概述

第1章 保护数据

想像一下，经过许多天的倾盆大雨，你家附近的河流开始上涨并且漫过了堤坝。当地政府发布了紧急疏散通告。你那幢有着三间卧室的平房将在今天午夜被洪水淹没，所以必须在两个小时之内尽快撤离。仅仅只有两个小时，你将会带走哪些东西？

首先，抱上孩子，然后带上宠物。也许你还会包上些首饰、换洗衣物以及放在屋里的少量现金。最后，带上家里所有的照片。两个小时过去了，该是离开的时候了。

为什么要带上照片？因为它们代表了你的人生经历——你的童年、婚姻、孩子、朋友、假期等等，它们唤起了你美好的记忆和体验。虽然保险公司会赔偿你损坏的家具和设备，但是一张理赔支票无法恢复你在婚礼上拍的照片。即使家人和朋友可以助你一臂之力，慈善机构也会提供捐赠，然而金钱永远无法买回8毫米录像带上你第一次学骑自行车时的模样。保险机构不会提供给你孩子第一次蹒跚学步的录像。这些照片和录像一旦失去了，就再也无法恢复原样，它们是那种一旦失去就无法修复的财产。

一个公司和家庭一样，也拥有重要的财产。公司拥有建筑、设备、商标名称、知识产权以及其他财产。公司需要保护这些资产。公司在晚上锁上门，并且派一名保安在大厦入口处监视。公司也购买保险，防止意外的火灾、水灾、暴风雨的破坏和盗窃。公司还要保护数据，就像一个家庭的照片，公司的数据是无法被修复的。他们怎么来保护那些数据？就是雇佣你——数据库管理员。

公司把数据保存在计算机和计算机磁带上，数据库的软件管理着这些数据。如果数据库失效，公司就可能失去它的数据。恢复这些数据几乎是不可能的。大多数情况，即使有可能恢复数据，那也是一个艰难和冗长乏味的过程。因为数据库损坏而导致的生产率降低，会使公司损失一大笔金钱。

在开始讲述本书的主要内容（Oracle数据库的备份与恢复）之前，首先要解释一些概念，并且要说明保护你的公司最珍贵电子财产——数据的重要性。你的工作就是防止数据丢失并维护系统正常运行，而公司数据只有在有效和正确的前提下才能被使用。

在这一章中，我将概述Oracle服务器，并且重点强调数据保护。对数据库体系结构和机制的清晰理解，对正在进行的数据恢复和备份工作大有帮助。你也许有过DBA的经验，或者学过某些课程，或者已经通过了Oracle认证考试，甚至你可能是对数据备份和恢复的概念感到陌生的Oracle雇员，牢固地掌握基本概念是成为一个成功的数据库管理员的关键，特别是在数据灾难性防护方面。

1.1 介绍Sid的家

我有时会以一种看似难以捉摸的方式来理解计算机技术。我喜欢通过联想来学习，通过同已经理解的事物比较来学习新的知识。我打算让你在脑海中构想一个景象，虚构一个家庭的活动。当你读到这个家庭的日日夜夜，你可能无法理解他们正在做什么，以及为什么这么做。然后，当你读到Oracle数据库是如何工作的时候，我希望你会发现一些相似之处。

让我们介绍一个有趣的例子来开始本章吧。人们从来不认为数据库管理员有趣，他们往往被认为是一个疯狂工作、言语简短、令人讨厌的群体。你可能不会从下面的讨论中得到乐趣，但是我会（如果下面的内容与你的亲戚，或者你曾经见过的家庭之间如有雷同之处，纯属巧合）。

一个名叫Sid的男人，狂热地爱好拍摄、保存和整理照片。Sid的妻子名叫Debbie，他们有三个儿子Logan、Archie和Chuck。他有一所很大的房子，同住的有男管家Simon和女仆Pam。我将介绍他的家庭、他的房子和他的爱好：拍摄、收集和炫耀他的照片。

现在，Sid快乐地生活着——家庭、朋友和假期。他不时地拍照。事实上，他总是随身带着相机。他不想错过任何东西。每一个早餐、午餐和晚宴，都会被拍下来。当孩子们从学校回家，镜头摄入他们对父亲的问候。当孩子们做家庭作业的时候，“咔嚓、咔嚓、咔嚓”，照相机的快门响个不停。棒球赛上，儿子Archie的每一个投掷都被拍摄下来。学校的舞会上，Chuck的父亲拍下了每一个缓慢的舞步以及在苏打水机器旁的交谈。而他的狗——Sadie则是这个世界上被拍摄次数最多的狗。

他和他的家庭收藏了数以百万计的照片。一些照片被销毁掉了，一些则经历了改变。Sid有一套神奇的系统来处理这些照片。让我来告诉你一些关于他的家庭、他的房子的事，以及关于他和他的家人如何致力于发展、组织和保护所有这些照片。

Sid用一架特殊的相机拍摄。他的相机每拍摄一次，就同时产生一张照片和一张底片。他从来不用换胶卷（这是一种科幻技术）。照片被贴在房间的墙壁上，而底片则被保存在一个安全的地方。

1.1.1 Sid的家

我介绍过Sid的家了吗？嗯。那里非常的大而且到处都是照片。因为照片实在是太多了，所以Sid必须寻找地方来放置它们。一些照片被挂在墙上，一些被藏在照相簿内，一些被插入剪贴簿里，一些被放在储藏室的一个盒子里再也不会被撇上一眼。所有这些照片都被放在他家中的某处。他称呼这些放置照片的地方为“相片空间”。一个相片空间可能是一个有着很多墙壁的房间，或者是有着很多页的剪贴簿。一个房间是照片的逻辑存储单元。房间内的墙壁则是相片空间中保存照片的物理结构。

1.1.2 Debbie的家务事

因为Sid忙于拍摄和处理照片，他没有足够的精力来放置和储存它们——那是她妻子的工作。Debbie的全名是Debra Wridner。她的T恤衫上印着DBWR——这是一种家庭传统——为所有的东西做标签。

把所有这些照片放置得井井有条是一件巨大的工作，不过Sid和Debbie已经建立了一个系统。当Sid拍摄结束后，他把照片都放在厨房里一个特定的架子上。那个特定的架子被称为“DB_BUFFER_CACHE”。Debbie时常检查那个架子，看看上面是否有新拍摄的照片。在白天，她把从架子上取回的照片分别放置到对应的相片空间。事实上，在一场棒球赛或者学校的舞会上，Sid制作照片的速度太快了。以至于Debbie几乎跟不上。这是一个特别的情景，

Sid拍摄照片，然后放置在那个特定的架子上，Debbie把这些照片放在剪贴簿里，挂在墙上或者插入照相簿内。

当Sid带着照片和底片进入厨房时，他和他妻子之间可能会有这样的一段对话：

Sid：“亲爱的，猜猜我给你带回来了什么？”

Debbie：“你又拍了很多照片，是吗？你知道我是多么喜欢整理照片啊。”

Sid：“当然了，Archie的棒球队赢得了最后一局。他在第九局以两个二垒打获胜，我全都拍摄下来了。”

Debbie：“太好了，那些照片放在相片空间9号墙壁的运动类里，看上去一定棒极了。我已经等不及去放置它们了，我将邀请所有的邻居来观赏。”

Sid：“Archie也非常高兴。OK，我再去拍一些照片，我将在23毫秒后回来。哦，顺便说一下，我刚才将那些照片放在DB_BUFFER_CACHE架子上。现在Sid的目录号为1 332 935。”

Debbie：“知道了，再见。”

为了易于管理照片、框架和日志，他使用一种特殊的数字来建立目录，称之为Sid的目录编号。他用这些独特的、不断递增的数字来追踪对这所房子内所有照片的操作。Sid和Debbie已经对查阅Sid的目录编号感到厌倦。Sid在所有的照片、录像带和底片上都写上“Sid的目录编号”，累得手要拍筋。最后，他们将“Sid的目录编号”缩略为SCN。记住SCN，Sid的生活都围绕着它展开。

现在，全家人都知道这个数字被简称为SCN。你可以在每一个地方都看到SCN。Sid把它们写在底片上，然后他把低位和高位的SCN写在每一包底片外面。Debbie把它们写在地放置照片的地方。这些数字在Sid的复杂家庭生活中显得尤为重要。

Debbie把照片放在框架里，房子里的每一个框架都是一样大小。一些框架只保存了一张照片，一些框架则保存了几张照片。一些特别大的照片则要占据多个框架。Sid有一张和原物同样大小的照片，拍摄的是他在跳伞后在空中做造型动作，这张照片占据了两个框架。当照片跨越框架时，Sid称之为照片链。

任何时候，只要Debbie改变框架内的照片， she就把照片的SCN写在框架上。房子内的每张照片都有一个惟一的数字。他们称照片的这个数字为照片ID。这些照片ID帮助Sid管理家中所有墙上的照片。如果他需要尽快找到一张照片，他通过照片ID来实现。通过照片ID，Sid能够迅速找到墙壁和保存照片的框架。

有时候，Sid会对照片进行一些改变。如果他要编辑一张照片，那么他让Debbie通过照片ID找到他的照片。Debbie给他一份包括所需照片的整个框架的拷贝。为防止Sid可能改变主意，不喜欢他已经做的一些改变，他把那份拷贝放在一个储藏室内，称之为“回退”储藏室。当Sid完成了更改并且确定了以后，Debbie把更改后的照片放回到墙壁上原先的地方，并且把新的SCN写在框架上。如果Sid不喜欢他的更改，他们就从“回退”储藏室里取出框架并且把它放回墙上。这样一来，照片就和Sid没有改动以前一样。

1.1.3 Logan的家务事

Debbie为了处理Sid的照片已经忙得焦头烂额，所以她没有足够的时间和精力来处理Sid

的底片。Sid认为他的儿子们在玩电子游戏和吃土豆片上浪费了太多的时间，他决定给他们安排任务。

Sid的大儿子叫做Logan William Randolph。Logan的衬衫上都印着LGWR。Logan有一个非常简单但是非常重要的任务。Sid的相机产生照片的同时产生了底片，Logan的任务是将底片放入笔记本里。家中的厨房里竖着另一个架子，架子上Sid贴了一个标签，用来标记那些发送给大儿子的底片。标签被称为“LOG_BUFFER”，Logan坐在厨房的桌子旁，观察这个特殊的LOG_BUFFER架子。当有底片被放到这个架子上时，他立即把底片放在笔记本里。Sid称这些笔记本为“重做笔记本”。他使用这个名称是因为以后他可以用里面的底片重新制作一张照片。Logan检查他的架子的频率要比Debbie的高得多。Sid表示，只要有底片，他就可以随时重新制作一份照片。Logan和他父亲的对话简短而温馨：

Sid：“Logan，底片在架子上了！”

Logan：“知道了，我会立即处理的。”

Logan按照他得到底片的顺序将这些底片放入“重做笔记本”。一旦当前的笔记本装满了底片，他就换另一个新的笔记本。每一次他更换“重做笔记本”后，都在“告警笔记本”上做一个记录，然后通知他的兄弟来进行他们的家务活。

悲哀，是不是？老爸和他的大儿子太专注于底片了！

顺便提到，不是所有的照片都有对应的底片保存。Sid指定某些照片不必被做日志，这些照片就没有对应的底片保存在“重做笔记本”里。Sid称其为“无日志”。

1.1.4 Archie的家务事

Sid的第二个儿子Archibald则另有任务。他的衬衫上写着ARCH。Archie从重做笔记本里提取底片后，把它们放置在一个安全的地方。

当Archie发现Logan更换到另一个重做笔记本的时候，他就开始行动。他拿起重做笔记本中的那些底片，把它们放在房子外面的一个安全地方。Sid在后院建造了一个控制好温湿的地下室。那是他保存底片的安全地点。如果暴风来到，破坏了房子，他可以使用保存在地下室的底片来替换那些被损坏的照片。每一份Logan的“重做笔记本”拷贝，被称为“归档的重做笔记本”。

可能某一天，你可以听到Sid和他次子之间的笑谈：

Sid：“Archie？你在哪里？”

Archie：“老爸，我在自己房间里，Logan现在切换笔记本了吗？”

Sid：“是的，他刚刚干完。”

Archie：“知道了，老爸。我会复制这些底片并且制作一个新的笔记本放在地下室里。然后，我会更新‘控制笔记本’，并且在‘告警笔记本’里写一个记录。”

Sid：“你是一个好孩子，我知道可以相信你。”

1.1.5 Chuck的家务事

Sid最小的儿子Chuck确保每一件事情都是同步的。他的全名是Charles Kenneth Patrick

Thomas。他衬衫上的字母是CKPT, Chuck的任务是将每一面墙壁、相片册和剪贴簿上的相片数字和控制笔记本中的数字协调起来。当他得到信号, 他就在房子里奔跑起来, 在每一面墙壁上写下数字, 同时将同样的数字也记录在控制笔记本里。这使得每一件事情都协调起来。以下是一段Sid和Chuck的对话:

Sid: “hi, Chuck。”

Chuck: “老爸, Logan又开始更换笔记本了吗?”

Sid: “是的, 儿子, 你知道规矩。带着目录编号1 332 935跑遍房间, 把这个数字写到每一面墙壁的开头, 每一个剪贴簿的封面和照片盒子的顶部。”

Chuck: “没问题, 给我几微秒时间搞定”。

Sid: “别忘记把1 332 935写在控制笔记本上!”

Chuck: “老爸, 我知道了, 同样的事我一天内已经做过几百次了。”

我敢打赌, 你已经对阅读这个疯狂的Sid之家感到厌倦, 是不是? 到现在为止, 你可能已经看出Sid有一个功能失常的家庭。功能失常? 是的! 不过却有效而可靠。Sid喜欢这样。

1.1.6 控制笔记本

这个家庭怎样维持所有这些活动? Sid、Debbie、Logan、Archie和Chuck断定, 要想把这项照片收集和整理工作处理的井井有条, 必须要制作很多详细的笔记。他们每个人都把自己的东西写入一个日志, 称之为“控制笔记本”。

控制笔记本有一个记录整个房子中每一堵墙壁、剪贴簿和照片盒子的家庭财产清单条目。如果Sid增加了一个房间, 来放置更多的照片, 他在控制笔记本里做一个记录。当Logan更换重做笔记本时, 他在控制笔记本里做摘要记录。每次Archie制作一个归档的重做笔记本放入地下室时, 他把名字和SCN的范围写在控制笔记本里。每次Chuck在房子里跑完一圈, 他把细节也记录在控制笔记本里。这特殊的笔记本动态保存了所有地方的资产情况, 和房子里以数字表示的进程。

1.1.7 告警日志

时不时的总会有些意外发生, 也可能出现错误。Debbie可能在把照片放进框架的时候发生问题, 因为那面墙上已经没有多余的空间了。Archie可能无法折叠起底片条, 因为有人把苏打水喷在上面。Sid考虑到了所有的可能, 他买了一个螺旋轴的笔记本, 放在厨房的抽屉里。在笔记本的面子上写上“告警日志”。每一个家庭成员将遇到的问题写在这个日志里。他们不仅要记下发生的错误, 还要记下正常的活动过程。这个告警日志是一个用来保存在Sid家里已经和即将发生的事情的便利场所。如果他们要解释某个情形下额外的细节, 他们制作一个新的“追踪笔记本”, 把信息记录在里面。这样一来, 这个告警笔记本不会变得比它原先的样子更杂乱。

1.1.8 清晨程序

全家早晨一起醒来, 也总是同时上床睡觉。他们称这些时间为一天的“启动”和“关闭”。

下面是早晨的启动程序：

Sid从床上起来，端上一杯咖啡开始他的一天。在咖啡壶边上，他将一系列指令保存在笔记本里，称之为初始化参数文件。阅读这列指令帮助他开始新的一天。啜着咖啡，他走过去确认控制笔记本是否放在老地方。他在房子里打转叫醒Debbie、Chuck、Logan、Archie、Simon和Pamela（你很快就会遇到他们）。

当每一个人都醒来后，就打开控制笔记本。确认根据控制笔记本的内容，所有的照片墙壁都没有问题，当所有的东西都被检查过以后，他打开房子迎接宾客，并开始拍摄照片。

Sid雇佣了一个男管家，Simon，在早晨帮助他，Simon的燕尾服上写着字母SMON（省掉名字里的一个字母）。有些早晨房子里特别凌乱，因为在前一个就寝时间，家庭的每一个成员都没有及时清理好各自的事物。当Simon就位后，Sid打开窗帘，挂好电话，打开门锁，他的家庭开放，准备好了迎接客人。Sid和Simon可能这样开始一天：

Simon：“早上好，先生。我今天能帮你做什么吗？”

Sid：“是的Simon，早上好。看来我们这里很凌乱啊，我们昨天晚上突然中断了。你能否处理一下所有这些没有完成的照片和‘回退’。”

Simon：“当然了，所有这些底片在重做笔记本里的都要被重新制作，然后以简单的顺序放置在正确的框架里。”

Sid：“你处理完这些后，接着处理‘回退’，是吗？”

Simon：“当然了，先生。我从回退储藏室里取出照片框架，然后把旧的版本放回它们原来的地方。”

Sid：“太棒了，你完成后，我就打开房子。”

1.1.9 夜间程序

Sid家的夜间程序也很有趣。当Sid躺下准备休息的时候，他要求家里所有的东西都放好、关掉和停下。就寝时间的第一个行动是要求房子里所有的客人回家去。Sid晚上的情绪有四种情况。第一种，有时他很有耐性，等待所有的男女客人自行离去后，再就寝；第二种，Sid没有耐性，他亲自来监督每一个客人，当客人干完各自的事情（看照片等等）后，他给客人指示出门的方向；第三种，Sid非常没有耐性，他揪着客人陪他们出门，不让他们干完正在干的事；第四种，Sid拉下房子的电闸，大叫“时间到”，房子里一片狼藉。Simon不得不在第二天早晨来清理混乱的场面。

当一个客人走后留下一团糟时，Sid叫来他的女佣Pamela Monica。她监控客人的活动，并且在客人走后处理善后。她的清洁设备上贴着标签PMON。

如果Sid没有突然拉闸，每个人都打开灯，进行清理。同步处理笔记本来完成各自的任务。Debbie从DB_BUFFER_CACHE架子上取走所有的照片，放到合适的墙壁上。Logan确保LOG_BUFFER架子是空的，所有的底片都在当前的重做笔记本里。Chuck在房子里兜一圈，用最高和最低的目录编号（SCN）来更新墙壁和笔记本。Archie把最后的那张底片放入归档的重做笔记本，放到后院的地下室。然后更新控制和告警笔记本。当每个人都干完后，Sid

关灯，大伙儿睡觉。

1.1.10 Sid的问题

有时候，Sid家会出现问题。当以下这些事件中任何一件发生时，房子里的活动就会停止。

- 当Sid醒来，而他无法找到初始化文件，则无法开始一天的工作。
- 有时，Sid需要用Archie保存在后院地下室的底片来恢复丢失的照片，如果他丢失了一张底片，或者在恢复过程中底片遭到破坏。他无法恢复任何在那张无效底片以后拍摄的所有照片。
- Sid一次丢失了几张底片。他不小心打开相机，曝光了胶卷。这使得他情绪极坏，他的世界好像走到了尽头。他联系了照相机厂家，他们提供了一个特殊的照相机，带有两个胶卷舱。如此一来，万一胶卷破坏了，他还可以指望另一个舱内的胶卷。Sid称之为镜像。
- 如果后院的地下室满了，Archie没有空间来放置新的底片。Archie只是在地下室里等着新的空间。这使得Sid和Debbie也跟着等待，整个Sid的家庭活动陷于停顿状态。
- 如果Debbie得了流感，卧病在床。你猜猜看会怎么样：Sid疯狂地关上了门。

尽管有这些问题和其他一些家庭挫折及功能失常，但Sid的家庭仍然可以处理百万计的照片和数以千计的同时到访的客人。

和我一样，你可能会想知道，Sid怎么会有那么多的时间和精力来进行照片收集活动。他没有工作吗？事实上，没有。钱不是Sid的目标。他不必工作。他在几年以前完成了一个聪明的投资。他在1984年以后买入了大量的电脑公司的股票，那时它们刚开始向公众发行。他所投资的公司总部在加利福尼亚州的红树海岸。

在介绍了Sid的家庭后，你会发现他为家庭的所有活动设计了结构。就像这个家庭的活动一样，Oracle数据库也有空间、过程和程序。在脑中与这个家庭类比，我现在简短地描述Oracle数据库体系结构、过程和操作中属于Oracle数据备份和恢复的部分。可能你会发现和Sid家的相似之处。

1.2 Oracle服务器

Oracle服务器包含Oracle数据库和Oracle实例。Oracle数据库是存储在文件中作为一个单元来处理的数据的集合。Oracle实例是后台进程和内存结构的结合。Oracle实例的系统标识符是SID。

1.2.1 数据库的内容

在考虑保护数据库数据的日常任务时，你希望知道要保护的是什么。虽然一个Oracle服务器包含了复杂的软件，但在这里我将用简单的术语来描述，就像我们开始时那样。保护数据库，着重于保护文件中包含的信息。数据库中的数据以不同的形式存在：表、索引、视图、PL/SQL代码、Java代码等等。备份一个数据库就是保护这些数据库对象，并且允许你在必要的时候替换它们。

1.2.2 内存的体系结构

加入到Oracle管理员队伍的DBA首要学习任务之一就是了解SGA (System Global Area) 的组成。在实例启动时分配的这一内存区域分成三个主要部分:

- 1) 共享池 (shared pool) 也分为库缓存 (包含共享SQL区域)、字典和行缓存 (包含了数据字典内的数据)。另外还有两个池, 分别是Large pool和Java pool。Large pool用来处理RMAN操作、并行执行 (并行查询)、多线程服务器的分时记忆。Java pool则被Java代码和数据所使用。
- 2) 缓冲存储区 (buffer cache) 保存刚使用过的、从磁盘上的数据文件内读取的数据块。当一个数据块被更新时 (例如, 在表内插入新行), 这个块必须被放置在缓存里, 然后才能被更新。过了一段时间, 数据块将会被写回磁盘, 反映这个新的更新。
- 3) 重做日志缓冲区 (redo log buffer) 当数据块被更新后, 对它们所作的修改被保存在重做日志缓冲区里。这些改变都被记录下来, 当出现故障的时候可以使用这些记录。在对实际的数据块进行操作之前, 就进行了这些重做记录。当这个日志缓冲区已经有1/3满了或者已经有了1Mb的重做记录时, 这个日志缓冲区的内容就执行一个处理动作, 被拷贝到联机重做日志里。

这些内容在数据库操作中具有同等重要的地位, 不过其中两个和数据备份和恢复直接相关的是缓存和日志缓冲区。当我们看到本章的其余部分后, 就会一目了然了。

1.2.3 数据库的体系结构

Oracle数据库在哪里保存表、索引和代码? 这些对象被存放在数据库的体系结构里。这里列出的是关于备份和恢复的关键逻辑结构的简单描述:

- 表空间 (tablespace) ——一个表空间是相关对象的逻辑存储区域。每一个数据库对象都保存在表空间里。可以定义一个表空间, 在那里建立表和索引。也可以在建立表和索引的时候, 改变表空间的位置。用户可以使用联机的表空间, 但是无法使用离线的表空间。你可以把一个表空间设定为只读, 以此来减少连续不断的对巨大的静态数据库部分的备份操作。一个临时的表空间有效地为排序操作安排空间。一个可迁移的表空间可以在数据库之间迁移。
- 段 (segment) ——一个段可以是数据、索引、临时数据或者回退段。一个Oracle数据段保存所有的数据, 表、一串表和表的一部分。类似的, 索引数据被放在索引段内。在某些查询过程中, 临时段提供了一个临时工作空间, 为SQL语句的解析和执行提供了迅捷的平台。一个回退段 (有时也指重做段) 存放旧的数据, 它们被每一次的数据处理所改变 (不管这个处理是否被实际执行)。回退段用来提供读取的一致性, 回退事务, 并从没有执行的事务来恢复数据库。一个段只能存放在一个表空间里, 但是一个表空间可以存放许多不同的段。
- 域 (extent) ——域是由一定数量的连续的数据块组成的, 是数据库存储空间分配的逻辑单元。一旦一个域装满了数据, Oracle则为段安排新的域。一个段包含了最初的域, 并且根据要求安排额外的域。

- **数据块 (data block)** —— 数据块是数据库里底层的存储容器，也是Oracle最小的输入输出单元 (I/O)。典型的数据库块的大小为2KB、4KB、8KB或者16KB。当创建数据库的时候，就为整个数据库的所有表空间定义数据块大小。在Oracle 9i中，一个数据库可以包含不同的数据块尺寸，这使得在数据库之间传输数据变得很容易。
- **行 (row)** —— 行存储了表和索引数据。一个行被包括在一个或者更多个数据块里。
- **行标识 (ROWID)** —— Oracle使用行标识来定位数据库内的每一个行。这个行标识标志出一个数据文件内的一个数据块里的一行。
- **系统更改号 (SCN, System Change Number)** —— 每一个数据库事务执行时都被分配了一个单独的系统更改号 (SCN)。这个不断增加的数字在整个数据库里用来控制并发和一致性，进行重做和恢复。SCN代表一个协调执行着的数据库版本，也可以把数据库想象成一个时钟。
- **更改矢量 (change vector)** —— 当要改变数据库的数据时，这些改变的细节被记录为更改矢量。通过这些记录，数据库处理过程可以被重现。简而言之，一个更改矢量就是记录了一个数据块从一个状态改变到另一个状态的过程。

注意 每一个Oracle数据库有一个系统表空间。这个系统表空间包含了整个数据库的数据字典表。所有代表了储存的PL/SQ程序单元（过程、函数、包和触发器）的信息都在系统表空间里。

Sid家的Oracle内容

就像Sid家保存照片一样，Oracle数据库保存数据。以下是一些对比：

- **Oracle实例** Sid进行他的家庭照片收集操作。Oracle实例包含存储结构和后台进程。
- **Oracle数据库** Sid的家保存照片和相关的笔记本、墙壁和储藏室。一个Oracle数据库在文件内存储数据。
- **行** Sid通过查看、修改和删除来处理照片。一个数据库的行保存了数据，可以被查看、修改和删除或者添加。
- **行标识** Sid家的每一张照片都有单独的标识，这个标识描述了照片的位置、框架和放照片的墙壁。每一个数据库行有一个单独的行标识，行标识定义了对对象的标识符、数据文件、数据块和行。
- **块** 在Sid家墙壁上的框架中保存照片。数据块在数据文件里保存行。
- **表空间** Sid在称为照片空间的房间里保存照片。数据则被收集后逻辑存放在表空间里。
- **系统更改号** 当Sid把照片放在框架里，并将一个永远增加的目录编号写在框架上。Logan和Archie把目录编号写在每一个底片的笔记本上。Chuck使用当前的目录编号来更新墙壁。系统更改号能够标识数据库内每一个发生的改变。
- **回退** 如果Sid改变了照片，他拷贝框架的内容，并且将其放在回退储藏室里。当一个数据块被改变了，重做信息被记录在回退段里。
- **更改矢量** Sid为他的照片制作了底片，所以照片可以被重新制作。数据库产生了更改矢量存放在重做文件里以备数据库的恢复。

这个简单而概括的描述作为对数据库内容的最基本的介绍。处理数据库对象的尺寸和参数与数据库的改变有关。建立数据库代码则关系到应用软件的发展。保护数据库数据、结构和代码则是本书的目的。现在我将重点介绍组成Oracle数据库的真正重要的文件。

1.3 Oracle数据库文件

当你在计算机上工作的时候，软件不断地将计算机信息储存到磁盘上。当在字处理软件中编辑了一封信或者一个文档时，点击SAVE按钮，文档的内容就被保存到硬盘上。一个Oracle数据库由计算机磁盘上的几个文件组成。这些文件包含了需要准确操作的数据库信息和结构。让我们看看每一个文件和它的功能描述，以及它们之间的关系。

注意 简而言之，Oracle数据库就是文件的一个集合。其中三个关键的文件类型是数据文件、重做日志文件和控制文件。

1.3.1 数据文件

数据库里的所有数据都保存在数据文件里。所有的表、索引、触发器、序列、PL/SQL代码、视图，都在数据文件里。即使这些和其他的数据库对象逻辑存储在表空间里，它们事实上是保存在计算机硬盘的文件里。

每一个数据文件有特定的Oracle软件的内部格式。应该特别注意的是数据文件有一个头和一系列数据块。Oracle数据文件的头包含一组结构，包括数据库ID（当数据库创建时给予的编码）、文件的编码和它的名字、文件类型、创建的SCN和文件状态。Oracle使用文件头来确定这个文件是否是所期望的（例如，它与数据库其他文件同步更新）。一个数据文件只能属于一个表空间，但是一个表空间可以由几个不同的数据文件组成。

1.3.2 联机重做日志文件

在对数据库进行操作的时候，一旦重做记录从重做日志缓冲内取出，它们中的大部分被记录到联机重做日志文件里。通过观察这个文件类型的每一个单词，你可以看出它的用途。

- **联机（online）** 数据库可以将这个文件作为存储数据事务的场所。它们之所以称为联机，是因为它们必须联机并且可以被数据库使用。
- **重做（redo）** 表示某些东西可以被重做。重做一个任务意味着重新执行一遍这个任务。就计算机而言，重做一个任务等于同刚才完全一样执行一遍刚才的任务。因此，一个联机重做日志文件就是保存数据库改动的相关信息，以便于以后可以重做。
- **日志（log）** 大多数的数据库改变都被记入日志。我不是在讨论一棵倒下的树上被锯下来的一段（些处语涉双关“log”另一个英文意思是圆木。——译者注）。“log”这个词原本的意思是船舶的船长和飞机的机长所保存的活动信息。
- **文件（file）** 就像任何其他的操作系统文件一样，当在文件所在的目录下进行列表操作时就可以看见。

这个联机重做日志文件（online redo log files）的名字取得很好，很有表达力。它也比较长，读写都需要花一些时间。后面将遇到所谓的重做日志、重做文件和日志文件，指的都是它。

每一个数据库必须有至少两个联机重做日志文件。为什么？因为当数据库操作的时候，每个数据库的改变都被记录到一个重做日志文件里。这个日志文件就是当前的重做日志文件。一旦哪个文件里重做记录满了，Oracle数据库需要在其他地方写入更多的重做信息。为了防止在当前的日志文件里写入过多内容，Oracle软件就把信息写到下一个重做日志文件里。这个更改被称为日志切换。

重做文件对完成数据库的恢复极其重要，为此Oracle提供了一个方法来镜像这些文件。联机重做数据文件以组的形式存在。Oracle服务器不是把重做信息写入每个组里的某个重做文件，而是同时写入两个（或更多）的联机重做日志文件里。每个文件都是组里的一员。

重做文件包含了更改矢量（它们一起被收集到重做记录中）以及其他恢复时需要的重要信息。这些重做记录是记录在重做日志缓冲里的拷贝。每个重做日志文件头包括如下信息：数据库ID、文件号、文件类型、日志内包括的最低的SCN号和下一个重做日志包括的最低的SCN号。

注意 可以将一个表、分区、索引或表空间设置为无日志属性。对一个数据库结构和对象设置这类属性，就是指定任何数据修改都不会产生重做信息。普通的SQL会对无日志的表制作重做信息，特定的选项将使SQL不制作重做信息。对象的无日志和特殊选项的组合，将防止对SQL语句进行重做记录。防止重做记录产生，就可以得到更好的系统运行性能，而不是通过重做记录来恢复数据。

1.3.3 归档重做日志文件

一旦一个联机重做日志文件被填满，Oracle服务器软件则开始向已有日志文件组里的另一个文件中写入。当这种日志切换发生时，Oracle可以把最近被写满的重做日志文件的内容拷贝到另一个地方。一旦这个重做日志被拷贝了，这个被拷贝的文件叫做归档的重做日志文件。归档重做日志文件包含数据库的历史操作。如果打算保留数据库的历史改动信息，则必须拷贝（归档）联机重做日志文件。为什么？因为联机重做日志文件会重复地使用，在联机重做文件里的信息会被覆盖。因此，归档重做日志文件保存了更改矢量信息的拷贝，可以用来重新执行数据库的操作（如此就是归档重做日志文件）。它们也被称为归档文件、归档日志文件和归档日志。

这些归档重做日志文件是成功恢复的关键。如果部分数据库丢失或者被破坏了，通常需要几个归档日志来修复数据库。归档日志文件必须按照顺序再应用到数据库上。如果其中的一个归档日志文件丢失，其他的归档日志文件就无法使用。因此，应保存好自从上次备份以来制作的所有归档日志文件，并且保护好它们。

1.3.4 控制文件

Oracle数据库是相关文件的物理集合，采用了一些方法来同步和控制它们。因此，Oracle数

数据库有一个控制文件。控制文件包含组成这个数据库的所有文件的详细记录。

控制文件是硬盘上数据库服务器的一个文件，这个文件包含了关键的有关数据库和数据库文件的数据库信息。例如，控制文件保存了数据库的名字、所有数据文件的名字、所有数据文件编号、数据库的块尺寸、联机重做文件的位置、联机重做文件组的定义、当前数据库的系统更改号、归档日志文件的位置、归档日志文件信息和备份的历史。

控制文件对操作数据库非常重要，Oracle软件允许拥有多个这类文件。你可以有几个控制文件的拷贝，即使硬盘上丢失了一个控制文件，也可以保护数据库。

一个Oracle数据库操作中需要的关键文件是数据文件、联机重做日志文件和控制文件。归档日志文件并不是必须的，但是一旦数据库出问题，要进行完全恢复时必须应用归档日志文件。

1.3.5 初始化参数文件

Oracle数据库需要的另一个文件是数据库参数文件。这个独立的文件只在数据库实例启动时使用，它必须存在而且必须能被Oracle软件读取。参数文件里包含了非常重要的初始化数据库信息，包括数据库控制文件的位置、数据库名称、归档日志文件信息以及其他内存和函数的参数。一旦数据库启动，直到下一次的数据库启动才会再次使用这个文件。这个数据库参数文件可以被称为PFILE、初始化文件或者init.ora。

在Oracle 9i中，服务器参数文件（SPFile）是一个文本参数文件的二进制镜像。Oracle 9i软件在一个服务器参数文件里保存系统动态参数。这意味着，在数据库运行时，每次改动系统参数时，不需要改变init.ora文件。

1.3.6 告警日志文件和跟踪文件

数据库操作时，事件和错误信息将被记录到数据库服务器的文本文件中。告警日志文件对于数据库管理员跟踪重要的数据库活动是有帮助的，例如数据库的启动和关闭、数据库启动时使用的参数设置、联机重做日志切换等。同时，为了便于检查，也将许多数据库错误写入该文件。任何数据库结构的改变（增加日志文件、增加表空间、减少数据文件等）也被记入告警日志文件。

数据库错误发生时，将生成跟踪文件。这个跟踪文件包含了错误事件的详细信息。当向Oracle的全球技术支持（Oracle Worldwide Support）请求帮助时，他们要求提供这些文件的拷贝。

1.3.7 口令文件

口令文件作为提供数据库操作安全的一种选择。不必担心任何人可以从网络客户机上关闭或启动数据库，除非授权给他们。这个加密了的口令文件为合法用户（包括自己），往往需要在数据库服务器之外的其他机器上执行特定的数据库操作。

这些是组成Oracle数据库和Oracle数据库环境的重要文件。下一节，我们学习在这些文件上执行操作的数据库进程。我们还将观察数据库结构活动相关的关键内存位置。

Sid家中的Oracle文件

Oracle数据库中的文件同Sid家中使用来保存相片的笔记本和墙壁相比较，它们被用来存储照片、标记改变、存储进程和清单。

- **数据文件** Sid的妻子把照片放到家里墙上的框架里。每个墙壁就像一个数据文件。数据文件把数据存储块中。
- **联机重做日志文件** Sid的照相机产生底片，Sid把这些底片放在一个架子上。Logan把这些底片插入笔记本以保护它们以备将来使用。联机重做日志文件把数据库的执行动作保存为更改矢量。日志写入进程将LOG_BUFFER中的内容刷新到当前的联机重做文件里。
- **控制文件** Sid和他的家人使用控制笔记本来管理所有的位置清单：所有墙壁上照片的位置、联机底片笔记本、放在地下室的归档底片。这个笔记本同时追踪每一个这些位置的目录号。一个数据库控制文件追踪所有数据文件的位置、联机重做日志和归档重做日志。使用数据库的系统更改号，控制文件可以确保所有的数据库文件保持同步。
- **归档重做日志文件** 当Logan装满一个底片笔记本以后，他把底片放到一个新的笔记本里。当发生笔记本切换时，Archie制作整个笔记本的拷贝，并且把它放到后院的地下室里。一个归档重做日志文件由ARCH进程产生，它拷贝了一个联机重做日志文件的内容。
- **参数文件** 开始新的一天时，Sid阅读他的参数文件内容。这个列表提供他和家庭一天的活动指南。数据库参数文件在数据库启动时被读取。
- **告警日志文件** 当Sid和他的家人在处理家务事遇到问题时，他们在告警笔记本里制作一个记录。Oracle数据库在告警日志文件记录进程记录和一些错误信息。
- **跟踪文件** 当Sid或他的家人需要详细记录某项情况的时候，他们建立一个新的跟踪笔记本并且记录详细情况。Oracle进程把详细资料写入硬盘上的跟踪文件里。
- **口令文件** Sid的家里没有口令文件。一个Oracle数据库拥有口令文件以便子设备的远程管理。

这些重要的文件组成了Oracle数据库和Oracle数据库的运行环境。接着，让我们看看数据库的进程是如何对这些文件进行操作的。我们也将观察数据库结构活动所涉及的关键内存位置。

1.4 Oracle数据库进程

Oracle数据库是一组数据库文件。Oracle进程和内存中包含数据库的实例。这些进程不停的运行并执行数据库任务。内存区域保存数据库进程使用的数据和信息。

进程定义

虽然许多进程组成了一个Oracle数据库，但在本书中我将着重介绍备份和恢复活动中的进程。进程能执行很多功能，我将着重于属于数据库备份和恢复的功能。

- **数据库写入器 (database writer)** 数据库写入器将数据从内存中写入到数据文件中。因

为数据库中基本的存储单元是数据块，数据库写入器写入数据块。当事务被提交时，数据库写入器并不立即将事务写入数据块。数据块可以作为变更了的数据块仍旧保留在内存中，有时称为脏数据块。当需要内存或在一个检查点时，数据库写入器将实际地将其写入到磁盘上的数据文件中，以清空“脏”数据块。在数据库中可以有多个数据库写入器。这一后台进程的简称是DBWn，n是数据库写入器进程的编号。数据库写入器在检查点也执行写入功能。

- **日志写入器 (log writer)** 日志写入器将信息从数据库日志缓冲中写入到当前联机重做日志文件。这一被写入的重做信息被称为重做实体、重做记录或更改矢量。重做实体连续地被写入并且包含了恢复中重新应用数据块所需的信息。如果将重做日志文件复用为组，日志写入器同时填写组中每一个重做日志文件。日志写入器的简称为LGWR。
- **归档器 (archiver)** 归档器由于安全原因向其他目的地拷贝重做日志文件的内容。当前联机重做日志文件填满后，Oracle软件将它的日志记录到下一个联机重做日志文件中。一旦发生这种切换，归档器进程拷贝这个文件到目标目录或其他数据库连接。因为一个重做日志切换可以被人工初始化，一旦切换发生则归档器将拷贝当前重做日志文件。每一个重做日志文件都要被拷贝。包含在这些归档重做日志文件中的重做记录包含了每一个数据库变化。它们对于完全数据库恢复是至关重要的。如果重做日志切换比当前归档进程拷贝速度快，日志写入器进程可以产生另一个归档进程。归档器简称为ARCH，H是指进程编号。
- **检查点 (checkpoint)** 检查点进程提醒数据库写入器覆盖内存中所有变更了的数据块。检查点也使用最新的检查点信息更新所有的数据文件头文件和控制文件。检查点同步所有的数据库文件。当重做日志切换以指定时间间隔发生时，或者当数据库脱机工作时，或者当表空间处于热备份模式，或者数据库关闭时，检查点进程执行。检查点简称为CKPT。
- **系统监视器 (system monitor)** Oracle数据库中，系统监视器提供了若干功能。用作恢复目的时，如果数据库正常关闭但清理失败，系统监视器则执行实例恢复。数据库正常关闭时，Oracle软件能同步文件、更新信息、清理用户事务。如果清理的关闭没有发生，系统监视器在数据库下次启动时则执行数据库灾难恢复。一个数据库实例仅有一个系统监视器进程，简称为SMON。
- **进程监视器 (process monitor)** 进程监视器清理失效的用户进程。当一个用户连接中断时，进程监视器回退未提交的事务并释放资源和用户会话中使用的锁。进程监视器的简称是PMON。在一个数据库实例中只有一个PMON进程。

这些进程完成的工作远不止上述这些，以上只是和数据库备份和恢复相关的重要特性的概述。

Sid家中的Oracle进程

Oracle数据库中的进程同Sid家的成员和雇员是类似的：

- **数据库写入器** Debbie从指定书架上取走图片并将它们放到墙上的框架中。数据库写入器进程在缓冲区中读取发生变化的数据块并将它们写入数据文件。
- **日志写入器** Logan把底片放到重做笔记本里。日志写入进程把数据库执行动作记录到

联机重做文件中。

- **检查点** Chuck在房子里忙碌，把最新的目录号写到每一面有照片的墙上。Chuck同时也以这个数字来更新控制笔记本。典型的，这也发生在每一次Logan存放底片改动笔记本的时候。检查点进程使用检查点SCN来更新数据文件和控制文件。
- **归档器** Archie拷贝Logan的底片笔记本的内容。他把这些拷贝存放到后院的地下室里。归档进程拷贝最近的非当前联机日志文件到参数文件指定的位置。
- **系统监视器** Sid拉闸突然中断的次日早晨，男管家Simon收拾房间，并且把一切恢复正常秩序。系统监视器在实例启动时执行灾难恢复。
- **进程监视器** 女仆Pam在客人不辞而别后清理房间。进程监视器在用户进程失败或断开连接后执行清理操作。

1.5 Oracle数据库操作

我们已经阅读了关于文件和进程的知识，现在我们简要谈谈数据库文件的初始化操作。在典型的数据库操作中有许多活动，Oracle数据库有许多命令和事件。我想强调某些典型的关键事件，它们影响着特定的数据库文件并且担当着数据库操作中的重要角色，尤其是在备份和恢复中。

当数据库被启动时，进程启动、分配内存区域。数据库关闭时，进程停止，内存区域释放。一个被启动的数据库称为打开的数据库，一个关闭的数据库则被称为关闭的数据库。

1.5.1 事务

数据库内容在事务中改变。事务是一个逻辑工作单元，包含在一个或多个SQL语句中。一个事务的所有语句被一起提交或回退。理解发生在事务处理中的数据库活动将帮助你理解备份和恢复机制。一个事务可能包含一个或多个插入、更新或删除语句。这些语句是DML（Data Manipulation Language）命令。在事务处理中，创建重做和回退信息并被跟踪。回退信息作为恢复的条目被写入到回退段。如果更改被回退，则回退段恢复条目允许用户恢复一个事务。因为回退段同表对象一样拥有数据块，当恢复块被更新时，它们也产生重做信息，这被称为撤销的重做。事务的重做信息被作为事务进展记录。一旦事务被提交，可以容易的恢复。

所有的事务信息作为更改矢量被记录在当前重做日志文件中。一个事务可以创建许多重做记录。重做记录是更改矢量的集合，描述数据库原子变化。更改矢量（也作重做矢量）描述数据库一个数据块的改变。要么应用重做记录中所有的矢量，要么一个也不用。单一的事务可能产生一个或多个重做记录，每个重做记录包含一个或多个重做矢量。

- 检查缓冲区，判断目标数据块是否在内存中。
- 如果内存中没有目标数据块，从磁盘中读取。
- 将重做记录写入重做日志缓冲区。
- 创建操作码和数据要求恢复数据块改变到一个恢复段（回退段）。
- 改变内存中数据块的值。

- 在日志缓冲区产生一个重做提交条目并为事务提供一个提交SCN。
 - 清理重做日志缓冲到联机重做日志。
 - 释放保存该事务恢复信息的回退段中的数据块。事务表条目被释放，因此释放回退段中的恢复记录。
 - 最后，重做记录由日志缓冲写入日志文件后，DBWR将变化了的数据块写入数据文件。
- 如果一个用户进程在执行上述任务过程时失败，PMON将处理混乱状态。

数据库事务的周期

一个名叫Jim的用户在雇员申请表中更改了他家里的电话号码，从111-222-3333改为111-222-4444。他通过浏览器上的一张表格来修改。他的电话号码信息会发生什么变化？假定职员表中有一栏：电话，并且职员表位于名为Users的表空间中，该表空间由名为user.dbf的文件组成。

当Jim提交他的改变时，包含他的职员信息行从user.dbf中取出并被读到缓冲区中。在将改变的电话号码放入内存中数据块之前，重做记录被写入日志缓冲。重做记录注释了数据块从旧的状态改变到新的状态。当Jim提交他的改变时，回退段中的数据块被释放，当前重做日志组接收Jim事务的提交。当Jim点击有关运动的Web页查看得分时，联机重做日志被填满，数据库开始写入下一个空的重做日志。包含Jim电话号码变更的数据块在检查点被写入user.dbf。归档器开始拷贝包含Jim事务细节的联机重做日志。一旦归档器的工作完成，Jim新的电话号码存在于两个地方：文件user.dbf的数据块中和归档目标位置的归档日志文件中。

1.5.2 启动

在数据库可供用户和应用程序使用之前，先启动数据库。数据库启动执行若干动作，经历三个阶段，分别是未加载、加载和打开。

- **未加载 (nomount)** 可以把数据库启动的这一阶段看作是Oracle实例的启动。这一启动命令：
 - 读取数据库参数文件。
 - 启动所需的后台进程并按参数文件中的定义分配内存。
 - 将进展情况写入告警日志文件中。
- **加载 (mount)** 在实例启动加载阶段，数据库参数文件中指定的控制文件被读取。记住控制文件将数据库各部分联系在一起。实例从控制文件中找到下列信息，然后将进展写入告警日志文件：
 - 所有数据文件和重做日志文件的名称和位置。
 - 数据库名。
 - 最新系统更改号 (SCN)。
- **打开 (open)** 包含在数据库中的每个联机数据文件在数据库打开前必须被同步。在数据库打开阶段：
 - 所有联机数据文件的头与控制文件信息相比较。

- 所有文件同步后，数据库打开。

数据库异常关闭后打开时，Oracle软件自动执行灾难恢复操作。在这一恢复过程中，SMON前滚数据库事务，这些事务被写入联机重做日志文件但未记录在数据文件中。然后SMON回退所有未提交的数据库事务。恢复信息存储在回退段，用于执行恢复未提交的改变到数据文件中的数据块。在数据库打开之前，所有数据文件必须同控制文件和重做日志一致。数据库启动时灾难恢复动作被记录在告警日志文件中。

注意 不要混淆前滚（roll forward）和回退（roll back）操作。前滚操作读取并应用重做文件中的重做条目来使数据块包含原先改变时的数据。回退操作读取回退段的恢复信息使数据块恢复到先前状态。

1.5.3 日志切换

在日志切换时Oracle软件接收日志切换时的重做条目以改变当前联机重做日志文件。典型情况下，日志切换发生在当前联机重做日志已没有更多的空间用来存储改变的记录时。管理员也可以初始化日志切换。日志切换包含下列任务：

- Oracle服务器标记最新联机重做日志文件组为活动状态。
- 为数据库生成一个检查点。
- Oracle服务器开始写入另一个联机重做日志文件组。
- 一旦切换完成，ARCn开始把最新的联机日志文件拷贝到归档目的地。

1.5.4 检查点

在特定的中断或者重做日志切换时，数据库产生一个检查点。当前内存里的数据被保存到磁盘里。这个检查点确保对当前的数据文件而言，在此之前的任何重做都是不需要的，因为所有改变了的数据块都已经写入了磁盘。

- CKPT进程提示DBWn把所有缓存中改变的数据块写入磁盘中。
- CKPT用检查点的细节来更新控制文件和数据文件头。
- 检查点被记录在告警日志文件里。

1.5.5 关闭

当数据库关闭时，取决于它的关闭方式执行不同的任务：

- **正常（normal）** 在所有的用户离线后发生的正常关闭。
- **事务（transactional）** 事务关闭就是当所有的用户执行完当前的事务后，将用户从数据库上清除。
- **立即（immediate）** 立即关闭就是从数据库清除所有当前用户之后，回退所有未完成的操作。
- **异常（abort）** 异常关闭没有给数据库任何整理的机会。这种方式关闭后需要实行崩溃恢复。

前三种关闭方式中的任何一种都允许数据库整理所有的任务，把所有改变的数据块写入到数据文件并以最新的数据库SCN来同步控制文件和所有的数据文件。当数据库重新启动时，不需要崩溃恢复。在一个干净的关闭过程中，将回退所有没有执行的数据库更改。

Sid家的Oracle操作

Sid家流畅的操作运行和Oracle数据库很相似。

- **打开数据库** 白天在Sid的家，当Sid的家庭执行他们的职责时，他们的房子对所有的客人开放。数据库打开后，用户可以连接，同时实例可用。
- **关闭数据库** 在夜间，没有客人可以进入，Sid全家都上床睡觉。当数据库关闭后，实例不可用。而且用户无法连接。
- **启动** Sid在打开房门迎接客人前，执行一个清晨常规程序。一个Oracle数据库运行任务以准备使用数据库。
- **事务** Sid在房子里处理一张或多张照片。一个Oracle数据库在表的行里进行改变操作。
- **日志切换** Logan切换笔记本来保存底片。一个Oracle数据库停止写入当前的联机重做日志文件，开始写入另一个。
- **检查点** Chuck同步墙壁和控制笔记本。一个Oracle数据库同步数据文件和控制文件。
- **关闭** Sid的家根据已有的规矩准备睡觉。一个Oracle数据库关闭的同时整理文件。

1.6 小结

本章定义了作为本书其余部分的基础的Oracle数据库术语。数据库包含文件、过程以及保存和保护数据的操作过程。我选择用熟悉的事物做类比来描述这些术语。家庭中拍摄、保存照片可以被比喻为数据库存储数据。在继续阅读本书时，可以回顾本章，复习关于体系结构、过程和数据库操作的概念。

习题

回答下列问题，以检查对文件和术语的理解。除非另有说明，每个问题只有一个正确答案。

1. 一个Oracle数据库包含若干数据文件。这些数据文件包含：
 - A. 在数据块中组织的数据行
 - B. 数据库事务中的重做记录
 - C. 为数据库恢复设置的归档事务
 - D. 会话跟踪信息
2. 在数据库中，联机重做日志文件有什么作用？
 - A. 向Web页面发布数据库内容
 - B. 连接时让用户撤销对数据库的改变
 - C. 从数据库事务保存更改矢量
 - D. 为数据库启动存储参数

3. 下面哪种数据库关闭发生时没有执行内存和文件的清理操作?

- A. 正常关闭
- B. 事务关闭
- C. 立即关闭
- D. 异常关闭

4. 哪种数据库事件引起日志切换触发? (选择多个答案)

- A. 数据库检查点
- B. 归档器向归档目录拷贝一个重做日志
- C. 参数文件的刷新
- D. 告警日志文件的日志条目

5. 哪个数据库文件用于定位和同步数据库中其他文件?

- A. 归档重做日志文件
- B. 控制文件
- C. 参数文件
- D. 告警日志文件

答案

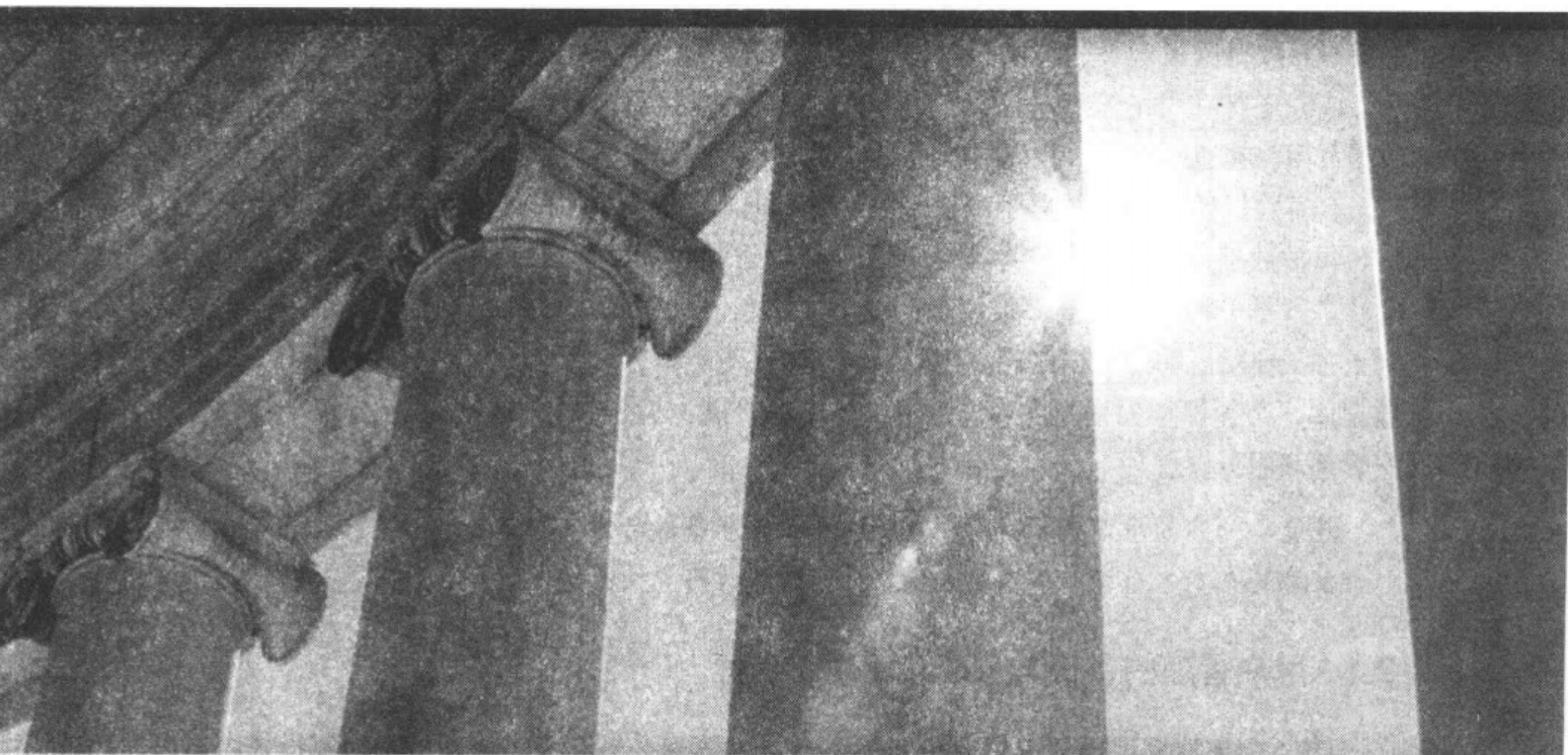
1. A。数据文件保存数据库中的数据。表、索引、存储代码等等存储在数据文件中。

2. C。

3. D。数据库异常终止时, Oracle实例不执行清理和同步操作。

4. A、B、D。日志切换引起检查点操作。日志切换后, 最新的重做日志文件可以被归档。日志切换记录在告警日志文件中。

5. B。控制文件跟踪每一个数据文件、重做日志文件和归档日志文件。



第2章 备份与恢复选项

作为数据库管理员，你的首要目标是：保护数据。当然还有其他责任，如调整、配置数据库，或许还有一些编程工作，以及保持系统中所有数据的正常运行。Oracle提供不同保护措施使数据可用。在这一章中，我们将介绍这些措施，同时了解本书其他章节的内容。

有时数据库会中断并需要修复。故障发生时，管理者和用户需要DBA使数据库重新运行起来。保护数据并使数据库可用意味着可以将数据库恢复到过去的某一状态或者恢复到故障之前的状态。部分或所有数据库被恢复到以前的状态，数据库就可以继续向前运行。使数据库恢复到正常工作状态叫做数据库还原（RESTORATION）和数据库恢复（RECOVERY）。要恢复一个数据库，必须保存数据库内容的拷贝，这个拷贝就称为备份。数据库备份与恢复分为两种：用户管理和服务器管理。

注意 一开始请不要担心不能掌握所有数据库备份与恢复的方法，本章讨论的每一种方法都将在本书后续章节中结合示例进行深入讨论。

2.1 用户管理的备份与恢复

ORACLE数据库用户管理备份策略意味着由管理员自己来实施备份和恢复操作。可以使用操作系统命令备份与恢复数据库文件，使用SQL*Plus RECOVER语句或SQL ALTER DATABASE RECOVER语句来恢复数据库。

在服务器硬盘上，数据实际上是以文件形式存储的，如上一章所讲的，数据库包括数据文件、控制文件、联机重做日志文件、归档重做日志文件、初始化文件和口令文件。拷贝这些文件称为物理备份。在物理备份中，操作系统拷贝命令拷贝数据库文件到不同的位置。物理备份的结果就是拥有Oracle数据库中数据文件的拷贝。物理备份也可指文件层拷贝或结构拷贝。本书中，数据库备份特指物理备份。物理恢复操作使用这些拷贝文件还原并恢复数据库。备份与恢复操作将这些拷贝文件读出或写入介质。介质可以是硬盘、磁带或光盘。

2.1.1 物理备份选项

可以通过多种方法执行数据库的物理备份，备份将文件拷贝到安全的存储介质上。物理备份可以是将当前位置数据库文件拷贝到数据库服务器其他硬盘上，或者将其拷贝到其他服务器的硬盘上。甚至可以将数据库文件作为邮件的附件发送。简而言之，数据库物理备份就是在Oracle数据库中拷贝部分或所有文件的过程。

实际上文件拷贝命令可以通过多种方式执行。可以在操作系统命令提示符下执行拷贝命令，也可以使用供应商提供的备份工具或者是使用脚本文件自动执行备份过程。自动的物理备份过程意味着不必一条一条键入拷贝命令。无论是建立自己的备份命令集，还是使用供应商工具执行备份命令，备份都仅仅只是数据库中文件的操作系统拷贝。

记住，Oracle数据库中包含的文件都是相互关联的。数据文件与控制文件同步，重做日志文

件包含了数据文件内容的正在运行的日志并与控制文件状态一致。这就是说为了使数据库可以被恢复，必须使用某一种方式来完成数据库的备份工作。如果数据库备份如同拷贝文件一样简单不是很好吗？不幸的是，无法这样操作。保持数据库文件的同步和一致使数据库物理备份变得有些复杂。

物理备份在数据库关闭或打开两种状态下均可完成。数据库关闭时的备份是指在拷贝文件之前数据库已经关闭，称为一致性备份。数据库打开时的备份指备份期间用户仍然可使用数据库，称为非一致性备份。

1. 数据库关闭时的物理备份

数据库关闭时的备份是数据库物理备份中最简单的一种方法。一旦数据库被关闭，便成为关闭的数据库。在关闭操作中，Oracle软件会使数据文件和控制文件同步。数据库缓冲区中的信息写进数据库文件，完整的数据库文件被拷贝到新的位置来保存完整的数据库结构。所有的数据库文件作为一个整体用来恢复完整的数据库。所有运行中的事务要么已提交，要么被回退，因此有这样的术语：一致性备份。

注意 数据库关闭时的物理备份也被称为冷备份或一致性备份。

数据库关闭方式对数据库物理备份的质量至关重要。如第1章所述，数据库关闭命令有四个选项，当数据库被“关闭”时，这些选项完成与数据库文件有关的工作。理解选择每种关闭选项时的文件状态有助于避免恢复数据库备份时可能发生的混乱。

注意 应仔细理解四种关闭选项的结果，掌握这些概念，避免以后数据库还原和恢复中的许多混乱情况。

2. 数据库打开时的物理备份

在大多数时间中许多数据库都必须保持可用，然而它们也需要被保护，因此，Oracle提供一种数据库打开并可用状态时的物理备份方式。数据库打开时的备份在表空间层完成。通过将表空间置于备份模式在表空间中备份数据文件，然后使用操作系统命令拷贝表空间数据文件。一旦所有的数据文件拷贝完成，表空间脱离备份模式。要完成整个数据库的打开备份，数据库中的所有表空间均需完成上述过程。在后面的章节将看到，Oracle的恢复管理器使用不同的方式处理打开数据库的备份。

注意 数据库打开时的物理备份也被称为热备份或非一致性备份。

注意 如果备份期间数据库文件一致，不用恢复便可打开数据库。如果备份期间数据库文件不一致，必须完成恢复使文件一致后方可打开数据库。

也许你同我一样有孩子。当妈妈告诉孩子该出去吃晚饭了时，每一个孩子在出门前都必须停止手上的工作。理想的情况是，当父亲说时间到了、该回家了时，孩子停止玩玩具，将玩具整齐的放入玩具箱，关掉电视，并告诉他的朋友该走了。我们希望孩子随后把盘子放到水槽里冲洗并将盘子放入洗碗机中。他关掉玩耍房间的灯，穿上外套，等在房门口，面带微笑。这可以看作正常的数据库关闭或者是清理性关闭（在我的家里，这种情况是反常的）。

通常情况下，当我告诉儿子到时间该出去吃晚餐了，他会抱怨一会儿，然后才走出大门。不收拾玩具，还开着灯和电视。有时，父亲不得不冲入他的房间，抓起他的胳膊，把他拖出大门。这个过程可以与数据库的异常关闭比较（不是清理性关闭）。当我们回家之后又得收拾玩具和房间，同样的情况又将开始。

2.1.2 物理恢复方法

物理恢复，与物理备份一样，通过数据库文件操作将数据库带回到过去的某一状态。物理恢复包含两部分：还原文件拷贝和恢复重做日志文件中的事务。恢复所有数据库文件称为完全数据库恢复。也可以恢复被中断的数据库部分。表空间恢复就是一个或多个被中断的表空间的恢复。数据文件恢复是一个或多个被中断的数据文件的恢复。如果控制文件丢失，可以重新创建它们。归档日志文件用于记录数据文件中的变化。如果重做日志文件丢失，该文件中的数据不能被替换。可以通过恢复备份的数据文件并按创建序列应用重做日志文件来完全恢复一个数据库。

一致性备份与非一致性备份

你在家中听到了玻璃窗被打碎的声音，在屋子后面，你看到一群孩子，惊慌而茫然。当你问他们发生了什么时，你从每个人那里得到回答。如果他们每人都给了你相同的回答，那么你将准备你下一步行动。在这种情况下，他们给了你一致的答案。如果你问发生了什么而得到了若干不同的回答，那么他们的答案是不一致的。在进行下一步打算前，你可能不得不要求某些孩子重新回答。询问更细节的问题直到得到一致的回答。一旦事件澄清，你准备进行补救。

1. 灾难恢复

数据库实例未进行同步操作而突然终止会造成数据库异常中断。实例会因为关闭退出命令导致崩溃。意外的硬件或软件事件，如操作系统错误、CPU硬件、内存或硬盘失效都可使实例崩溃。Oracle软件的意外错误可能导致实例的毁坏，服务器电源的插拔也可导致实例的毁坏。

实例毁坏后数据库启动时，Oracle自动执行灾难恢复，有时也称为实例恢复。因为实际在并行服务器(OPS)环境中，失败的实例由某一存在的实例恢复。首先，数据库使用当前重做日志文件前滚，一旦重做日志要求修改，所有回退段中未提交的事务将被撤销。这两步完成后，数据库便可以被打开使用。必要时，灾难恢复在数据库启动时自动进行。

2. 还原

通过物理操作修复中断的数据库，有时必须将一个或多个数据文件由备份位置拷贝到数据库位置。物理数据库还原指用备份文件替代所有数据库文件。还原使这些文件恢复到备份期间拷贝数据库文件时的状态。

用备份替代表空间称为表空间物理还原。用备份替代数据文件称为数据文件物理还原。从冷备份中还原文件，数据库不必恢复便可以打开。如果将某一数据文件还原到一个正在运行的数据库，则被还原的文件与其他文件不一致，访问该文件将导致数据库错误直到文件被修复。

注意 许多数据库恢复时产生混乱是由于还原使用的文件相互间不一致。出现这种情况通常是由于还原使用的文件来自不同时间的不同备份操作，但没有足够的重做日志使其同步；或者由于数据库备份前没有被彻底的关闭；或者表空间未被置于备份模式时进行了表空间数据文件备份。

3. 恢复

让数据库恢复工作状态，需要恢复所有的数据文件或者恢复到特定的时间点状态。恢复时，Oracle读取重做日志文件中的事务并将其应用于数据文件。既然数据库重做日志中记录了每一个事务，就可以将这些文件应用于先前的事务来恢复数据文件的拷贝。还原和恢复的范围可以是一个或多个数据文件、一个或多个表空间，或者是整个数据库。如果有重做日志文件，就可以还原和恢复部分或整个数据库。所有一致的数据库文件达到相同的数据库状态，恢复就完成了。

注意 术语“恢复”有两种含义：重新应用重做日志的事务，或者通过还原和恢复修复中断数据库的完整过程。还原和恢复一起，使Oracle系统返回正常工作状态。

物理数据库恢复可以通过多种方式完成，包括：完全数据库恢复、不完全数据库恢复、表空间时间点恢复。

(1) 完全数据库物理恢复

典型情况下，将完全恢复一个数据库。这就是说用最近的物理备份替换所有毁坏的数据文件。这种还原将是数据库中所有数据文件的还原。一旦还原数据文件完成，便可以发布恢复命令。Oracle软件读取所有联机数据文件的头信息来辨别其是否有最新的信息。被还原的文件因为是旧版本所以有较低的系统更改号。然后，适当的重做日志文件被找到、读取和应用。以这种方式使用重做日志文件直到每一个数据库事务都被应用。数据库经过这种恢复后打开，将是完全的数据库恢复，不会有任何数据丢失。使用完全恢复，将用到所有的归档重做日志、最新的联机重做日志、以及任何未归档的联机重做日志。

(2) 不完全数据库物理恢复

利用还原和恢复使数据库返回到工作状态也许不需要重新应用每一重做日志文件中所有的事务。如果不希望再发生数据库的操作，就使用这种恢复，如删除一个重要的产品级表。完成整个数据库的不完全恢复，需要使用最新的物理备份还原所有的数据文件。一旦数据文件完成还原，便可发布恢复命令并指定期望的数据库恢复停止的时间点。设定停止参数来指定停止恢复的位置是一个重点，通过为RECOVER DATABASE命令提供日期时间参数（基于时间的恢复）、在参数中指定重做日志文件（基于取消的恢复）或在参数中指定数据库SCN（基于改变的恢复）来定义恢复停止参数。Oracle软件试图寻找到适当的重做日志文件并读取应用它们。以这种方法应用重做日志文件直到满足指定的停止参数条件。数据库经过这种恢复后打开，成为不完全恢复数据库，停止参数后的数据和改变都已丢失。

经过不完全恢复，得到新的数据库。不完全恢复后，打开数据库时必须重置日志，这是因为不完全恢复后，不再需要重做日志文件。使用RESETLOGS打开数据库，数据文件中的信息和重做日志文件发生改变，所有数据文件头获得一个新的RESETLOGS SCN和时间戳。联机重做日志序列号将被重置为1，控制文件被更新以反映新的数据库。

二种时间点恢复方法是：

基于时间的恢复 可以指定期望的数据库恢复的时间。指定恢复停止的时刻，Oracle软件将所有的变化应用于数据库以满足指定时间的状态。这一过程得以进行是因为数据库的每一改变都有日期和时间戳。每一重做日志文件中的每一事务都包含了日期和时间戳。基于时间的恢复结果，数据库中包含了所定义时间点的所有数据。数据库打开时，所有未提交的事务被回退。

基于取消的恢复 可以指定恢复到某一重做恢复文件。这种恢复操作结束时，所有的改变通过重做日志被重新应用于数据库，包括取消操作前最新的重做日志。基于取消的恢复结果，数据库中包含了最后重做日志被填写时的全部数据。数据库打开时，所有未提交的事务被回退。

基于更改的恢复 可以指定恢复停止时的Oracle数据库的SCN。指定数据库恢复停止的SCN，使用恢复操作直到指定SCN。基于改变的恢复结果，数据库中包含了数据库达到所定义系统更改号时的所有数据。数据库打开时，所有未提交的事务被回退。

(3) 表空间时间点物理恢复

还原和恢复特定的表空间到先前某一不同于数据库中其他表空间的时间点也可以修复中断的数据库。表空间时间点恢复（TSPITR）可以快速恢复一个或多个非系统表空间到一个与数据库中其他表空间不同的时刻。TSPITR可以恢复一系列数据集，但是数据集必须是一个完整的表空间，而不能仅仅是一个对象。在数据表被错误的关闭和删除或者数据表遭到逻辑损坏的情况下，TSPITR最有用。使用错误的DML语句而没有改变相关联的对象，会造成数据表逻辑损坏。执行TSPITR，首先还原由系统表空间、回退表空间以及所需的表空间（有时会使用临时表空间）组成的克隆数据库，然后对该数据库执行不完全恢复将表空间恢复到失效前的某一状态。恢复后的表空间对象可以被最初的数据库使用。这些对象既可以由克隆数据库导出到派生数据库，也可将表空间从克隆数据库传送到派生数据库。

数据库还原与数据库恢复

为了帮助理解还原和恢复的不同之处，可以考虑损坏的家具与折断的胳膊的关系。

重建损坏的家具，就是将它还原到最初建造时的状态。通常会将整个家具还原，如果仅仅还原家具的一部分，还原的部分与家具保留的其他部分看上去不大一样。还原一个数据库是指将数据库返回至备份时的状态。如果仅还原部分数据库，那么只有该部分还原至备份时状态。

可以将恢复的过程想象为断骨的康复过程，胳膊折断时，医生首先将胳膊上的骨头恢复到原来的位置，然后，身体产生骨细胞来恢复断骨。恢复始于还原。恢复部分或整个数据库后，使用重做日志信息将数据库恢复到指定状态。

恢复数据库可能是完全的或不完全的。当你的断臂得到了完全的恢复，你可以像从前一样使用它，就好像它从来没有断过。不完全的恢复意味着断臂虽然康复却不能如骨折前那样尽其所能。完全数据库恢复拥有数据库中断前所有的数据，不完全数据库恢复可以找回大部分数据，但不是全部。

2.1.3 复制数据库

备份和恢复活动并不仅在数据库失效时进行，数据库管理员需要创建现行数据库的拷贝。

这一拷贝过程称为克隆或复制数据库。创建复制的数据库是一个备份和恢复操作，不是将数据库恢复到原来的位置，而是将它恢复到一个新的位置。新位置可以在同一台机器上，但多数情况下是将它复制到其他机器上。复制数据库可以用于测试和开发。有时由于机器性能和维护原因需要将数据库由一台机器迁移到另一台机器上，创建复制的数据库可以用来检验当前备份工作能否如期完成。

2.1.4 备用数据库

一些Oracle安装要求数据库失效后的修复时间大约为几分钟或几十秒，在这么高可用性情况下，可以启用备用数据库。备用数据库是原始数据库的复制，它不断地通过使用原始数据库的归档重做日志文件进行更新。使用原始数据库的归档重做日志文件时，备用数据库处于恢复模式。备用数据库可以以只读方式打开。在只读方式下，备用数据库可以被访问但不能被修改和更新。一旦原始数据库发生数据失效，可以迅速打开备用数据库来替代原始数据库工作。在Oracle 9i中，备用数据库特性被称为数据卫士（Data Guard）。数据卫士可以在一些特定硬件平台上的Oracle 8.1.6和Oracle 8.1.7中应用。

2.1.5 逻辑操作

仅拷贝数据库中的数据，可以保护数据库的数据内容。执行逻辑备份可以实现通过拷贝数据来保护数据库数据。数据被拷贝到文件或其他数据库表中。拷贝的数据可以完全地被恢复到原始数据库或其他数据库中。逻辑拷贝无法创建和恢复整个数据库。数据库中的数据被恢复前，数据库必须存在（恢复数据前必须有一个数据字典）。

1. Oracle的导出（export）

Oracle提供拷贝数据库数据的导出功能。导出创建一个文件，包含了用户模式（schema）和模式中数据的拷贝。模式是数据库用户的数据对象集，包括：表、视图、数据链接关系、触发器、过程、函数、包等。导出文件包含了还原数据库中一个或多个用户所有信息的全部命令。从导出的用户模式和数据恢复到Oracle数据库，需要使用Oracle中与导出对应的导入功能。

2. Oracle的导入（import）

通过读取导出文件并且重新创建用户模式和数据到目标数据库中，Import工具从导出文件中恢复数据。可以导入部分或整个导出文件的内容。可以将文件内容导入模式最初被创建的原始数据库，也可将文件内容导入原始数据库的其他用户或其他数据库。

注意 术语“导出”指执行导出功能的过程。术语“导入”则指执行导入功能。提到工具时，名称采用大写。

3. Oracle的LogMiner

每个Oracle数据库事务均被写入重做日志文件。重做日志文件可以被归档和存储以备恢复时使用。Oracle的LogMiner工具读取日志文件并且还原其中的SQL语句。可以使用Logminer提供的SQL语句重新创建或恢复以前数据库事务创建的SQL语句。重做日志文件采用Oracle私有结构格式化，不能被打开、读取，LogMiner可以帮助你还原过程中恢复SQL语句。这种方法比转储

重做内容再进行解码容易得多。

2.2 服务器管理备份与恢复

Oracle数据库的服务器管理的备份和恢复策略由RMAN（恢复管理器——Recoverg Manager）来处理备份和恢复操作。指导RMAN制作备份、执行还原、完成恢复。这一概念是在Oracle8.0中开始提出的。

Oracle的RMAN提供一个数据库备份、还原和恢复的弹性、有效和智能的机制。RMAN完成块层次的数据库备份和恢复。Oracle数据库的最小原子单位是数据块。数据文件、重做日志文件和控制文件均由数据块组成。RMAN向备份位置读写这些数据块。恢复时，RMAN从备份位置读取数据块的拷贝来还原数据库。虽然RMAN也能制作数据文件和控制文件的映像拷贝，但以数据块的方式工作仍是它最有价值的特点。

2.2.1 恢复管理器的特征

如果你是Oracle数据库管理新手，可能无法完全感受到RMAN的益处和特性。这个工具完成保护数据库的多数工作。过去那些艰苦细致的工作现在只需RMAN的几条命令即可完成。

Oracle的RMAN有着多方面的特性供你任意使用，这些性能在保护数据库方面极其有用。

智能化 作为一个DBA，你要做许多工作。创建物理备份，你必须创建一组命令执行备份。需要恢复时，你必须找到适当的备份文件、还原它们，定位归档日志文件、还原它们，并执行恢复。所有这些细节是单调乏味又很难跟上的。然而，不丢失任一个环节是十分重要的。虽然许多数据库管理员编写脚本自动备份，但几乎没有脚本可以完成恢复。并且恢复通常都要求很快完成。有一个程序可以、并且随时可以为你处理所有备份和恢复的麻烦不是很好吗？仅用一条备份数据库的命令就可以备份数据库不是很好吗？或者仅用一条恢复数据库的命令还原数据库，或者仅用一条恢复数据库的命令恢复一个数据库？好，现在可以了。使用RMAN，你可以下达备份数据库的命令。RMAN判断出数据库中包含的文件，将这些文件中的所有数据块拷贝到你指定的位置。需要还原中断的数据库时，只需下达还原数据库的命令。RMAN读取目标数据库的控制文件或它自己的目录，找到备份文件，从正确的数据库备份中还原所需的文件。执行恢复时，RMAN是智能化的软件，它可以判断出为完成用户指定的结果，在文件中必须执行何种操作。

数据块层操作 RMAN数据库备份发生在数据块层次。执行备份时，仅仅拷贝已被数据库使用了的数据块。你可以指定作增量备份。增量备份时，只备份变化了的数据块。通过指定备份什么和在何种层次上备份，可以显著的提高备份的速度，减轻服务器的工作负荷。RMAN减少了备份的数量和创建备份的时间。举例来说，20GB的数据库有5GB的数据，RMAN将仅仅备份包含5GB数据的块，而不存储15GB的空数据文件。

备份范围 运行备份和恢复时，可以指定希望执行操作的数据库部分。可以只备份控制文件、一个或多个数据文件、一个或多个表空间、整个数据库、部分或所有归档日志文件。恢复同样如此，可以只恢复数据库失效的部分。

目录 关于数据库备份和恢复中的争论之一是记录数据库备份过程中的事件和细节。作为

数据库管理员，你会经常的自动执行备份。需要恢复时，你会匆忙的寻找最近备份的细节。大多数DBA（就像我）都太懒或太忙而没有对备份的细节作认真的记录。RMAN对数据库操作时，备份信息被写入备份数据库的控制文件。备份时间、备份的文件以及备份层次等细节被存储到正在备份的数据库的控制文件中。控制文件丢失，备份记录细节便也丢失。更重要的是，RMAN丢失了恢复过程中所需的备份信息。RMAN还可以使用目录存储数据库备份和恢复细节。这个目录是表、视图、序列的摘要，包含了所有数据库备份的历史。你可以将这个目录放在网络上的任意一个Oracle数据库中。需要恢复时，RMAN使用目录中的信息还原和恢复中断的数据库。

介质管理层 备份操作过程中，通常要将备份数据写入磁带设备中。配置介质管理层，使RMAN和服务器的磁带设备间的命令和数据容易通信。一旦配置好介质管理层，就可以发布读写磁带设备的RMAN备份和恢复命令。磁带设备可能是一个磁带或者是一个磁带阵列。对RMAN来讲，都是一样的。同磁带设备的接口由介质管理层和磁带供应商的软件来处理。

存储脚本 发布RMAN命令时，你可以启动这个程序（RMAN），在RMAN提示符下输入命令。将命令输入一个文件，在RMAN中调用它，你就可以自动完成重复的任务。你也可以创建一个一组命令脚本并将其保存到恢复管理器目录中。这个存储的脚本可以在RMAN命令行下调用，就如同在SQL*Plus提示符下调用存储的过程一样。

SQL解释器 在RMAN提示符下，可以键入SQL命令完成数据库操作。例如，可以关闭数据库、启动数据库，或者改变数据库归档日志模式。这些命令使你能自动操作函数，完成数据库维护。所有这些都是RMAN提示符下完成的。

性能 对一个大型数据库来说，备份是耗费时间和资源的事情。RMAN提供不同的参数来控制备份过程中创建的文件数量和大小、同时被读取的文件数量，以及每秒被读取的最大数据量。RMAN使用通道完成备份和恢复。通道是到数据库的连接或到备份介质的流连接。当速度是重要性能指标时，你可以指定备份的通道数量来提高吞吐量。在数据库可用性是重要指标时，你可以在数据库打开时备份它同时供用户和应用程序使用。按照RMAN处理打开数据库备份的方式，没有必要将表空间置于热备份方式。因此，没有额外的重做日志产生，这样比用户管理热备份完成得更好。

如果暂时作为一名数据库管理员，你也许已经开始熟悉目前的备份和恢复工具集了。我想说，许多经验丰富的DBA，他们打算学习RMAN却还没有开始。也许RMAN提供的性能可以促使你花时间学习并配置自己数据库环境中的RMAN。

RMAN还有许多其他的用处，我将在本书第三部分恢复管理器中详细解释和示范这些性能。

2.2.2 恢复管理器的功能

使用恢复管理器可以完成许多数据库任务，这些任务同你在使用物理数据库恢复操作时一样。使用RMAN，每一步操作将变得简单。

数据库文件拷贝 如果想创建数据文件或控制文件的映像拷贝，RMAN可以完成这项任务。这同前面讲述的数据文件的热备份是一样的。

复制数据库 使用RMAN，可以创建整个数据库的完整拷贝。这种数据库复制工作可以在

同一机器或在不同机器上完成。RMAN中的模拟克隆命令是复制数据库命令。使用RMAN, 拷贝整个数据库只需两个单词: DUPLICATE DATABASE (还需要配置通信通道, 也许还要使用其他的恢复命令)。复制的数据库可以使用同当前数据库相同的名字或者可以在复制过程中更改名称。

备用数据库 RMAN也可以创建备用数据库。这个备用数据库可以在主数据库中断时作为失效恢复的数据库使用。同样, RMAN使一些复杂的事情变成简单和直接的操作。

表空间时间点恢复 表空间时间点恢复可以由RMAN命令完成。使用RMAN, 可以使除了指定的一个或多个表空间之外的所有数据库部分保持当前状态。通过这种方式, 将数据库丢失数据减到最少的同时将表空间的内容恢复到失效或错误发生的时刻。

2.3 Oracle备份与恢复

本书中, 将介绍Oracle 8i中有关用户管理备份和恢复以及服务器管理备份和恢复的内容。但是要记住保护数据库时, 确实有一些选项需要考虑。已经发布的Oracle 9i版有一些新的特性和选项。也可以通过Oracle的企业管理器完成许多数据库备份和恢复选项。最后, 供应商软件解决方案也可以帮助你保护数据库。

2.3.1 Oracle 9i的增强特性

本书中, 我们基于Oracle 8i (8.1.7节) 讨论Oracle备份和恢复, 但是, 一些新的有用的特性出现在Oracle 9i中。举例来讲, Oracle 8i备用数据库功能在9i中被显著加强, 并作为Oracle Data Guard (数据卫士) 重新包装。这一特性在8.1.6节和8.1.7节中仅作为硬件平台的一小部分提供。Data Guard将复杂任务自动化并且提供了加强的监视、告警和控制机制。新模块帮助你幸免于错误、讹误、和其他可能破坏备用数据库的灾难性事件。同样, 故障修复时间要求改善, 如硬件和操作系统的检修时间, 使用Oracle 9i备用数据库这一时间将被极大的缩短。

RMAN也有所改进。现在, RMAN允许DBA指定可能遭到破坏的特定数据块还原与恢复。当RMAN在块介质恢复中初次检测到丢失或破坏的重做日志时, 它并不立即停止恢复, 但是它会标记错误。这样做是因为在重做序列中数据块可能成为新建的数据块。数据块被新建时, 数据块中所有的重做日志变得毫无关联, 这是因为重做日志应用到数据块原来的形式中。例如, 如果数据表的存储单元被释放, Oracle可以新建一个数据块, 数据块被重新使用。RMAN的另一显著增强性能是配置备份拷贝的保留策略。保留策略起作用时, RMAN认为备份和数据文件、控制文件的拷贝是过时的, 在将来的恢复中不再需要。你可以定期的或经常的发布过时的备份报告并通过配置保留策略删除这些备份。

上述这些Oracle 9i增强特性并不能涵盖所有的改进。要了解全部的增强功能, 可以参考Oracle手册中Oracle 9i 数据库新功能。

2.3.2 Oracle 企业管理器

每一种ORACLE本地备份和恢复操作都可以在操作系统的提示符下被调用。可以用SQL*Plus在操作系统提示符下完成物理备份和恢复的步骤。执行诸如文件拷贝这样的操作系统

命令时，在操作系统提示符下键入命令；执行数据库命令时，在SQL*Plus提示符下执行。逻辑备份和恢复操作在操作系统提示符或SQL*Plus提示符下被调用。Oracle导入和导出在操作系统提示符下被调用。恢复管理器在操作系统提示符下被调用。

为了帮助你完成数据库管理工作，Oracle开发了Oracle企业管理器（OEM）。用Java编写，OEM提供用户图形界面来执行大多数的数据库管理任务。如果你想完成物理数据库备份，你可以使用OEM控制数据库的启动，配合操作系统命令拷贝文件。可以使用OEM完成物理数据库恢复。可以使用OEM产生逻辑备份方案，可以调用导入、导出数据拷贝。可以通过OEM的图形用户界面创建和执行RMAN脚本。

OEM还允许你为将要进行的多数备份和恢复操作创建脚本。你可以在OEM中创建脚本并且在确定的时间执行它们。你也可以在OEM之外创建脚本，而通过OEM的任务安排功能来控制这些脚本的执行。在本书中，我将讨论在命令提示符下的备份和恢复操作而不是OEM下的备份和恢复。

2.3.3 供应商的解决方案

购买并采纳某一Oracle数据库方案时，随之提供了所有保护数据所需的备份和恢复工具。如SQL*Plus、导出和导入工具，使用它们并不需要额外的许可。恢复管理器也是如此。也许你希望了解来自其他供应商软件公司的购买选项。

先进的性能、可配置的部件，Oracle数据库可能十分复杂。你的公司可能在数据库中保存了重要的公司数据资产，你的组织可能变得完全依赖于Oracle系统的有效性。鉴于这些至关重要的要求和系统的复杂性，软件供应商提供解决方案帮助你完成备份和恢复工作。供应商的解决方案需要许可证费的开销。可以找到一个或更多供应商解决方案，它们增强了Oracle恢复方案，使你的工作变得更加容易。

1. 硬件解决方案

有一些硬件选项在硬件失效时保持系统高有效性。标准的硬件解决方案是使用独立磁盘的冗余阵列（RAID）。支持Oracle数据库软件运行的硬件有时会失效。存储Oracle数据库文件的磁盘可能发生故障。为了在硬盘中断时防止文件丢失，RAID提供许多保护存储方案。最常采用的是RAID 0+1和RAID 5。RAID 0+1提供磁盘内容的镜像和分带。如果你的数据库存储在5个磁盘上，那么你需要另一套5个磁盘来镜像数据。数据被分带到5个镜像磁盘上。磁盘丢失时，便可以使用镜像磁盘。RAID 5通过奇偶校验位使带集数据可以跨越多个磁盘。磁盘丢失时，可以通过存储在其他磁盘上的奇偶校验数据重新创建丢失的磁盘目录。

现在，许多硬件供应商提供数据镜像、带集、增加奇偶校验的不同组合。这些技术，结合大型的电源备份磁盘缓冲，让DBA不必太担心当磁盘崩溃时，他们所保护数据库数据会丢失。

警告 使用RAID或镜像磁盘方案是一种高可靠性的方案而不是备份方案。如果讹误或者用户的某一错误被镜像到另一磁盘，会发生什么情况呢？讹误和错误被再现在镜像磁盘上。你仍然需要一个好的、健壮的备份和恢复策略。硬件失效控制和高可靠性不是备份和恢复，它在硬件失效的时候提供可用性。

2. 软件解决方案

许多软件供应商都已开发出了功能强大且易于使用的备份和恢复方法。有些方案有助于物理备份和恢复。有些供应商提供逻辑备份方案，有些供应商提供软件使其备份方案同Oracle恢复管理器协调工作。下面是我所知的公司列表以及他们的解决方案。这个列表不包含全部内容，并且在本书中我也不认为某个供应商的解决方案一定优于其他方案，但是你可以调查这些产品，也许你会发现世界其他公司也提供类似有帮助的解决方案。

- BMC SQL-BackTrackTM为Oracle产品提供的备份和恢复产品。www.bmc.com。
- Computer Associates ArcServe提供Windows操作系统中文件系统的备份。www.ca.com。
- Hewlett-Packard OmniBack提供文件系统和联机数据库备份。www.hp.com。
- Legato Networker®和Networker BusinessSuiteTM提供文件系统和Oracle数据库备份保护，www.legato.com。
- Veritas NetBackupTM提供文件备份和联机数据库备份。www.veritas.com。

这些公司大多提供可配置模块。你可以购买基本产品和附加模块，使联机备份和Oracle的恢复管理器一体化。某些产品在特定的操作系统上可用，而其他产品在大多数流行的Oracle操作系统上都可用。每一种产品提供备份和还原Oracle数据和文件到磁带的备份设备。这些公司仅仅为你提供了少数选项，为了解更多的产品和公司，可以查找它们的主页或直接联系他们。本书中我将着重讨论Oracle软件提供的Oracle备份和恢复解决方案。你可以在Oracle网站上找到兼容介质管理层的磁带供应商。

2.4 小结

这一章中，我们对Oracle DBA使用的备份和恢复选项有了一个总体认识。你可以自己通过物理文件操作或逻辑数据操作来使用备份和恢复选项。你通过恢复管理器让服务器使用备份和恢复操作。RMAN程序允许你以最小的数据库原子单元（数据库块）来备份和恢复数据库。

这些解决方法之间并不相互排斥。在实际系统中，可能会选择几种方法的组合来备份数据库。在修复一个中断的数据库时，可能需要多个可用的选项来执行恢复操作。

本章是对全书的一个总览。本书第二部分介绍了基本的用户管理备份和恢复操作，第三部分讲述了通过恢复管理器执行的基于服务器的操作。第二部分学习的概念和细节是第三部分通过RMAN执行操作的重要基础。下一章中，我将介绍建立一个小的数据库，通过这个数据库练习本书后面章节中讲到的内容。

习题

回答下列问题，检查对术语、备份选项和恢复操作的理解。除非特别说明，每一个问题只有一个正确答案。

1. 可以通过物理备份备份数据库。通过下列哪种操作可以完成物理备份？
 - A. 关闭数据库
 - B. 从数据库中提取数据
 - C. 在数据库中制作数据库文件的备份

D. 打开时运行导出程序

2. Oracle导出功能可以将数据库数据拷贝到数据库文件集之外的文件中。这种导出属于哪种类型的备份?

- A. 物理备份
- B. 冷备份
- C. 逻辑备份
- D. 恢复管理器备份

3. 恢复数据库时, 存储在重做日志文件中的数据库事务被应用于现行数据库。恢复数据库文件指用先前的物理备份替代数据库中的文件。为什么恢复可能需要被恢复的文件?

- A. 数据文件可能来自其他数据库
- B. 变化必须反映到被恢复的文件使其和数据库其他部分同步
- C. 被恢复的数据文件是导出文件
- D. 数据库没有数据文件

4. 下列哪些任务恢复管理器不能完成? (选出所有正确答案)

- A. 完成完全的数据库恢复
- B. 创建复制和备用数据库
- C. 进行表空间时间点恢复
- D. 煎鸡蛋

5. 恢复管理器使用目录跟踪备份和数据库信息。为什么要使用RMAN的恢复目录?

- A. 如果目标控制文件丢失, 它可以作为一个替代
- B. 排序存储
- C. RMAN可以更有效地运行
- D. Oracle如是说

答案

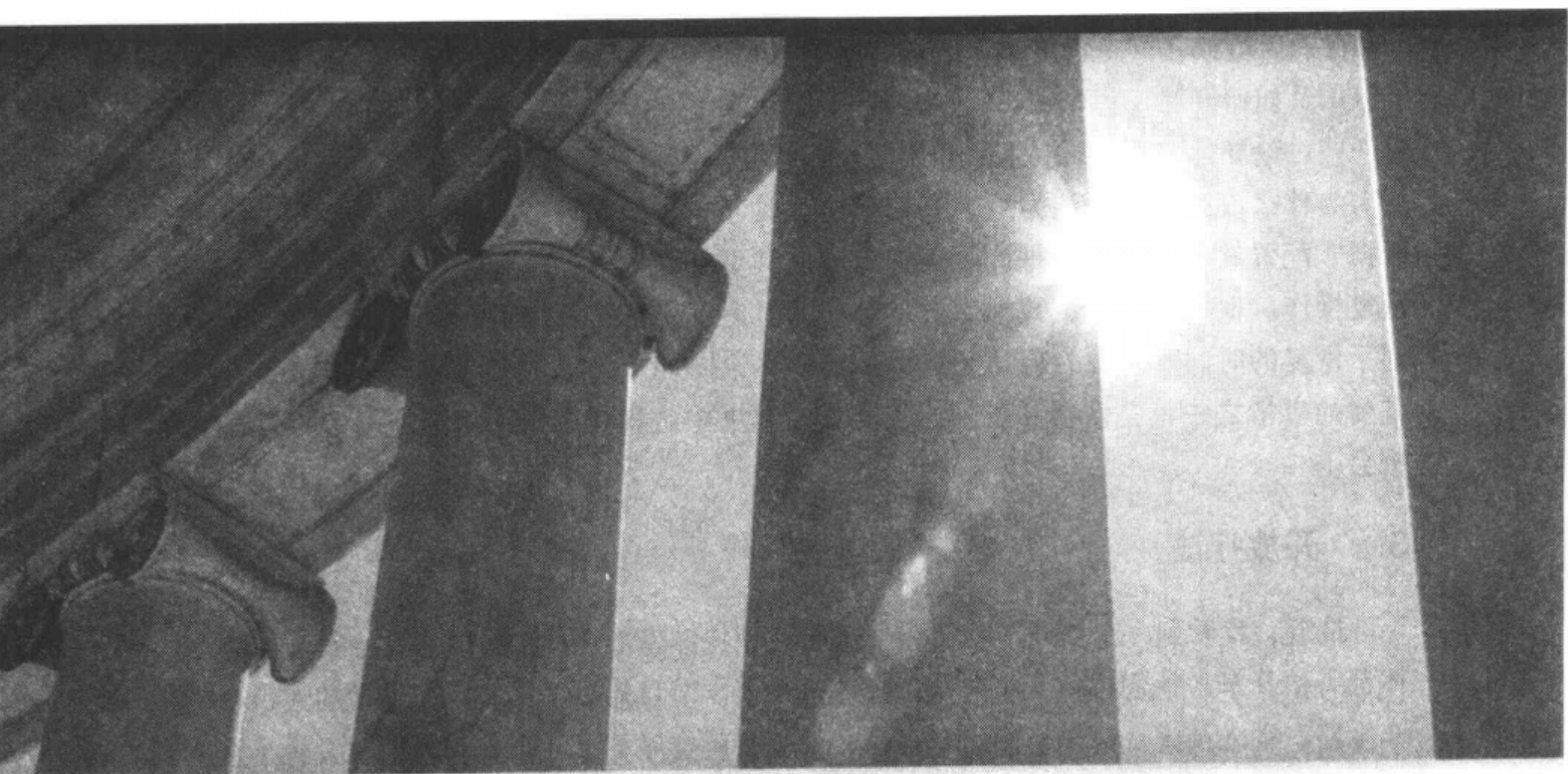
1. C。数据库中部分或全部数据文件的拷贝组成了物理备份。

2. C。导出工具向一个文件中拷贝数据库的模式和数据。这种数据提取过程是一个逻辑操作。

3. B。如果必须替换丢失的或损坏的数据文件, 应首先还原该数据文件的拷贝。然后使用重做信息直到所有的更改都被应用到该文件, 就恢复了该文件。

4. A、D。恢复管理器不能煎鸡蛋。然而, RMAN可以执行完全数据库恢复、表空间时间点恢复、创建副本数据库或备用数据库以及其他有用的备份与恢复任务。

5. A。在RMAN备份期间, 备份信息存储在目标数据库的控制文件中。如果目标控制文件丢失了, 则RMAN在恢复时就不能使用恢复目录。运用恢复目录有许多优点, 包括可以在一个目录中管理许多数据库以及备份信息, 独立于控制文件, 除非进行删除, 否则目录一直保留着备份记录。



第3章 专项练习

在书店里或在网上售书站点查询，可以找到许多关于Oracle软件的书籍。大多数书籍介绍了SQL及如何开始使用Oracle，许多书讨论了Oracle的调整。有些书针对特定的操作系统，还有一些讨论如何使用Oracle工具和语言进行开发。但是没有多少书籍涵盖了Oracle的备份和恢复。为什么没有呢？我认为绝大多数DBA太忙了。他们忙着安装软件、安装补丁程序、迁移数据、配置软件、调整数据库、组织SQL使其更有效率等，以至于他们没有时间和机会执行恢复操作。他们计划测试数据库的备份功能但还没来得及做。实际上，许多DBA都成功地规划了恢复的时间和资源，但这还远远不够。不论DBA有什么日常要求，都应该投入精力的最重要的事情之一就是恢复和备份。因此，在本书中，我将讲解备份与恢复，并且提供你练习机会。

3.1 开放疗法

最近，我看到一个新闻节目，讨论患有一种特殊恐怖症的个体的强迫性紊乱行为：这些个体都害怕开车，他们害怕开车时车子会碾过某人。节目播放了这些行为异常的人小心翼翼地驱车上路，驶入街道。还没有驶过邻居家，驾驶者就会停下车，下来检查周围看看是否压到了某人。一旦发现车下并没有人，驾驶者才继续驶上街道，仍旧小心翼翼地避免撞上行入。在驾驶的过程中，他们多次停车、下车、检查是否碾过某人。这个极端的恐惧症削弱了病人的精力，限制了他们的行为。

节目主持人讲述了人们在这种特定状况下的治疗方法。一位心理学专家讨论如何用开放疗法（Exposure Therapy）帮助这些恐怖症病人。节目中病人在一个大的空旷的停车场开车，这个心理学家坐在乘客的位置上。医生在停车场的人行道上放置一个沙袋，病人被指示开车碾过沙袋。病人鼓起所有的勇气，才能驾驶车辆碾过沙袋。心理学家对病人解释说，如果他的车子压上行入，他会有一种本能的感觉就如同驱车压过沙袋一样。经过一系列的开放治疗，患者可以面对并克服恐惧了。

在Oracle数据库保护中，这种驾驶的恐惧如何克服呢？好问题！答案就是开放疗法。数据库管理员担负着保护公司重要资产（公司数据）的责任。数据库管理员有责任保证数据库的有效运行来支持公司必须的应用程序。尽管这一责任落在DBA的肩上，但他们都很少作各种失败想定下的数据库恢复练习。给自己一个机会，完成在不同情况下的多种数据库恢复。你可以首先练习小规模地从备份到恢复操作来建立你的信心和经验。一旦练习成功，你就可以在实际的Oracle系统中完成数据库恢复操作了。

专家说过，读过的记住少许，听到的记住部分，做过的记住大多数。虽然你不做正文练习也可以使用本书，但完成练习肯定更有价值。本章讲述如何为这些练习创建数据库。

在本章中，将安装Oracle软件、创建三个数据库并在其中一个数据库中添加两个用户。图3-1是即将创建的目录结构。运行一个数据库，首先需要安装Oracle软件。

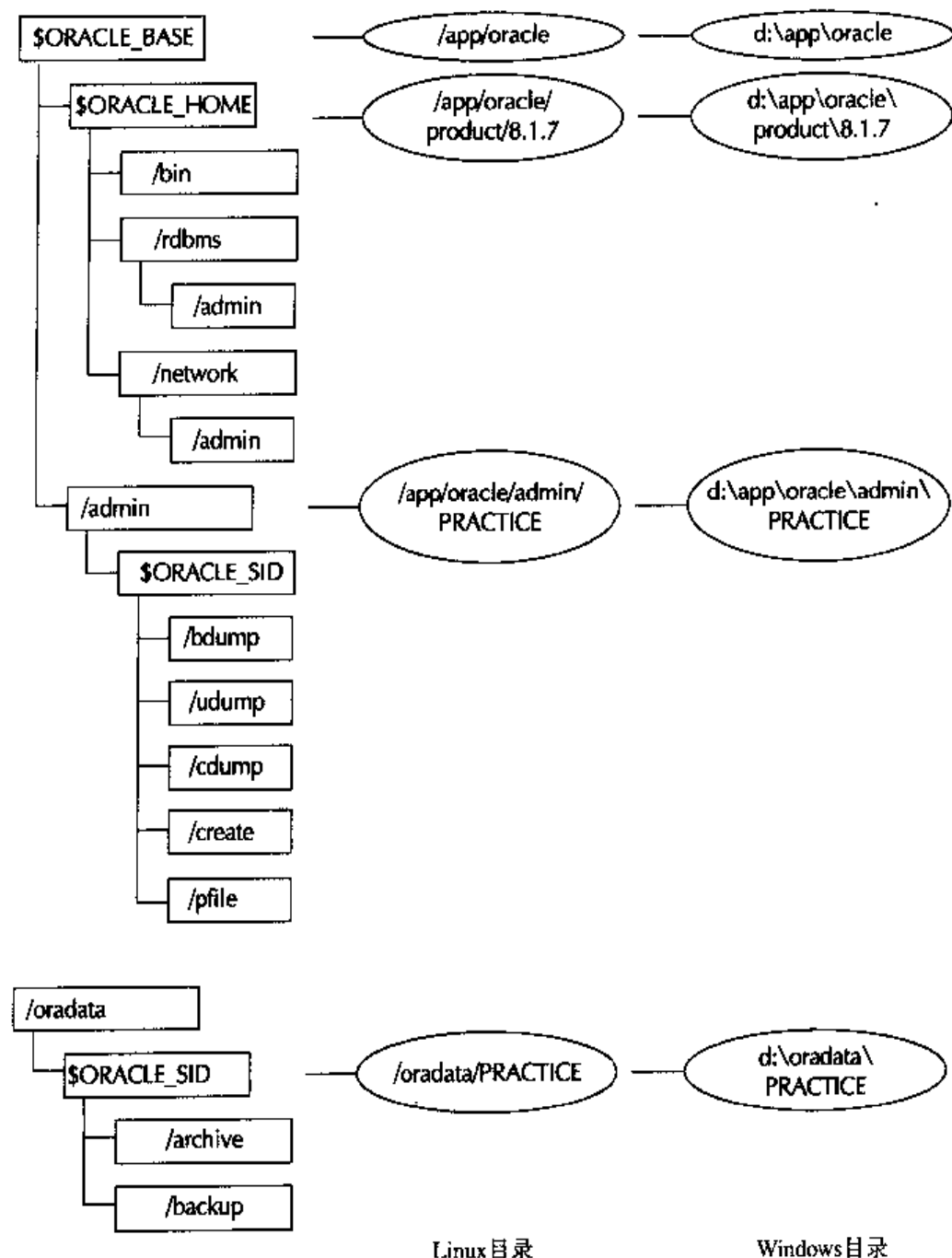


图3-1 本书中Oracle软件、数据库、备份/恢复练习的目录结构

3.2 操作所需的机器

需要一台计算机安装一些数据库，你必须有权访问办公室或家里的机器。如果不能使用类似Sun或HP这样功能强大的计算机，你可以使用初级的PC机，工作中多余的PC或家中的PC。需要硬盘足够大来装入Oracle软件和若干数据库。

本书使用了Oracle 8i (8.1.7) Enterprise Edition Release 3。这一版本的安装需要1GB的硬盘空间。每个安装的数据库大约为200MB。你的机器至少要有2GB的剩余空间才刚刚可以满足本书的工作要求。还需要大量的内存来满足数据库的使用要求。我建议至少使用128MB的RAM。

在各章节的某些练习中，计算机将会同时运行三个数据库。CPU完成数据库计算和其他任务。一个快速的CPU对你的工作是有帮助的，但并不是必须的。

每一章都有练习和作业，包括教程、命令、输出和结果。你可以在阅读本书时上机完成作业。在每章的最后，有疑难解答，还有一些问题来检查你对内容的理解。章节小结和Oracle手册有关，从手册中可以得到更多的信息。

当你为练习挑选了一台机器后，你还需要选择一个操作系统来在这台机器上运行数据库。我使用的两个操作系统是Linux和Windows NT。

3.2.1 环境变量和Oracle软件

Oracle管理自己的软件时，大量使用了操作系统环境变量。环境变量是分配给操作系统中某一名称的值。软件可以使用名称来调用变量的值。

Oracle软件使用了许多环境变量，非常重要的三个是ORACLE_HOME、ORACLE_BASE和ORACLE_SID。ORACLE_HOME指定了你希望运行的Oracle执行程序的目录位置。一台计算机上可能安装了若干个版本的Oracle软件，大多数Oracle程序查询ORACLE_HOME来决定运行何种版本的软件。ORACLE_BASE定义了数据库管理文件安装的基本目录。ORACLE_SID定义了你希望在其上工作的本地Oracle数据库名称。如果是Linux环境，在命令提示符下设置变量。如果是Windows环境，则可以在命令提示符下或注册表中设置。

3.2.2 Linux环境下的Oracle服务器

你可以在机器上毫不费力地安装Linux操作系统。Linux运行在基于Intel的个人计算机上，它可以从Internet上下载或者从计算机商店购买。Linux是类Unix操作系统。因此，Linux上许多关于Oracle安装的配置与在Sun Solaris、HP UNIX、IBM AIX和其他基于Unix的服务器上的配置相同。你将发现，在Linux上安装Oracle为你提供了一个省钱的类Unix环境来学习和实践Oracle数据库恢复。

你可以选择任何与Oracle 8.1.7兼容的版本。我选择SuSE Linux 7.2 Personal。我安装了第二块硬盘为了我的孩子玩游戏，然后在新硬盘上安装了Linux。

如果你选择了在Linux上安装Oracle，安装的目录如表3-1所示。表3-1描述了文件内容、完整的目录路径位置和访问的简化方式——通过环境变量描述目录。

表3-1 Linux下的目录

目录内容	Linux目录	描述目录的环境变量
Oracle基本目录	/app/oracle	\$ORACLE_BASE
Oracle软件	/app/oracle/product/8.1.7	\$ORACLE_HOME
数据库管理文件	/app/oracle/admin/\$ORACLE_SID	\$ORACLE_BASE/admin/\$ORACLE_SID
数据库文件	/oradata/\$ORACLE_SID	/oradata/\$ORACLE_SID

3.2.3 Windows NT环境下的Oracle服务器

我选择Windows NT Service Pack 6作为练习Oracle数据库恢复操作的平台，原因是大多数机

构都使用Windows NT，而且许多PC机都预先安装了Windows NT。因为Oracle软件在Windows 2000下的操作与Windows NT下的操作方式相同，所以你也可以用Windows 2000来完成章节中的练习。当在Windows机器上安装了Oracle，就创建了如表3-2所列的目录。若干数据库可以同时在一个安装软件上运行，每一个数据库都有自己的管理和数据库文件目录。

表3-2 Windows NT下的目录

目录内容	Windows NT目录	描述目录的环境变量
Oracle基本目录	D:\app\oracle	%ORACLE_BASE%
Oracle软件	D:\app\oracle\product\8.1.7	%ORACLE_HOME%
数据库管理文件	D:\app\oracle\admin\%ORACLE_SID	%ORACLE_BASE%\admin\%ORACLE_SID%
数据库文件	D:\oradata\%ORACLE_SID%	D:\oradata\%ORACLE_SID%

3.2.4 UNIX环境下的Oracle服务器

大多数Oracle产品安装在UNIX操作系统上。如果你正在阅读本书并能访问家里或办公室的一台UNIX机器，也可以使用它来实践本书的练习。UNIX下的命令和目录结构与前面讨论的Linux结构完全相同。

在本书中，示例通常首先描述Linux下的实现。大部分Linux命令在Windows NT中也同样提供。如果在Windows NT中数据库操作有不同的方法，我会指出不同之处。我选择Linux作为本书默认的操作系统。

3.3 安装Oracle软件

现在已经有了一台可以工作的机器，下一步就是安装Oracle软件。这一节的第一个练习就是在你的Windows NT或Linux机器上安装Oracle软件。

练习3.1：安装Oracle软件

安装软件包括三个主要练习。在每一个练习的开始我都会估算出练习的时间，以便于安排时间。

任务描述	时间（分钟）
1. 寻找Oracle软件	—
2. 安装Oracle软件	30
3. 确认软件安装	10
总计时间	40

我在这儿提供的安装指导是简单的并针对本书使用范围的。如果你想学习更详细的内容或者遇到了问题，可以参考Oracle文档，找到Oracle8i在Linux或Windows NT下的安装指南，获取详细信息。可以在docs.oracle.com上在线获取这些文档或在安装光盘上找到它们。

任务1：寻找Oracle软件

在机器上安装Oracle，需要Oracle软件。可以从光盘安装Oracle软件或者从<http://technet>.

oracle.com下载。如果是OTN (ORACLE TECHNOLOGY NETWORK) 注册用户, 则可以下载试用的Oracle软件。可以选择Linux或Windows NT下的Oracle服务器8.1.7企业版。你也可以安装其他版本, 但是可能无法跟随某些练习。

注意 安装软件时, 确保阅读并同意Oracle的许可条款。

任务2: 安装Oracle软件

安装Oracle软件时, 将有一个基于JAVA技术的图形界面在整个安装过程中给出指导。这个程序叫做通用Oracle安装器 (Universal Oracle Install)。在软件安装过程中将看到几个屏幕显示, 回答屏幕上的问题, 如下面描述的那样安装软件。确定将Oracle软件安装在机器指定的目录上, 以便可以完成本书中的练习。安装软件时, 遵循下面的指导原则:

- 1) 安装Oracle企业版8.1.7。软件版本的后两位数字可能不同, 但是应确保前三位数字是8.1.7。
- 2) Linux下安装时, 在路径 (Path) 字段输入/app/product/oracle/8.1.7作为ORACLE_HOME目录。
- 3) 在Windows NT下, 为Oracle Home Name字段键入817。在路径字段, 键入D:\app\product\oracle\8.1.7。Windows NT下的Oracle命名为Oracle Home, Linux与它不同。
- 4) 在选择安装类型时, 选择最小安装。软件安装完成后, 将创建已经定制的数据库。

其他的屏幕选择时, 接受默认值设定但不要创建数据库。在下面的练习中将完成数据库创建。

任务3: 确认软件安装

一旦安装了软件, 就可以在硬盘上找到它。启动命令提示符或文件管理器浏览Oracle软件目录。在\$ORACLE_HOME目录下你将看到bin、network和rdbms目录。Windows下安装完成后, 在开始菜单上会显示新的条目。

可以设置环境变量来帮助你管理软件和数据库。在Linux操作系统命令提示符下, 使用export命令来设置环境变量:

```
LINUX> export ORACLE_BASE=/app/oracle
LINUX> export ORACLE_HOME=/app/oracle/product/8.1.7
LINUX> export ORACLE_SID=PRACTICE
LINUX> export PATH=PATH:$ORACLE_HOME/bin
```

你可以在注册登录脚本 (如profile) 中添加上述命令行, 在每次登录时设置环境变量。

在Windows NT中, 可以使用set命令定义环境变量:

```
WINNT> set ORACLE_BASE=d:\app\oracle
WINNT> set ORACLE_HOME=d:\app\oracle\product\8.1.7
WINNT> set ORACLE_SID=PRACTICE
WINNT> set PATH=%PATH%;%ORACLE_HOME%\bin
```

或者可以按照Oracle8i管理员指南中的描述修改注册信息。在本书中, 我将使用在此已经定义好的环境变量。

3.4 练习所需的数据库

每一章中都包含了使用Oracle工具和命令进行备份和恢复的练习。在本书学习过程中将以不同的方式操作三个数据库。本书中将反复使用同样的数据库、表空间、数据表和用户。

- 1) 名为PRACTICE的数据库——本书中主要使用的数据库就是名为PRACTICE的数据库。这个数据库包括200MB的数据文件和3MB的重做日志文件（三个）。这个数据库将被备份、中断、修复和复制。
- 2) 名为RCAT的数据库——恢复管理器将使用一个目录。这个目录是存储在数据库中的一个模式。Oracle 8i文档中的示例都使用了一个名为RCAT的数据库（恢复目录的缩写）。在本书中也采用同样名称。你将创建一个名为RCAT的数据库并在数据库中维护一个名为RMAN817的恢复目录用户。本书将在恢复管理器的有关章节中使用这个数据库。
- 3) 名为CLNE的数据库——在本书的某些章节中，你将创建PRACTICE的副本数据库，这个复制的数据库被命名为CLNE。

开始在这些数据库上工作之前，你必须先创建它们。我将描述如何创建数据库，并在下一个使用Oracle数据库配置助手练习中讲述它们支持的服务。

练习3.2：创建PRACTICE数据库

对初学者来说，创建数据库的一个简单的方法是使用Oracle的数据库配置助手（Database Configuration Assistant）。这个JAVA应用程序将接收你指定的参数并创建所有数据库文件和所有支持该数据库所必需的目录结构。通过一系列的交互行为，该工具就可以为你创建一个数据库。你也可以用配置助手将创建数据库命令存入脚本文件，以后你就可以在SQL*Plus提示符下运行这个脚本文件。

注意 如果你希望用自己的脚本文件来创建数据库，那么按照本节练习的指导创建名为PRACTICE的数据库。表3-3和表3-4作为你创建数据库的指南。

第一个练习是创建名为PRACTICE的数据库，完成这一任务的步骤如下：

任务描述	时间（分钟）
1. 启动数据库配置助手	5
2. 配置数据库配置助手	10
3. 创建数据库	30
总计时间	45

我希望你以指定的方法创建PRACTICE数据库。在本书的每一章，我将详细说明对这些特定位置文件的操作。如果你有完全一致的文件，会方便你对照自己的步骤和结果，这样更有助于你跟随本书完成练习。创建拥有表3-3定义的表空间的数据库。

表3-3 PRACTICE 表空间参数

Tablespace	Size MB	Datafile	Datafile	Auto Extend	Storage:				
					% Increase	Initial KB	Next KB	Min	Max
SYSTEM	100	/oradata/PRACTICE/ system01.dbf	d:\oradata\PRACTICE\ system01.dbf	Off	50	64	64	1	4096

(续)

Tablespace	Size MB	Datafile		Auto Extend	Storage:				
		Linux	Windows NT		% Increase	Initial KB	Next KB	Min	Max
TOOLS	20	/oradata/PRACTICE/ tools01.dbf	d:\oradata\PRACTICE\ tools01.dbf	Off	0	64	64	1	4096
USERS	10	/oradata/PRACTICE/ users01.dbf	d:\oradata\PRACTICE\ users01.dbf	Off	0	64	64	1	4096
RBS	20	/oradata/PRACTICE/ rbs01.dbf	d:\oradata\PRACTICE\ rbs01.dbf	Off	0	64	64	4	4096
INDX	20	/oradata/PRACTICE/ indx01.dbf	d:\oradata\PRACTICE\ indx01.dbf	Off	0	64	64	1	40946
TEMP	20	/oradata/PRACTICE/ temp01.dbf	d:\oradata\PRACTICE\ temp01.dbf	Off	0	64	64	1	4096

任务1：启动数据库配置助手

在Windows NT或Linux下启动数据库配置助手方法如下：

```
LINUX> /oracle/8.1.7/bin/dbassist &
```

```
WINNT> D:\Oracle\8.1.7\bin\launch.exe D:\oracle\8.1.7\assistants\dbca DBAssist.cl
```

在Windows下安装完成后，会在Windows启动菜单上看到这一工具的快捷方式。数据库配置助手通过提问一系列问题来帮助你定制数据库。

任务2：配置数据库配置助手

运行数据库配置助手时，你会遇到许多简单明了的屏幕显示。创建一个定制的（不是典型的）数据库，需要注意以下几点：

- 1) 不要选择任何可选的数据库特性（Oracle的Time Series、Spatial、Jsever和Intermedia等）。这些选项会使SYSTEM表空间增大，这是本书练习不必要的。
- 2) 设置数据库名为大写的PRACTICE。设置SID为大写的PRACTICE。按照8.1.0设置兼容性参数。在你键入数据库名称时，初始化文件位置被填充。
- 3) 按照表3-3，为数据库中每一个表空间设置值。图3-2是设置表空间和参数时的配置屏幕。
- 4) 在/oradata/PRACTICE目录（使用Linux时）或d:\oradata\PRACTICE目录（使用Windows NT时）下创建三个1MB的重做日志文件，分别命名为redo01.log、redo02.log和redo03.log。

任务3：创建数据库

现在，让数据库配置助手在提示符下为你创建数据库。你也可以保存它创建的脚本文件并再次使用或在其他环境中使用。

一旦数据库创建完成，你就拥有了一个名为PRACTICE的Oracle数据库。在Linux或Windows NT下的数据库目录和文件也被创建了。在Windows下安装时，为Oracle数据库添加了服务。查看Windows服务，可以浏览控制面板中为启动和运行PRACTICE所必需的数据库服务。你可以看到，新的服务名为Oracle817PRACTICE。

创建的数据库文件和目录列在表3-4中，管理目录也一同列出。一旦你建立起这个数据库，

就可以连接和浏览了。

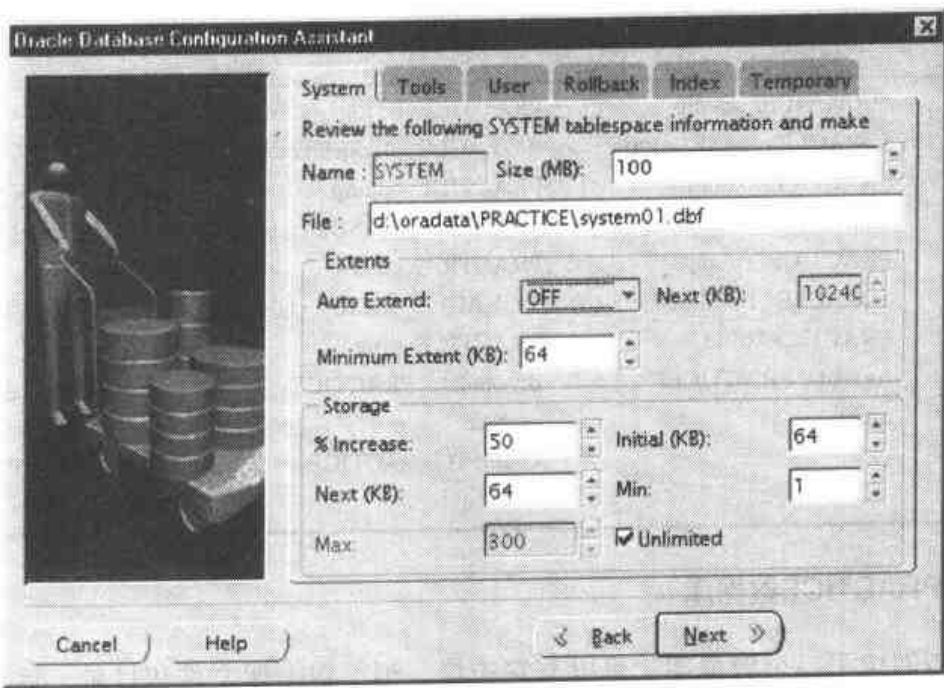


图3-2 在数据库配置助手中配置表空间数据文件

警告 这个数据库的创建并没有采用最优的方式，这些设置也不是创建数据库的最好练习。你仅仅是需要一个数据库来完成备份和恢复练习。因此，在产品级数据库中，分别在磁盘中放置控制文件并且镜像重做日志，而在不同的磁盘上放置重做日志。

表3-4 PRACTICE数据库中的重要文件、目录和服务

目录、文件或 服务名	Linux下的位置	Windows NT下的位置	注 释
数据文件	/oradata/ PRACTICE/	d:\ oradata\ PRACTICE\	PRACTICE数据库的数据文件存放在该目录下
控制文件	/oradata/ PRACTICE/ control01.ctl /oradata/ PRACTICE/ control02.ctl /oradata/ PRACTICE/ control03.ctl	d:\ oradata\ PRACTICE\ control01.ctl d:\ oradata\ PRACTICE\ control02.ctl d:\ oradata\ PRACTICE\ control03.ctl	三个控制文件同数据文件和重做日志存放在同一目录下
重做日志文件	/oradata/ PRACTICE/ redo01. log /oradata/ PRACTICE/ redo02. log /oradata/ PRACTICE/ redo03. log	d:\ oradata\ PRACTICE\ redo01. log d:\ oradata\ PRACTICE\ redo02. log d:\ oradata\ PRACTICE\ redo03. log	三个联机重做日志文件同数据文件和控制文件存放在同一目录下
用户转储目的	\$ORACLE_ BASE/ admin	%ORACLE_ BASE%\ admin\	服务器进程跟踪文件存放在

(续)

目录、文件或 服务名	Linux下的位置	Windows NT下的位置	注 释
文件	/PRACTICE/ udump	PRACTICE\ udump	该目录下
后台转储目的	\$ORACLE_BASE/ admin/	%ORACLE_BASE%\ admin\	所有后台进程和告警日志的
文件	PRACTICE/ bdump	PRACTICE\ bdump	跟踪文件存放在该目录下
核心转储目的	\$ORACLE_BASE/ admin/	%ORACLE_BASE%\ admin\	Oracle转储核心文件存放在
文件	PRACTICE/ cdump	PRACTICE\ cdump	该目录下
参数文件目的	\$ORACLE_BASE/ admin/	%ORACLE_BASE%\ admin\	数据库初始化参数文件在该
地	PRACTICE/ pfile	PRACTICE\ pfile	目录下
归档日志目的	/oradata/ PRACTICE/	d:\ oradata\ PRACTICE\	归档重做日志存储在该目录
地	archive	archive	
数据库服务		Oracle817PRACTICE	启动和运行PRACTICE数据库所必需的WindowsNT服务

练习3.3: 浏览PRACTICE数据库

现在, 已经创建了一个数据库, 可以与它连接, 浏览数据库文件和目录, 启动和停止数据库。你可以把它当作是一个熟悉的练习。完成下列任务, 快速浏览你的新数据库:

任务描述	时间 (分钟)
1. 连接数据库	15
2. 查询数据字典	10
3. 浏览数据库管理文件	10
4. 关闭数据库	5
5. 启动数据库	5
6. 检查DBA Studio	10
总计时间	50

现在完成的某些练习在本书的其他部分还会重复。

任务1: 连接数据库

安装Oracle软件时, 你会装入所有的数据库执行体、支持文件和许多实用程序。SQL*Plus就是这样一个有用的程序。SQL*Plus好比是Oracle数据库的命令提示符, 你可以用它来控制数据库操作、查询或更改数据。在Oracle文档中有SQL*Plus的手册。Oracle9i中, 不再提供服务器管理器实用程序, 所有服务器管理命令都由SQL*Plus支持。本书中使用SQL*Plus进行数据库访问和控制。

注意 在本书中, 当命令在SQL*Plus提示符下键入后, 你会在命令行前看到SQL>提示符。对于操作系统命令, 提示符是WINNT>或LINUX>。

在操作系统命令行下启动SQL*Plus, 并且用两个不同的用户 (SYS和SYSTEM) 来登录到PRACTICE数据库。执行下列命令连接数据库:

```
LINUX> sqlplus /nolog
SQL> connect sys/change_on_install
```

```
Connected.
SQL> connect system/manager
Connected.
```

恭喜！你已经连接到数据库上了。

当用户连接到数据库时，该用户就具有权限和角色。权限是对数据库用户的认可或赋予完成任务的角色。角色是权限集合的命名。大多数对数据库管理操作都可以通过作为SYS和SYSTEM连接来完成。但是，你不能作为SYS或SYSTEM来启动和关闭数据库，除非你作为这些用户担任SYS和SYSTEM角色。试验这些连接，验证SYS和SYSTEM担任SYSDBA角色。

技巧 使用INTERNAL用户是作为SYS担任SYSDBA角色的捷径。在Oracle9i中，不再使用INTERNAL的用户名。

```
SQL> connect sys/change_on_install as sysdba
Connected.
SQL> connect system/manager as sysdba
Connected.
```

SYS用户默认担任SYSORER角色，但SYSTEM不是。在本书中，大多数操作都是由SYS担任SYSDBA的角色来完成的。

到此，你已经使用本地连接（未使用Net8）连接到PRACTICE数据库了。数据库必须运行一个接收过程来确认Net8的连接。用下列连接命令测试Net8到PRACTICE数据库的连接性：

```
SQL> connect sys/change_on_install@practice as sysdba
Connected.
SQL> connect system/manager@practice as sysdba
Connected.
```

连接命令中的@符号指示数据库连接时使用Net8。如果没有指定Net8连接字符，连接命令将连接到由ORACLE_SID环境变量定义的数据库。参考Net8管理员指南，它详细地讲解了Net8的使用和配置。在本书中，所有的连接都是连接到本地数据库服务器。

任务2：查询数据字典

你创建的ORACLE数据库包含一个数据字典。如同英语字典定义单词一样，数据字典定义有关数据库的细节。从数据字典中选择一些信息来证明你创建了前面描述的数据库。运行下面的查询，检查连接上的数据库名称和归档日志模式：

```
SQL> SELECT name, log_mode FROM v$database;
```

运行下面的查询可以查看数据库的版本：

```
SQL> SELECT * FROM v$version;
```

新的PRACTICE数据库有许多动态数据字典视图，可以用来管理备份和恢复操作，表3-5列出了本书中将要用到的一些视图。在SQL*Plus下，用列出的每一个视图查询PRACTICE数据库。如果你想查看数据库的数据文件，你可以按照下面的方法查询数据字典：

```
SQL> SELECT name, bytes FROM v$datafile;
SQL> SELECT file_name, bytes FROM dba_data_files;
```

这两种查询都可以提供数据库数据文件信息。在某些备份和恢复操作中，只使用动态视图。

许多v\$动态视图在数据库控制文件加载而数据库未打开时可用。这是因为数据字典表包含了SYSTEM表空间的信息，而v\$动态视图包含了数据库控制文件中的信息。

表3-5 备份和恢复中使用的动态视图和字典视图

数据库对象或结构	动态视图或字典视图
数据库信息	v\$DATABASE, v\$INSTANCE
数据库参数信息	v\$PARAMETER
表空间信息	v\$TABLESPACE, DBA_TABLESPACES
数据文件信息	v\$DATAFILE, v\$TEMPFILE, v\$DATAFILE_HEADER, DBA_DATA_FILES, DBA_TEMP_FILES
重做日志文件信息	v\$LOG, v\$LOGFILE, v\$LOG_HISTORY
控制文件信息	v\$CONTROLFILE
数据库恢复信息	v\$RECOVERY_FILE_STATUS, v\$RECOVERY_LOG, v\$RECOVERY_PROGRESS, v\$RECOVERY_LOG, v\$RECOVER_FILE

如果键入下列查询，你可以看到PRACTICE数据库联机重做日志文件的位置和大小：

```
SQL> SELECT member, bytes
2 FROM v$logfile lf, v$log l
3 WHERE lf.group# = l.group#;
```

通过查看v\$controlfile视图来确定数据库控制文件的位置，方法如下：

```
SQL> SELECT name FROM v$controlfile;
```

任务2中数据文件、重做日志和控制文件查询结果产生相同的文件，这些文件作为操作系统文件列在/oracle/oradata/PRACTICE/directory目录下。

技巧 动态视图的名称从不使用复数，数据字典名称通常使用复数。

本书后续部分，你将学习怎样使用这些动态视图和字典视图。任务2只是一个基本的查询练习，让你对从数据库中选取数据库相关信息有所体会。

任务3：浏览数据库管理文件

在数据库创建过程中，将一起产生几个并不包含数据库数据的重要文件。找到这些文件，在编辑器中打开它们并查看其中的内容。

- **参数文件** 数据库实例启动时，Oracle将读取数据库参数文件。PRACTICE数据库参数文件是initPRACTICE.ora，位于\$ORACLE_BASE/admin/PRACTICE/pfile目录下。当你在数据库配置管理器中回答问题时，那些回答被记录到这个文件中。如果你的数据库块大小设置为8192字节，在参数文件中你将看到db_block_size被设置为8192。
- **告警日志** 重要的数据库事件和错误被写入告警日志。告警日志位于\$ORACLE_BASE/admin/PRACTICE/bdump目录下。用编辑器打开名为alert_PRACTICE.log或者practiceALRT.log的文件，查看数据库启动、关闭和重做日志切换等记录。
- **口令文件** 口令文件使你可以在另一台机器上管理这一数据库。在Linux下，口令文件为\$ORACLE_HOME/dbs/pwdPRACTICE.ora，而在Windows NT下口令文件为%ORACLE_HOME%\database\PWDPRACTICE.ora。你无法查看这个文件的内容。

任务4：关闭数据库

在SQL*Plus提示符下，使用shutdown命令可以关闭Oracle数据库。shutdown命令中有四个可选项：normal、immediate、transactional、abort，现在练习使用这些可选项。

```
SQL> connect sys/change_on_install as sysdba
Connected.
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

当数据库关闭时，它将信息写入告警日志，然后以干净、稳定的方式关闭数据库文件。

记住，当你关闭一个数据库时，Oracle服务器要做许多工作，包括将SCN写入数据库文件、清除缓冲区中无用的数据块、回退未提交的事务。控制文件也必须随之更新。干净的关闭允许数据库整理所有的内存和文件，你可以确信数据库文件处于一致的状态。如果你按如下方式终止数据库：

```
SQL> Shutdown abort;
```

Oracle不执行清理（cleanup）操作。正因为如此，你会发现shutdown abort完成的比其他shutdown命令快。

任务5：启动数据库

发布startup命令来启动Oracle数据库：

```
SQL> connect sys/change_on_install as sysdba
Connected.
SQL> startup
ORACLE instance started.
Total System Global Area      30484508 bytes
Fixed Size                     75804 bytes
Variable Size                  13553664 bytes
Database Buffers               16777216 bytes
Redo Buffers                    77824 bytes
Database mounted.
Database opened.
```

在数据库启动过程中，读取初始化文件、分配内存结构并启动后台进程。数据库文件被加载，打开数据库。数据库启动时要经历三个阶段，你可以从命令的输出中看到：NOMOUNT、MOUNT和OPEN。你可以在某时刻处于打开数据库的一个阶段，尝试如下分阶段启动PRACTICE数据库：

```
SQL> startup nomount
ORACLE instance started.
...
SQL> alter database mount;
Database altered.
SQL> alter database open;
Database altered.
```


任务6：研究DBA Studio

Oracle提供了一个图形用户界面的工具，在管理数据库时十分好用。Oracle企业管理器的DBA Studio集中了多种数据库管理工具，使用DBA Studio可以管理以下内容：

- 实例：包括启动、关闭和数据库参数
- 模式：包括表、索引和视图
- 安全：包括数据库用户、职责和权限
- 存储：包括表空间、数据文件、回退段、重做日志、控制文件和归档日志。

DBA Studio在Linux和Windows NT下的工作方式完全一致。你可以用如下方式在操作系统命令提示符下启动这个工具：

```
LINUX> oemapp dbastudio
```

```
WINNT> oemapp dbastudio
```

在Windows下安装时，Oracle在启动菜单中为DBA Studio创建一个菜单选项。

打开并作为SYS用户连接到PRACTICE数据库。如果有问题，参考DBA Studio的联机帮助。

图3-3展示了DBA Studio导航屏幕显示。打开PRACTICE看看，浏览数据库以下四个主要部分：

- 存储 (Storage)：扩展导航树的存储节点。浏览你的控制文件、表空间、数据文件、回退段、重做日志和归档日志。
- 实例 (Instance)：在实例节点上检查数据库和所有的初始化参数。
- 模式 (Schema)：打开模式节点，检查你能否找到拥有的v\$数据文件视图。
- 安全 (Security)：导航到安全节点，检查SYSTEM被赋予了何种角色。

你会发现DBA Studio为你查看和管理数据库提供了一个方便的手段。许多动态视图和字典视图中的信息在这个工具中以图形方式显示。虽然你可以使用这个工具完成许多管理事务，但本书只介绍在命令提示符下完成备份与恢复的操作。

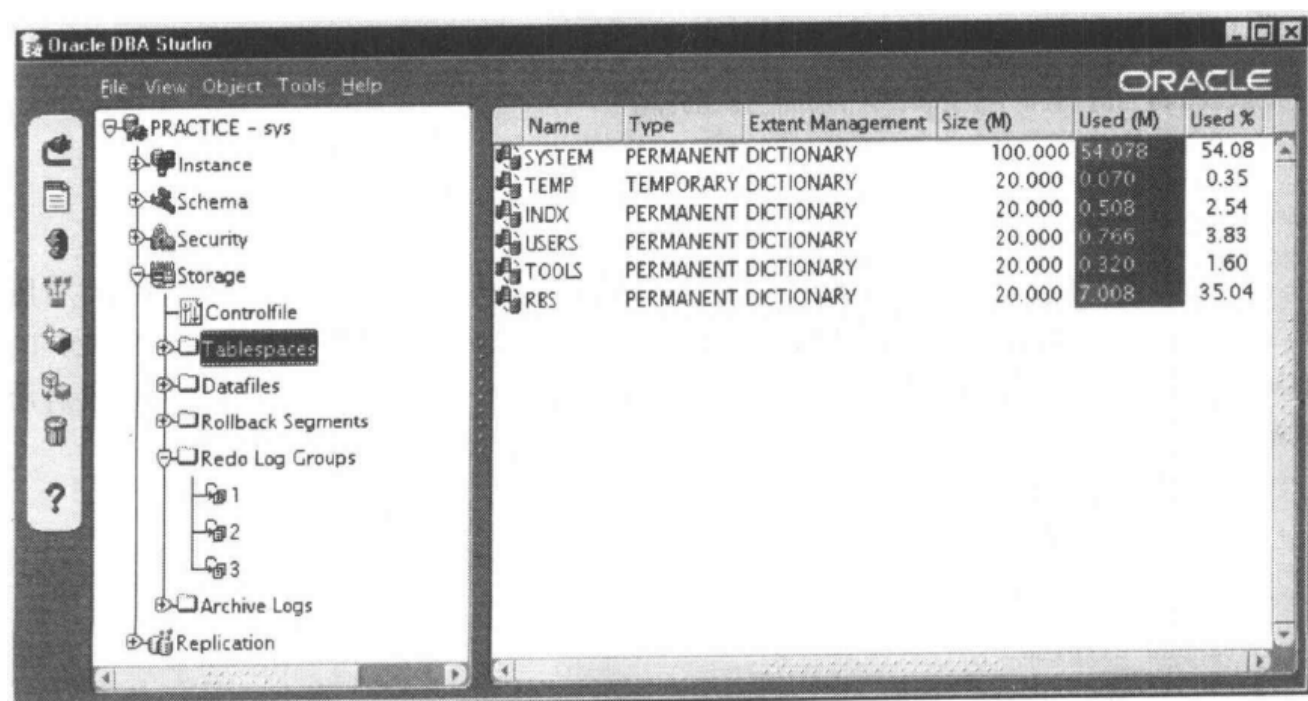


图3-3 使用DBA Studio管理PRACTICE数据库

练习3.4：修改PRACTICE数据库

需要修改PRACTICE数据库以便在本书的练习中使用它。完成下列任务为后续章节准备数据库。

任务描述	时间（分钟）
1. 更改口令	10
2. 添加数据文件	10
3. 创建SCOTT用户	10
4. 创建TINA用户	10
5. 创建TINA的对象	10
6. 产生数据库行为	15
总计时间	65

这一练习通过创建结构、增添用户和产生数据来准备PRACTICE数据库。

任务1：更改口令

数据库使用默认口令自动创建两个用户，分别是SYS和SYSTEM。SYS的初始口令为change_on_install，SYSTEM的初始口令为manager。因为安全原因在实际数据库中默认口令需要更改。使用ALTER USER命令为这些用户更改口令：

```
SQL> connect sys/change_on_install
Connected.
SQL> alter user sys identified by practice;
User altered.
SQL> alter user system identified by practice;
User altered.
```

非常好。我已经十分厌倦了键入change_on_install，你呢？你当然可以根据自己的愿望更改这些口令。

任务2：添加数据文件

使用10MB的数据文件而不像其他非系统表空间使用20MB数据文件来创建表空间，原因是即将开始的备份和恢复练习需要一个包含两个数据文件的表空间。使用ALTER TABLESPACE命令为用户表空间创建第二个数据文件：

```
SQL> ALTER TABLESPACE users ADD DATAFILE
'/oradata/PRACTICE/user02.dbf' SIZE 10M;
Tablespace altered.
```

现在，用户USERS表空间为20MB并且包含两个数据文件。

任务3：创建SCOTT用户

SCOTT用户有四个表。你可以运行\$ORACLE_HOME/rdbms/admin目录下的脚本文件utlsampl.sql来创建用户SCOTT。在SQL*Plus下使用符号@或者start命令运行脚本文件。

```
SQL> @/app/oracle/product/8.1.7/rdbms/admin/utlsampl
```

这一用户在后面章节中被用于逻辑备份和恢复练习。

任务4：创建TINA用户

还需要创建另一用户TINA来阐述本书中的表空间时间点恢复。TINA拥有一个表，该表包含了用来跟踪一段时间内行为的若干记录。可以创建和安排一项Oracle任务来为这一数据表填充数据。完成了数据库恢复之后，我们可以检查该表来帮助证实恢复的结果。按如下方法创建TINA并赋予它权限和作用：

```
SQL> GRANT CONNECT, RESOURCE, UNLIMITED TABLESPACE TO tina
IDENTIFIED BY panda;
SQL> ALTER USER tina DEFAULT TABLESPACE tools;
SQL> ALTER USER tina TEMPORARY TABLESPACE temp;
```

任务5：创建TINA的对象

PRACTICE数据库中，TINA的惟一作用是跟踪时间。TINA拥有一个表、一个过程和一项任务，其中数据表命名为DATE_LOG，过程命名为CREATE_DATE_LOG_ROW，它的任务将在“任务6”中添加。

```
SQL> CONNECT tina/PANDA
SQL> DROP TABLE DATE_LOG;
SQL> CREATE TABLE DATE_LOG
    (create_date DATE CONSTRAINT create_date_pk PRIMARY KEY);
SQL> CREATE OR REPLACE PROCEDURE create_date_log_row
IS
-- Purpose: Insert a row with the current date/time into DATE_LOG.
BEGIN
    INSERT INTO date_log (create_date) VALUES (SYSDATE);
END;
/
```

这段代码创建了一个表和一个过程。在PRACTICE数据库中运行这些命令。

任务6：产生数据库行为

在备份和恢复练习的过程中，你需要一些正在进行的数据库行为来检验你的恢复操作是否已经正确完成。通过在TINA.DATE_LOG中插入行来产生数据库行为。可以使用插入声明、过程调用或数据库工作来向表中插入行。

使用如下简单的INSERT命令向表中插入当前日期和时间：

```
SQL> INSERT INTO tina.date_log VALUES (SYSDATE);
```

也可以使用你刚创建的过程来完成相同的插入操作，方法如下：

```
SQL> execute tina.create_date_log_row;
```

每种方法都是向表中插入新的一行记录。

现在，你需要创建一项任务，它不断地向TINA.DATE_LOG表中插入行。使用下列命令创建和安排任务：

```
SQL> VARIABLE jobno number;
SQL> BEGIN
-- Run the job every 10 minutes
    DBMS_JOB.SUBMIT(:jobno, 'create_date_log_row;', SYSDATE, '{SYSDATE +
```

```
1/(24*60))');
    commit;
END;
/
SQL> print jobno
```

通过提交任务自动进行插入行操作，你只需创建Oracle任务调度程序。任务调度程序按照任务定义的有规则的时间间隔执行任务。任务调度程序在数据库启动时开启，在数据库打开时动态工作。在PRACTICE数据库中设置两个参数以便任务调度程序自动启动。

```
job_queue_processes = 2
job_queue_interval = 30
```

参数job_queue_processes启动两个Oracle调度过程。参数job_queue_interval指出了任务调度程序每30秒唤醒一次来确定是否要执行任务。

数据库打开时启动任务调度程序，使用ALTER SYSTEM命令：

```
SQL> ALTER SYSTEM SET job_queue_processes = 2;
```

这样就启动了两个后台过程来运行确定时间的数据库任务。一旦数据库打开就不能更改任务间隔时间，默认的时间间隔是60秒。

尝试通过数据库参数文件或动态命令启动任务过程。几分钟后，你就可以从TINA.DATE_LOG中选择数据。你将看到通过提交的任务表中已经被插入了行。

```
SQL> SELECT * FROM TINA.DATE_LOG;
```

你也许决定要更改数据库任务运行的时间间隔。你可以作为用户TINA通过查询USER_JOBS表找到任务编号方法如下：

```
SQL> SELECT job, what FROM USER_JOBS;
```

然后你可以使用这一过程和一些日期函数改变时间间隔，任务编号是1：

```
SQL> execute DBMS_JOB.INTERVAL(1, null, null, 'sysdate+1'); -- Every Day
SQL> execute DBMS_JOB.INTERVAL(1, null, null, 'sysdate+1/24'); -- Every Hour
SQL> execute DBMS_JOB.INTERVAL(1, null, null, 'sysdate+1/(24*60)'); -- Every
Minute
```

你可以利用中断来停止任务。如上所述，你需要检索任务编号。然后执行下列命令来终止任务（这里我使用了1234作为任务编号）：

```
SQL> execute DBMS_JOB.BROKEN(1234, TRUE);
```

Oracle不再运行该项任务，直到你使用下面的代码改变它的中断状态：

```
SQL> execute DBMS_JOB.BROKEN(1234, FALSE, SYSDATE);
```

任务继续运行并且返回到最初创建任务时定义的时间间隔。

用户TINA现在拥有一个表、一个过程和一个任务，用来调用过程按照你选定的时间间隔向DATE_LOG表中插入记录。这次练习，就是在参数文件中设置参数，让任务调度程序完成其他工作。

创建RCAT数据库

用于备份和恢复的工具之一是恢复管理器。恢复管理器目录存储在一个Oracle数据库中，其

中存储了供RMAN使用的重要的备份信息。为恢复管理器使用的数据库是RCAT (Recovery Catalog的缩写)。创建PRACTICE数据库的同时创建名为RCAT的数据库。使用自己编写的脚本或使用数据库配置助手创建RCAT。对于本书讲述的内容来说, RCAT数据库文件存放的位置无关紧要。

使用数据库配置助手来创建RCAT数据库时, 拷贝现存的数据库文件。这样做比创建并运行所有的数据库脚本要快。完成后, 计算机上包含了数据库文件、目录和RCAT数据库的服务程序。这些文件最好放在RCAT目录下。记住要更改SYS和SYSTEM的口令。

3.5 疑难解答

在完成本章练习过程中, 你也许会遇到一些不期望的结果或错误。浏览下面关于疑难解答的清单, 如果在本章或其他章节中还需要更进一步的帮助, 请在<http://metalink.oracle.com>上查找。注册后, 你可以找到大量的疑难解答和建议。

3.5.1 Connect命令

在本章的连接示例中, 你也许会遇到以下这些问题:

- 1) File not found。如果你键入SQL*Plus而操作系统没有找到该程序, 将会遇到这个提示。检查PATH环境变量的设置是否正确。
- 2) ORA-01031: insufficient privileges。也许是你连接了一个数据库, 该数据库未经授权便使用了SYSDBA或SYSOPER。以具有此权限的其他用户登录或给这个失败的用户授权。
- 3) ORA-01034: ORACLE not available。当你在SQL*Plus使用本地连接(未使用Net8)时, 数据库必须启动连接。以SYSDBA身份连接并启动数据库, 同时确保ORACLE_SID设置正确。
- 4) ORA-12500(TNS-12500)TNS: listener failed to start a dedicated server process。当你在SQL*Plus通过Net8连接时, 数据库服务程序、数据库启动连接开始工作。确保这些组件当前正在运行并重新连接。
- 5) ORA-12560(TNS-12560)TNS: protocol adapter error。因为实例的退出而使Oracle连接无效。退出SQL*Plus重新启动连接。Windows NT下, 数据库服务程序可能没有运行。
- 6) ORA-24314: service handle not initialized。未使用Net8连接数据库并且没有设置ORACLE_SID变量。改变ORACLE_SID的值重新连接。
- 7) ORA-24323 value not allowed。因为实例的退出而使Oracle连接无效。退出SQL*Plus重新启动数据库连接。

3.5.2 Select命令

当你从数据字典中选择数据时, 也许会发生以下这些错误:

- ORA-00942 table; or view does not exist。选择数据库视图时, 可能遇到这一提示信息。如果不是键入错误, 那么视图存在但连接的用户可能没有访问权限。设置正确的权限来选择用户。你还要在数据表名称前注明数据表的所有者。

- ORA-01219: database not open: queries allowed on fixed tables/views only。也许是你连接到一个已加载但没打开的数据库，试图从不能访问控制文件的视图或表中获取数据。打开数据库或者从v\$动态视图中选择数据。

3.5.3 Shutdown命令

关闭数据库时，你会看到一些提示，列举如下：

- Shutdown hangs for seconds。要等待几秒钟，Oracle软件需要完成一些清理工作。
- Shutdown hangs for minutes。如果你正常地发布了关机命令而尚有用户和数据库相连，关机命令一直等到该用户断开连接后执行。你可以通过SQL*Plus多次连接到你自己的数据库来练习用户连接状态的关机操作。打开其他的SQL*Plus对话（从其他命令提示符）并关闭数据库。正常的关机要等到最初的连接断开后才进行。
- ORA-01109: database not open。数据库关闭时，当然未被打开，因此不必理会这条消息。
- ORA-01507 database not mounted。数据库关闭时，当然未被加载，因此不必理会这条消息。

3.6 小结

到目前为止，你已经为练习数据库恢复操作选择了一台计算机，并且在机器上安装了Linux或Windows NT以及Oracle 8.1.7企业版，你还创建了两个数据库，学习了如何在数据字典中查看数据。完成了这些基础工作，你就可以完成本书所有章节中的每一个练习了。

在本书随后的章节中，我将逐步指导你完成备份与恢复操作。如果你在家中跟随我们一起完成了这些数据库操作，将比单纯看书学习更多的知识。把本书作为一个实践的训练工具，将促进你对Oracle备份与恢复的理解和使用。

在本章中，我简要讲述了Oracle的安装和数据库的创建。要了解更多在Linux和Windows NT上安装Oracle的内容，请参考《Oracle8i Installation Guide for Linux》或者《Oracle8i Installation Guide for Windows NT》。你可以在docs.oracle.com在线找到这些文档或在安装盘上找到。参考《Oracle8i Administrator's Guide for Windows NT》或《Oracle8i Administrator's Guide for Linux》，将学习更多使用数据库配置助手创建数据库的内容。如果你想学习更多的有关工作系统组织的内容，可以参考《Oracle8i Supplied PL/SQL Packages Reference manual》中的第17章（DBMS_JOB）。

习题

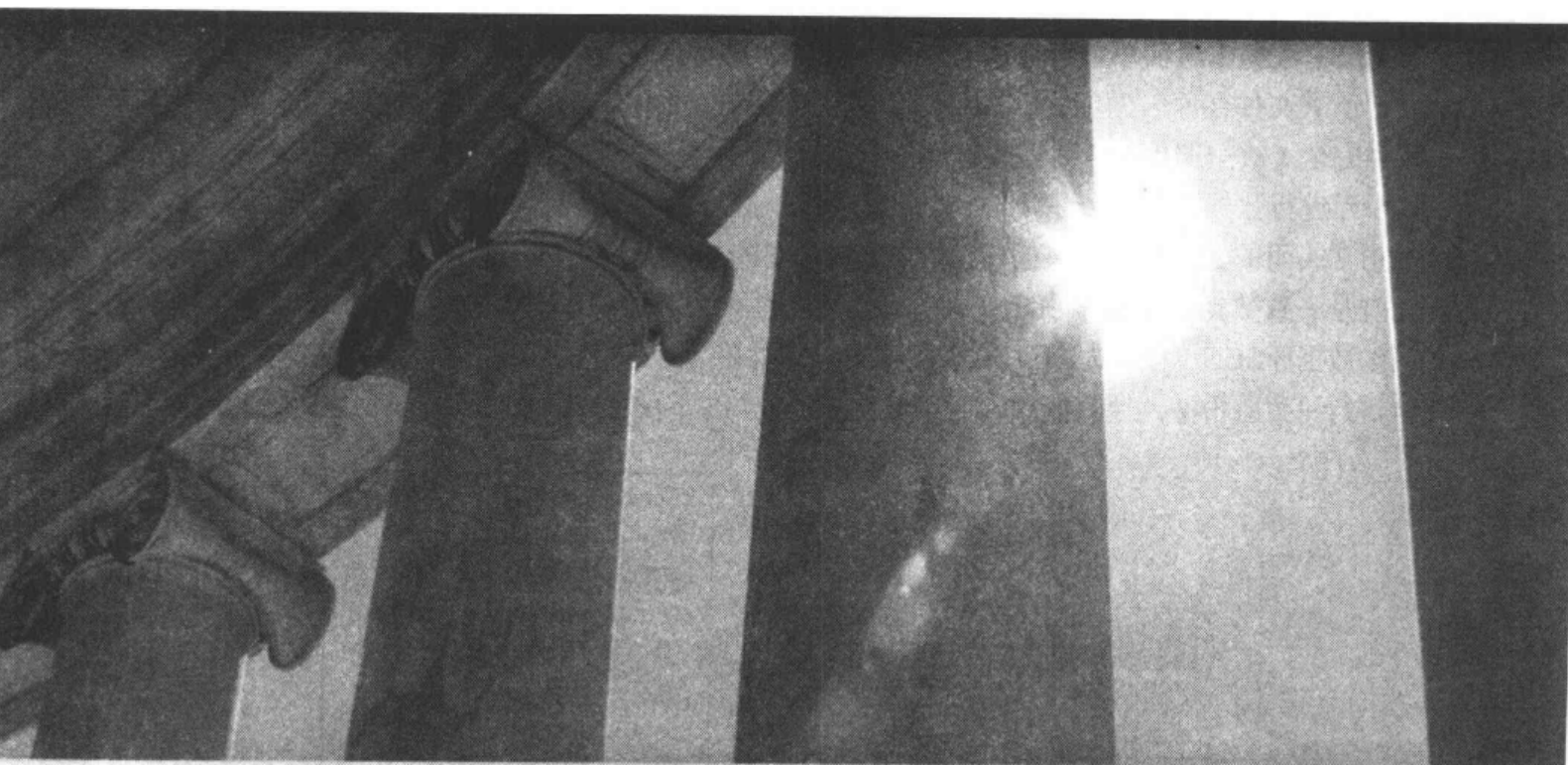
回答下列问题，复习本章中的重要概念。

1. 使用环境变量ORACLE_SID目的是什么？
 - A. 定义Oracle软件所在的位置
 - B. 定义Oracle数据库和管理文件的位置
 - C. 定义当前OS对话中的数据库实例名
 - D. 定义Oracle安装的序列号

2. 使用哪种工具创建和删除一个Oracle数据库?
 - A. Oracle Universal Installer
 - B. Oracle Database Configuration Assistant
 - C. Oracle DBA Studio
 - D. 活动扳手
3. 动态视图和数据字典视图的区别是什么? (选出所有正确答案)
 - A. 大多数动态视图在数据库加载而未打开时使用; 字典视图不是这样
 - B. 动态视图显示当前控制文件或内存结构中的信息; 字典视图显示SYSTEM表空间的信息
 - C. 所有的动态视图都以字符v\$开头; 大多数字典视图以DBA_、ALL_或USER_开头
 - D. 动态视图有一个单数名; 大多数字典视图有一个复数名
4. 使用什么命令启动数据库实例、加载控制文件却并不打开数据库为用户访问?
 - A. STARTUP
 - B. STARTUP NOMOUNT
 - C. ALTER DATABASE MOUNT
 - D. STARTUP MOUNT
5. 使用何种视图来查询数据库中当前的数据文件?
 - A. v\$LOG, v\$LOG_HISTORY
 - B. v\$CONTROLFILE
 - C. v\$DATAFILE, v\$TEMPFILE, DBA_DATA_FILES
 - D. v\$PARAMETER

答案

1. C。ORACLE_SID指示了与数据库相联系的Oracle执行程序。
2. B。使用Oracle Database Configuration Assistant创建Oracle数据库, 没有数据库配置助手这个工具时, 你也可以用SQL命令创建数据库。
3. A、B、C、D。动态视图和字典视图提供对数据库中相似信息的访问。
4. D。STARTUP MOUNT启动实例并加载数据库; 单独的STARTUP命令打开一个数据库; NOMOUNT选项指示不加载控制文件; ALTER DATABASE MOUNT仅在实例被加载时有效。
5. C。视图v\$DATAFILE、V\$TEMPFILE、DBA_DATA_FILES反映了数据库当前的数据文件。



第二部分 用户管理的备份与恢复

第4章 关闭数据库的备份与恢复

保护数据库最简单的方法就是把所有的数据库文件复制到另一个地方。一旦出现故障，可以把所有文件在原来的位置上恢复，然后启动数据库，又能够进行正常工作了。这种操作叫做完全（或全部）一致数据库备份与恢复。

当进行数据库还原或恢复时，使用的是在一致或非一致数据库上生成的备份。一致数据库是正常关闭的，并包含经过同步的文件。图4-1中给出的数据文件、重做日志以及控制文件说明此时是一致数据库状态。由于在关机之前数据库是正常关闭的，因此所有这些文件都被标以相同的数据库SCN（被称为Stop SCN）。作为一组文件，这些文件代表了一个一致数据库。本章中，将进行一致备份并还原与恢复PRACTICE数据库。在下一章中，我将讲解非一致备份与恢复。

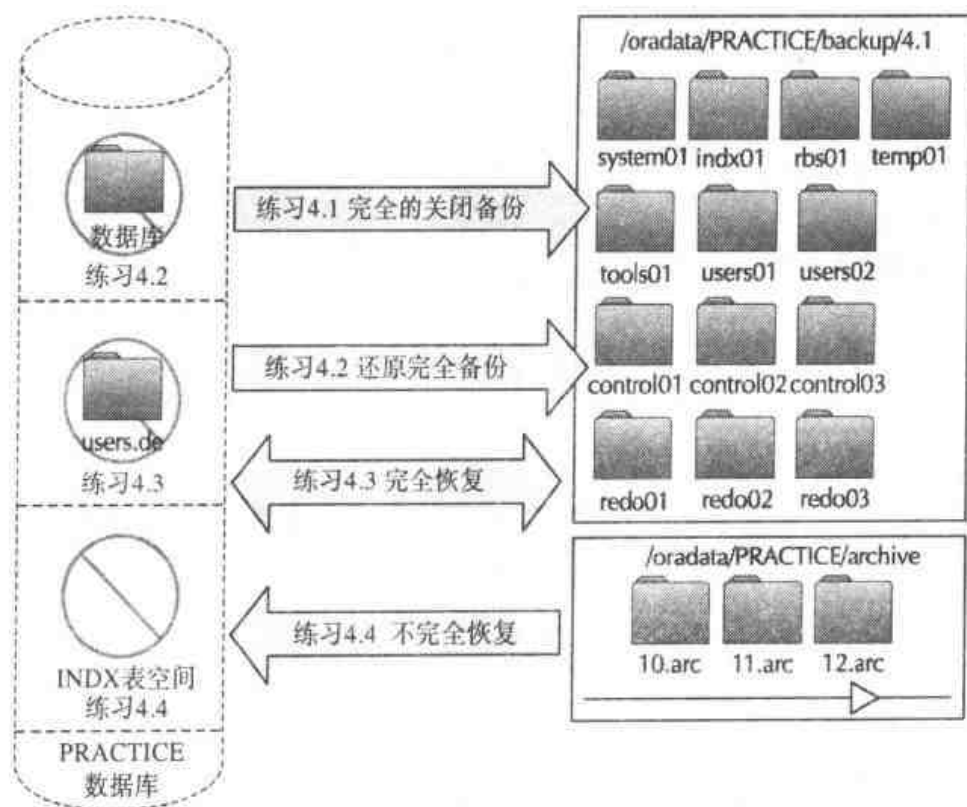


图4-1 关闭备份、还原和恢复操作

注意 一致数据库备份通常是指冷备份。此术语起源于数据库文件并不是处于激活或变动之中的概念。当数据库关闭时，数据库文件是被冻结的，是冷的。

本章中，将要实践多种完全的物理恢复。第一种情况是完全的关闭备份和还原，但不恢复；第二种情况是完全的关闭备份和恢复；第三种情况是不完全恢复。本章中所有的操作都在PRACTICE数据库上进行。还需要配置PRACTICE数据库以便使用归档日志。

4.1 关闭数据库的备份与完全还原

拷贝数据库中的所有文件来进行一次完全备份。在拷贝所有的数据文件、控制文件和重做

日志文件之前，先关闭数据库。已关闭的数据库（正常关闭）将包含一致的文件。记住：当数据库打开时，事务被写入内存中，数据文件、控制文件和重做日志文件得到更新。如果在数据库打开时对文件进行物理拷贝，得到的复制文件之间将会不一致，而且也不可能借助这些文件来还原数据库。数据库在打开状态时是可以对其进行备份的，但需要在表空间级进行，这将在第5章中予以讨论。

警告 如果错误地在产品级重做日志文件上还原重做日志文件，冷备份时拷贝重做日志文件就会引发问题。这一错误将使完全恢复数据库的希望变得渺茫。因此，许多DBA选择在备份时不复制联机重做日志。如果数据库在备份前已经正常关闭，那么联机重做日志对于完全恢复数据库就不是必须的了。

练习4.1：备份关闭的数据库

本练习中，将要备份PRACTICE数据库中所有的文件。接下来，模拟出现严重的计算机故障，删除所有的数据库文件，然后从数据库备份上还原所有的数据库文件，并打开数据库。

任务描述	时间（分钟）
1.生成并启动数据库	5
2.创建备份脚本	10
3.运行备份脚本	10
总计时间	25

本练习是最简单的一种数据库备份形式：把数据库文件完全拷贝到一个备份场所。

任务1：生成并启动数据库

在本书的许多练习中，都希望改变数据库，还希望确认所进行的操作产生了预期的效果。为了达到这两个目的，可以如第3章中所讨论的那样，向Tina的DATE_LOG表中插入行。查看是否已经在表中生成了新的行。最后，记录TINA.DATE_LOG表中添加的最后一行。

```
SQL> SELECT TO_DATE(max(create_date),'HH24:MI:SS') latest_time
        FROM tina.date_log;
```

当还原数据库时，可以检查最大时间，并将它与查询到的时间进行比较。

任务2：创建备份脚本

脚本是执行数据库任务的一种有用方法，可以使用如下所示的脚本来备份数据库中的所有文件。在本书中，我将提供类似这个脚本的一些基本脚本，你可以学习这些脚本，并在它们的基础之上构建自己的脚本。下面这一脚本将运行于Linux数据库上。修改拷贝命令和路径名后，该脚本就可以在Windows系统的数据库上使用了。

```
1. Remark Set SQL*Plus variables to manipulate output
2. set feedback off heading off verify off trimspool off
3. set pagesize 0 linesize 200
4. Remark Set SQL*Plus user variables used in this script
5. define dir = '/oradata/PRACTICE/backup/ch04'
6. define fil = '/tmp/closed_backup_commands.sql'
7. prompt *** Spooling to &fil
```

```
8. Remark Create a command file with file backup commands
9. spool &fil
10. select 'host cp '|| name ||' &dir' from v$datafile order by 1;
11. select 'host cp '|| member ||' &dir' from v$logfile order by 1;
12. select 'host cp '|| name ||' &dir' from v$controlfile order by 1;
13. select 'host cp '|| name ||' &dir' from v$tempfile order by 1;
14. spool off;
15. Remark Shutdown the database cleanly
16. shutdown immediate;
17. Remark Run the copy file commands from the operating system
18. @fil
19. Remark Start the database again
20. startup;
```

首先可以从该脚本中看到运行一个完全备份数据库中所有文件所需的命令。

- 第2-3行设置 SQL*Plus 变量，避免从数据库中提取的结果在命令文件中显示不必要的内容。在这些行中，可以设置系统变量以控制当前 SQL*Plus 会话的作用。
- 第5-6行在脚本范围内为命令指定用户变量。用户变量可以省去键盘输入。定义一个名为 dir 的文本字符串变量。在接下来的脚本中，可以通过在变量名前面加上一个 “&” 来引用该变量值。该脚本中 dir 的值指定了备份文件将被拷贝到的路径位置，而 fil 则指定了包含备份命令的文件名称。
- 第7行示范了如何使用 SQL*Plus 提示命令显示输出结果。
- 第9行通知 SQL*Plus 将所有的屏幕输出结果写入一个文件。文件名称包含在变量 fil 的值中。在此脚本中，后续行的输出结果被写到文件/tmp/closed_backup_commands.sql 中。
- 第10-13行给出了多个 OS 命令，执行对 PRACTICE 数据库中每个数据文件的文件拷贝。这些行从数据字典中提取信息，创建 OS 命令。符号 “||” 用于串接 SQL 命令中的字符串。使用 V\$ 动态视图复制各个数据文件、联机重做日志文件、控制文件以及临时文件。“host” 一词位于每个复制命令之前。在 SQL*Plus 提示符下，可以运行 OS 命令，只需在行首以单词 “host” 开始（或者在 Linux 系统上用符号 “!”）。PRACTICE 数据库没有任何临时文件，我在这里写上第13行只是为了体现完整性。
- 第14行停止向文件/tmp/closed_backup_commands.sql 的写操作。
- 第16行正常关闭数据库。
- 第18行开始执行前面创建的所有命令。这一步将耗费大部分时间，因为正在拷贝数据库文件。如果在运行了 closed_copy.sql 之后打开 closed_backup_commands.sql 文件，将看到拷贝命令后面跟着关键字 “host”。确信已检查了整个命令文件脚本中的这些命令。
- 第20行再次启动数据库。

为了运行这个脚本，需要以 SYSDBA 的角色 SYS 或 SYSTEM 连接数据库。

技巧 从数据字典中选择信息来学习创建脚本。这不仅能节省键入文字的时间，还能确保脚本在其运行时的正确性。从数据字典中选择信息保证了命令将显示出所有你需要的输出结果。

任务3：运行备份脚本

刚才给出的脚本可以不做变动就运行，或分片运行。可以创建一个文件包含这些脚本命令，然后从SQL*Plus执行该文件。如果创建了一个名为closed_backup.sql的文件，则可以用start命令或@符号来运行该文件，列举如下：

```
SQL> @closed_backup.sql
```

或

```
SQL> start closed_backup.sql
```

警告 本书中的脚本出于练习目的设计得比较简单，不是为产品级数据库使用而准备的。当在产品机器上进行备份时，还应添加错误处理、通告、验证以及其他必要的部分。

练习4.2：还原整个数据库

本练习中，将要还原上一练习中拷贝的PRACTICE数据库中的所有文件。一旦文件还原了，不需恢复就能打开数据库，因为在上一练习中备份时这些文件是一致的。（PRACTICE数据库不再是运行于归档日志模式，因此，在还原后使用已归档的日志文件进行恢复是不可能的。）

任务描述	时间（分钟）
1.删除数据库文件	5
2.运行还原脚本	10
3.打开数据库	5
4.确认数据库还原	5
总计时间	25

如果机器出现重大故障，可能会丢失所有的数据库文件。典型的解决方法是将备份复制拷贝到磁带上，安全存储。当还原整个数据库时，将所有备份文件从备份磁带上以组的形式拷贝到最初位置。数据库文件的位置可以改变，在本书后面将会看到。但现在，我们以最简单的方式操作。

任务1：删除数据库文件

假设一名新的系统管理员在数据库处于关闭状态时，意外删除了数据库的所有文件。为了模拟这种事故，关闭数据库，并按如下所示删除PRACTICE数据库中的所有数据库文件：

```
rm /oradata/PRACTICE/*
```

在没有数据库文件的情况下，尝试打开数据库时，启动命令将会失败。试试看会得到什么样的提示信息。

任务2：运行还原脚本

在PRACTICE数据库上的还原操作非常简单，因为所有的备份文件都位于同一目录下。你可以按如下方法还原所有的数据库文件：

```
cp /oradata/PRACTICE/backup/ch04/* /oradata/PRACTICE
```

只需要把备份文件从备份目的地拷贝到初始位置。在真实系统中，文件常常是从磁带上还原到许多不同的文件系统上。

任务3：打开数据库

在打开这个还原的数据库之前，先加载控制文件，并熟悉一些重要的动态视图。本任务中接下来的讨论并非是数据库还原必需的部分，但它将有助于你熟悉重要的数据库还原和恢复信息。在打开数据库之前，首先加载数据库如下：

```
SQL> startup mount
```

在OS提示符下，用编辑器打开告警日志，找到练习4.1中的备份进行之前数据库关闭的时间。（在本书的练习中，将常常同时使用多个OS命令提示符或SQL会话。）例如：

```
ALTER DATABASE CLOSE NORMAL
Tue Jan 1 11:40:35 2002
SMON: disabling tx recovery
SMON: disabling cache recovery
Tue Jan 1 11:40:37 2002
Thread 1 closed at log sequence 212
Tue Jan 1 11:40:37 2002
Completed: ALTER DATABASE CLOSE NORMAL
```

列出的告警日志给出了数据库关闭时的时间和日志顺序。对于已加载的数据库，查看v\$datafile视图中的上次更改（last change）和检查点更改（checkpoint change）列。在正常关闭情况下，Oracle服务器执行一个检查点，并更新所有的联机数据文件首部。下面检查一下最后检查点的时间是否与备份前的关闭时间一致：

```
SQL> alter session set NLS_DATE_FORMAT='HH24:MI:SS';
SQL> SELECT file#,
2         status,
3         checkpoint_change#,
4         checkpoint_time,
5         last_change#,
6         last_time
7 FROM v$datafile;
```

数据库关闭的时间正是数据文件检查点发生的时间：

FILE#	STATUS	CHECKPOINT_CHANGE#	CHECKPOI	LAST_CHANGE#	LASTTIME
1	SYSTEM	43857	11:40:36	43857	11:40:36
2	ONLINE	43857	11:40:36	43857	11:40:36
3	ONLINE	43857	11:40:36	43857	11:40:36
4	ONLINE	43857	11:40:36	43857	11:40:36
5	ONLINE	43857	11:40:36	43857	11:40:36
6	ONLINE	43857	11:40:36	43857	11:40:36

确信数据文件的更改时间与数据库的关闭时间吻合，检查点和上次更改的值是数据库关闭时数据文件的SCN值。

当数据文件正在经历一个检查点时，将在该文件上标记一个SCN值。事实上，SCN被写入数据文件的首部和控制文件中。比较数据文件检查点的更改值和联机重做日志的第一个更改值。数据文件中的更改值将比我们例子中当前重做日志的第一个更改值大。

```
SQL> SELECT group#, sequence#, status, first_change#, first_time
2 FROM v$log
3 ORDER BY first_change#;
```

```
GROUP# SEQUENCE# STATUS FIRST_CHANGE# FIRST_TI
-----
3      210 INACTIVE      43372 11:32:30
1      211 INACTIVE      43470 11:32:32
2      212 CURRENT       43538 11:32:35
```

上述输出中，数据文件检查点的SCN是43857，当前重做日志中的第一个更改号是43538。因此，数据库备份发生在重做日志的第2组为当前状态时。

注意 当从视图V\$中提取信息时，常常扫描已设置的控制文件的内容，看到的并不是数据文件或重做日志文件的内容，而是控制文件认为数据文件和重做日志所包含的那些内容。

在扫描了已加载的控制文件的这些视图之后，可以使用ALTER DATABASE OPEN命令打开数据库。在命令执行时，Oracle服务器比较数据文件首部和控制文件中的SCN。如果所有的数据一致，数据库就可以打开了。

任务4：确认数据库还原

当数据库打开时，Oracle自动确认数据库文件是一致的。除非数据文件、控制文件和联机重做日志是一致的，否则不会打开Oracle数据库。为了测试数据库中的数据就是备份时的数据，可以从TINA.DATE_LOG中提取最大的date/time值。这在上一练习中已经示范过了。这个时间应当就是数据库备份时的时间。

本练习举例说明了对全部数据库文件进行冷备份和完全还原。下一练习将对数据库进行归档，并做冷备份，然后进行完全恢复。

技巧 把/oradata/PRACTICE/backup/ch04目录中所有文件的内容拷贝成一个文件，并保存在其他地方。可以把这个拷贝的文件作为数据库的一个基础备份，以防将来数据库备份和恢复时出问题。

4.2 完全与不完全恢复

如果希望能够把数据库还原成它被备份时的那样，利用备份和还原工作就可实现这一点。但是，数据库还原自身存在很大的弊端：丢失了数据库备份之间出现的事务。如果每晚备份数据库，而且数据库在午餐时出现故障，那只能将数据库还原到它出故障前一夜的状态，然后只能告知用户，他们得重新做一遍那天上午完成过的工作。

用户可不愿意重头再做他们已完成的工作！

Oracle恢复使你能够对一个已还原的数据库进行重做。数据库比用户或必用的数据库应用程序更擅长重做这些工作。重做这些工作所必需的信息全部都包含在重做日志文件之中。

练习4.3：完全数据库恢复

本练习中，将要删除一个数据文件，然后恢复该数据文件，使它与数据库的其他部分保持

一致。在此之前，必须将数据库设置于归档日志模式，这样才能保留所产生的重做信息。

任务描述	时间（分钟）
1. 配置数据库归档	15
2. 运行备份脚本	10
3. 提前重做日志	10
4. 删除一个数据文件	5
5. 还原丢失的数据文件	5
6. 恢复还原的数据文件	10
7. 确认数据库恢复	5
总计时间	60

就像其他练习一样，本练习也是以前面的数据库为基础。如果继续进行下去的话，可能需要参考前面的练习。

任务1：配置数据库归档

你可以指示Oracle将联机重做日志文件拷贝到一个或多个脱机的目的地。把联机重做日志文件拷贝到归档重做日志文件中的过程叫做归档。为了恢复数据库，必须启用归档。只有在数据库处于“归档日志”（ARCHIVELOG）模式时归档才会发生。当数据库处于“归档日志”模式时，可能会希望归档自动进行，这样就不必担心自己管理归档的过程。任务1包括指示Oracle配置归档参数，将数据库设置为归档模式。

首先，查看v\$database视图，确定PRACTICE数据库是否处于“归档日志”模式，命令如下：

```
SQL> select dbid, name, log_mode from v$database;
      DBID NAME          LOG_MODE
-----
2629144163 PRACTICE    NOARCHIVELOG
```

数据库标识符（dbid）是数据库创建时由Oracle分配的一个惟一编号。上面给出的查询结果说明PRACTICE数据库并没有处于“归档日志”模式，因此，每当数据库切换到下一个重做日志以记录数据库变化时，Oracle将覆盖前一个重做日志中的事务。联机重做文件中的重做变化信息一旦被覆盖，就不能再使用了。

在将数据库设置成“归档日志”模式之前，可能会希望通过参数文件中的数据库参数来配置归档过程。Oracle的后台归档进程，archiver，必须知道把拷贝的重做日志文件放在什么位置、如何为文件命名，以及是否自动进行归档。

- **Archive Destination（归档目的地）** 通过在参数文件中添加如下行，告知数据库把归档重做日志文件存放在什么位置：

```
log_archive_dest_1 = "location=/oradata/PRACTICE/archive"
```

归档文件在被拷贝后被放置于该目录下。引号中的“location”一词意味着将文件拷贝到本地硬盘上的一个目录下面。归档文件可以被拷贝到多个目录中（最多5个）。在本练习中，一个目录就够了。

- **Archive Format（归档格式）** 在参数文件中指定归档文件的命名约定。在文件中添加以

下行:

```
log_archive_format = %s.arc
```

归档文件将以习惯方式命名, 其中%S是序列号。在定义归档日志名称的格式方面, 有一些选项, 但是在这里的练习中, 使用简单的命名格式。每当联机重做日志切换时, 就会产生一个新的序列号。在告警日志和v\$log视图的SEQUENCE#中可以看到这个序列号。

- **Archive Start (归档启动)** 设置LOG_ARCHIVE_START如下, 确保PRACTICE数据库自动归档重做日志文件:

```
LOG_ARCHIVE_START=TRUE
```

当下次启动数据库时, 设置的这些值才开始生效, 因为当数据库实例运行时, 参数文件才被读取。

修改完参数文件之后, 将数据库设置为“ARCHIVELOG”模式。为此, 先关闭数据库, 然后加载数据库。重新启动实例, 将读取文件initPRACTICE.ora中修改的参数。

当数据库已被加载, 但还未打开时, 如下所示改变数据库的归档模式:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE ARCHIVELOG;
```

打开数据库, 每次当前日志文件切换时, 都会在归档目的路径下生成一个文件。这些文件中保留了对PRACTICE数据库所做改动的运行记录。

在开始下一个任务之前, 我想就数据库参数提一点注意事项。许多数据库参数, 也被称为静态参数, 它们只能在参数文件中设置。其他一些参数, 动态参数, 还能够在数据库打开之后被修改。如果设置一个动态数据库参数, 设置的值将被保留, 直到重新设置参数值或重新启动数据库。当数据库重新启动时, 参数将采用初始化文件中的值或使用某一缺省值。LOG_ARCHIVE_DEST_1是一个动态参数; LOG_ARCHIVE_START和LOG_ARCHIVE_FORMAT是静态参数, 在数据库运行时不能修改。在使用SPFile特性的Oracle9i数据库的重新启动过程中, 动态参数具有持续作用。

使用如下alter system命令在一个已经打开的实例上设置归档目的地:

```
ALTER SYSTEM SET log_archive_dest_1 = "location=/oradata/PRACTICE/archive";
```

尽管无法动态设置参数LOG_ARCHIVE_START, 但可以在SQL*Plus输入提示中使用以下命令, 在一个已经打开的实例上启用自动归档:

```
ALTER SYSTEM ARCHIVE LOG START;
ARCHIVE LOG START;
```

所有参数都已确定, 归档模式也已设置, 请运行一些命令来验证参数设置是否起作用了。查看视图v\$archive_dest中的内容。检查一下归档路径是否正确。

```
SQL> SELECT dest_id, status, destination
2    FROM v$archive_dest
3    WHERE dest_id = 1
DEST_ID STATUS DESTINATION
-----
1 VALID /oradata/PRACTICE/archive
```


有效状态 (Valid) 意味着你已经正确的初始化了目的地, 而且该路径可以用于归档。

可以检查每个归档参数, 以确认每个设置项。可以从视图 `v$parameter` 中提取或使用 SQL*Plus 显示参数命令 (show parameter), 来查看任何一项参数设置:

```
SQL> show parameter log_archive_start
SQL> show parameter log_archive_dest_1
SQL> show parameter log_archive_format
```

另一个可用于查看归档信息的 SQL*Plus 命令是归档日志列举命令 (archive log list):

```
SQL> ARCHIVE LOG LIST
Database log mode                Archive Mode
Automatic archival               Enabled
Archive destination              /oradata/PRACTICE/archive
Oldest online log sequence       200
Next log sequence to archive     202
Current log sequence             202
```

该命令的输出结果给出了有关当前归档设置和状态的重要细节。在上述输出结果中, 可以看到:

- 数据库工作于 ARCHIVELOG 模式;
- 数据库将自动归档重做日志文件;
- 已归档的重做日志位于目录 /oradata/PRACTICE/archive 中;
- 最早的联机重做日志群组其序列号为 200;
- 将被归档的下一个已加载联机重做日志群组的序列号为 202;
- 当前联机重做日志文件的序列号为 202。

一旦检查发现参数文件设置已经起作用了, 打开数据库, 以便生成一些归档日志文件:

```
SQL> ALTER DATABASE OPEN;
```

现在设置好了 PRACTICE 数据库对重做日志文件进行归档。在只有少量事务活动的 PRACTICE 数据库上, 1MB 大小的重做日志不会经常被填满。当 LGWR 停止向一个联机重做日志群组写入, 并开始向另一个日志群组进行写入时, 出现日志切换。缺省情况下, 在当前联机重做日志群组填满时, 自动发生日志切换。也可以使用 ALTER SYSTEM 命令手工切换日志文件:

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

运行这个命令, 强制进行几次日志切换, 重新查看视图 `v$log` 和 `v$log_history`。请注意, 日志文件的序列号正在增长。每当切换日志文件时, 日志文件的序号就会增加 1。同样, 可以查看到归档目的文件的目录下现在有多个文件。如果使用通配符 %s 或 %S 定义文件格式, 那么文件中将包含日志文件的序列号。文件名中的序列号与视图中的序列号相符。观察日志文件的序号。请注意已归档的重做日志文件大小不一, 而且大都小于 1MB。归档进程创建最近的联机重做日志文件中重做内容的一个拷贝。由于强制执行了日志切换, 联机重做日志并不完整, 因此归档日志文件不是 1MB。

任务2：运行备份脚本

在将数据库PRACTICE设置为ARCHIVELOG模式并设置了必要的参数之后，准备进行数据库的另一个关闭备份。前面在练习4.1中所做的备份不能将数据库及时恢复到当前点。什么原因呢？因为从首次备份以来，数据库已经发生了变化。在对联机重做日志文件覆盖写入时重做信息已经丢失了。为此，Oracle推荐在数据库设置为ARCHIVELOG模式后进行完整的备份。使用整体一致的数据库备份和所有归档日志文件的拷贝，能够及时地把数据库恢复到任何时间点上。这种备份将涵盖练习4.1中创建的备份。

任务3：提前重做日志

完成了完整的数据库备份之后，在TINA.DATE_LOG中插入行，生成数据库的一些日常活动，然后强制执行日志切换。在下一任务中暂停数据库时，恢复进程需要至少一个归档日志文件来完成完全恢复。执行强制日志切换时，会在告警日志文件中看到以下内容：

```
Thread 1 advanced to log sequence 219
Current log# 3 seq# 219 mem# 0: /oradata/PRACTICE/redo03.log
```

重做日志中的该记录验证了日志切换的出现。也可以查询视图v\$log和v\$log_history，或使用ARCHIVE LOG LIST命令来验证日志顺序提前了。

任务4：删除一个数据文件

为了恢复PRACTICE数据库，先破坏这个数据库：用OS（操作系统）的delete命令删除users01.dbf文件。在Linux环境下，可以在数据库处于打开状态时删除文件。删除文件之后，关闭数据库，并注意看到的错误。在数据库关闭过程中，数据库试图更新所有数据文件的头部。当数据库试图更新数据文件users01.dbf的头部时，它无法找到该文件。这一错误会在SQL*Plus会话以及告警日志文件中给出。在Windows环境下，首先必须关闭数据库，然后才能删除文件。Windows会锁定这些文件以防它们被删除。当试图打开数据库时，会看到同样的出错信息，如下所示：

```
ORA-01157: cannot identify/lock data file 3 - see DBWR trace file
ORA-01110: data file 3: '/oradata/PRACTICE/users01.dbf'
ORA-27041: unable to open file
OSD-04002: unable to open file
O/S-Error: (OS 2) The system cannot find the file specified.
```

Oracle现在无法打开丢失的文件了。在\$ORACLE_BASE/admin/PRACTICE/bdump中会产生一个包含错误的额外细节的DBWR跟踪文件。

任务5：还原丢失的数据文件

虽然可以在数据库打开时恢复丢失了的数据文件，但这里建议你以中止实例的方式（SHUTDOWN ABORT）关闭Linux数据库。一旦数据库关闭了，就可以尝试使用正常启动。当尝试打开数据库时，会面临同样的关于users01.dbf文件无法找到的错误。当数据库正在使用时，查看v\$recover_file中的内容。该视图中列出了所有需要恢复的文件，并就各个文件为何必须恢复给出了一个错误提示信息。

技巧 丢失了一个非系统数据文件时，在该数据文件的表空间脱机后，将可以恢复数据文件。在恢复数据库时，数据库还将向用户开放。我首先举例说明恢复一个正在用的数

数据库。在下一章中，将学习如何在数据库打开时恢复一个丢失了的数据文件。

```
SQL> select file#, error, change# from v$recover_file;
FILE# ERROR                                CHANGE#
-----
3 FILE NOT FOUND                          0
```

文件序号3肯定会被恢复，这是因为数据库打开过程试图打开该数据文件，但是未能找到该文件。从任务2中所做一致备份的备份目录下复制users01.dbf文件。有了这个文件之后，v\$recover_file将会显示出某些不同：错误项没有了，CHANGE#列有一个数值。将此数值与v\$datafile视图对应于所有文件的检查点变化数值进行比较，将会看到v\$datafile视图中的SCN要比v\$recover_file中的数值大。v\$datafile从控制文件中读取关于PRACTICE数据库中数据文件的信息，而v\$recover_file则显示刚刚还原的文件其头部的变化次数。因此，除非对应于所有联机数据文件的SCN与数据文件头部以及控制文件中的相同，否则数据库无法打开。这几个数值必须相同。在打开数据库之前为了达到这种一致性，必须进行数据库恢复。

任务6：恢复还原的数据文件

为了使所有的数据文件拥有一致的变化序号，需要来自重做日志中的重做信息。所需的每个变化都包含在一个或多个归档日志和当前联机重做日志中。查看是否能确定为实行完全恢复而需要对文件users01.dbf应用的重做文件。注意v\$recover_file和v\$datafile中对应此文件的变化序号。查看v\$log_history和v\$log，找出哪个重做日志文件是必须采用的。这需要那些序号范围跨越了从备份时起到当前重做日志文件这段时间的重做日志文件。

```
SQL> SELECT * FROM v$log_history WHERE ROWNUM < 10
2 ORDER BY sequence# DESC;
SQL> SELECT * FROM v$log;
```

输出的结果可能与表4-1类似。例如，假设v\$recover_file给出的users01.dbf文件的CHANGE#是64300，v\$datafile视图给出的检查点为64440，那么哪个重做日志中包含了使该文件与控制文件一致的那些变化呢？完全恢复需要来自序号为218的已归档重做日志文件和联机重做日志文件219中的重做信息。

使用recover database命令恢复数据库：

```
SQL> RECOVER DATABASE;
```

使用这个命令时，Oracle确定哪个数据文件需要被恢复以及需要哪个重做日志文件。当介质恢复完成后，系统将给出如下提示：

```
Media recovery complete.
```

表4-1 重做日志文件中的变化序号

v\$log_history Sequence	First Change#	Next Change#
218	64160	64349
217	44155	64160
216	44154	44155

(续)

v\$log Sequence	First Change#	Archived
219	64349	NO
218	64160	YES
217	44155	YES

完成恢复之后，检查v\$logrecover_file的内容。没有任何返回行。这表明users01.dbf文件所需的所有重做信息均已被采用。查看告警日志，在恢复过程中都发生了哪些行为？

```
ALTER DATABASE RECOVER database
Sun Jan 2 08:20:28 2002
Media Recovery Start
Media Recovery Log
Recovery of Online Redo Log: Thread 1 Group 2 Seq 218 Reading mem 0
  Mem# 0 errs 0: /ORADATA/PRACTICE/redo02.log
Recovery of Online Redo Log: Thread 1 Group 3 Seq 219 Reading mem 0
  Mem# 0 errs 0: /ORADATA/PRACTICE/redo03.log
Media Recovery Complete
Completed: ALTER DATABASE RECOVER database
```

你是否正确判断出了恢复需要哪个重做日志文件呢？

任务7：确认数据库已恢复

在转入本章最后一个练习之前，请检查TINA.DATE_LOG中的数据库最近的日期/时间值（date/time）。该时间应当是在任务4中删除文件以破坏数据库之前，从表中提取的最新时间。

恭喜！你已经成功地完成了一次完全数据库恢复。

练习4.4：不完全数据库恢复

到现在为止，你已经完成了一次成功的数据库还原和一次数据库完全恢复。不完全恢复使用一个数据库备份，并将其回退到此前的某个时间点。采用的不是最近的备份之后的所有重做记录，而是截止到某一特定时间点的重做日志。在以下情况下可能会需要对整个数据库进行不完全恢复：

- 由于失误而丢失了一个数据库对象。
- 丢失了部分或全部联机重做日志。
- 在恢复过程中丢失了一个已归档的重做日志。
- 错误地删除了表空间。

不完全介质恢复是以从备份中还原所有的数据文件开始的。给出恢复命令来指定在哪个时间点上希望恢复操作终止。当打开数据库时，将不得不忽略剩余的数据库重做信息。这么做是因为在采用了所有的重做信息之前选择了停止重做信息应用。一旦数据库打开，被忽略的重做信息就不能再用了。忽略剩余的重做信息是通过以RESETLOGS选项打开数据库来实现的。重置（resetting）日志文件创建了数据库的一个新实体。已恢复的数据库的序号将以新的日志序列流开始，起始的日志序号为1。

为了说明备份和不完全恢复，在接下来的练习中，需要完成一些与练习4.3中类似的任务。

在本练习中，删除INDX表空间，并将数据库恢复到此“误操作”发生前的时间点。

任务描述	时间（分钟）
1. 删除表空间	10
2. 检查恢复	10
3. 还原数据文件和控制文件	10
4. 不完全恢复数据库	10
5. 以重置日志选项打开数据库	5
6. 确认数据库恢复	5
总计时间	50

在进行任务4时，有三种恢复选项。如果每种选项都希望尝试一下，那就需要重复本练习前面的所有3个任务。

任务1：删除表空间

本练习的第一个任务就是营造一个不完全恢复起作用的环境，因此删除数据库PRACTICE中的INDX表空间。在做这项工作之前，确保在TINA.DATE_LOG中插入了新的行。练习结束时，将检查该表中插入的最后一行。还有，同样也要进行至少3次强制日志切换。这种恢复将使用已归档日志，而不只是联机日志。

在本练习中我选择了删除一个表空间，这是因为删除表空间的命令（drop tablespace）会向告警日志写入一条消息，而删除表（drop table）或截断表（truncate table）却不会。在恢复检查中会用到这个告警日志信息。

在删除表空间之前，请注意检查TINA.DATE_LOG表中早于drop命令的最近时间：

```
SQL> SELECT TO_DATE(max(create_date), 'HH24:MI:SS') latest_time
2 FROM tina.date_log;
```

为了删除INDX表空间，使用以下命令：

```
SQL> drop tablespace indx including contents;
Tablespace dropped.
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

执行完此命令之后，请查看v\$tablespace和dba_tablespace视图。注意，INDX表空间不出现在这些视图中了。在进行不完全恢复之前，请确信已经在TINA.DATE_LOG中插入了更多的行。

警告 删除表空间命令（drop tablespace）是一种无法取消的操作。请绝对确保在使用此命令之前，已经连接上了PRACTICE数据库。

任务2：检查恢复

在数据库出现错误时，你可能会接到反应问题的电话。用户或开发人员可能并不知道事故发生的准确时间。当然他们也不会知道事故发生时日志文件的序号或SCN的值。根据已经发生的事故，有时可以确定出事故是何时发生的。因为对数据库结构所做的变动（例如删除某个表空间）将为alter.log添加一行新内容。而对于对象变动，比如删除某个表，你就必须利用有关人员提供的近似时间。另外，也可以使用LogMiner试着找出更精确的时间（在第10章中介绍）。本练习中，在还原并恢复数据库之前，请检查告警文件、v\$log表和v\$log_history表，以确定如何

最好地恢复数据库。

查看告警日志文件，找出DROP TABLESPACE命令以前的日志切换。其他主要的数据库事件也记录在此文件中。告警日志将指出该表空间已被删除了。打开告警日志，查找与下面这些文字类似的内容：

```
Thread 1 advanced to log sequence 223
Current log# 1 seq# 223 mem# 0: /oradata/PRACTICE/redo01.log
...
Fri Jan 4 09:07:10 2002
drop tablespace indx including contents
Fri Jan 4 09:07:12 2002
Completed: drop tablespace indx including contents
```

结果表明，在一月四日，星期五，有人执行了删除表空间命令（drop tablespace）。该命令在2秒钟之内执行完毕。就在该命令之前，告警日志指出发生了一次日志切换。日志序号223是表空间被删除时的当前序号值。为了找出在重做日志序号223中的第一个变动序号，需要使用以下查询命令查看视图v\$log_history：

```
SQL> SELECT sequence#, first_change#
2 FROM v$log_history WHERE sequence# = 223;
```

对于本示例，first_change#出现在SCN 64300处。

在从一致备份中还原数据文件和控制文件之前，必须对PRACTICE数据库进行检查。一旦从现有的控制文件上进行复制，就会丢失执行不完全恢复所需的信息。你可能会想在这个数据库关闭后（就像练习4.1中的那样）将它拷贝到其他地方。如果选择了错误的停止点来终止恢复，可能会需要这个拷贝。然后可以还原并启动原来的数据库，以检查更多的内容。

根据检查结果可以得出结论：在一月四日星期五上午9时7分10秒时执行了一次删除表空间命令。当前重做日志文件是序号为223的日志文件，并从更改号码64300处开始记录数据库的变化。可能希望将数据库恢复到删除表空间之前的某一点。你可以将这个不完全恢复执行到某一点上、某一数据库更改号上或某一特定的日志文件上。当数据库不完全恢复时，将丢失结束恢复点之后出现的事务。但是，为了找回INDX表空间，付出这种代价是值得的。

任务3：还原数据文件和控制文件

当执行不完全恢复时，希望从备份的时间之前的某个备份还原除联机重做日志文件以外的所有数据库文件。没有一个数据文件能够拥有更改号大于所希望备份到的时间的更改号。因此，关闭数据库，从练习4.3所做的关闭备份中复制所有的数据文件和控制文件。剩下当前重做日志存放在/oradata/PRACTICE目录下，因为可能需要联机重做日志文件中的重做信息，将数据库恢复到表空间被删除之前的时间点上。

任务4：不完全恢复数据库

在开始不完全恢复之前，确认/oradata/PRACTICE目录中包含了来自最近的数据库一致备份的数据文件和控制文件。重做日志文件应该就是当你“不慎”删除掉INDX表空间时存在于数据库中的那个。

使用alter database命令恢复数据库，并用UNTIL子句指定恢复过程的持续时间。可以将数据

库恢复到某一特定的时间、更改号，或重做日志。

- **基于时间的恢复** 通过指定时间参数，告知PRACTICE数据库实例应用重做日志文件，直到日期所指定的时刻。重做日志是在已归档日志中找到的，同时对于任何数据文件，联机重做日志文件以最小的更改号开始。日期必须是符合YYYY-MM-DD:HH24:MI:SS格式的字符。

打开告警日志，找到前面讨论过的删除表空间命令的日期。将这个日期值转换为ALTER DATABASE命令要求的格式。用类似下面这样的命令对PRACTICE数据库执行一个基于时间的恢复：

```
SQL> ALTER DATABASE RECOVER AUTOMATIC UNTIL TIME '2002-01-04:09:07:10';
```

按照上面给出的告警日志，删除表空间出现在7分12秒。恢复命令将把数据库还原到7分10秒的时刻。重做将被应用到删除表空间命令刚出现之前的时刻。如果在此次恢复操作之后没有遇到问题，会得到一个DATABASE ALTERED响应。此时数据文件包含了它们在事故发生之前2秒时的所有信息。为了打开数据库，必须重新设置重做日志。

注意 恢复常常并不能完全精确到某一指定时间。估计错误出现时间，并将恢复停止点设置在此时间之前不久的地方。

- **基于变更的恢复** 可以指定一个SCN值用于恢复。你可以这样提供SCN值：

```
SQL> ALTER DATABASE RECOVER AUTOMATIC UNTIL SCN 64300;
```

在RECOVER中的CHANGE选项说明这是基于变更的恢复。这个参数将数据库恢复到整数所给定的系统更改号（SCN）之前的某一事务一致状态。关键是在恢复命令上设置哪个SCN值。数据文件将与结尾处的更改号64300协调一致。拥有问题出现时的准确SCN（这对于非结构化的数据库变化是较少见的）时，使用基于变更的恢复会比较准确，否则基于时间的恢复是一种更为精确的用于停止恢复的方法。

- **基于取消的恢复** 可以使用CANCEL关键字，执行不完全数据库恢复。基于取消的恢复过程对数据库进行恢复，直到输入了CANCEL。重做日志是一个接一个地应用，直到用到了希望停止的那个重做日志文件。

用ALTER DATABASE命令启动基于取消的恢复：

```
SQL> ALTER DATABASE RECOVER UNTIL CANCEL USING BACKUP CONTROLFILE;
```

如果使用关键字AUTOMATIC，Oracle会自动从init.ora参数中提取出归档日志文件名称和路径，并应用所找到的各个文件。当所推荐的下一个日志无法找到时，恢复将会停止。当某个归档日志文件丢失时这会有用，它可以防止完全恢复。对于本练习中基于取消的恢复，必须手工应用日志文件名称，然后在遇到表空间被删除时的当前日志文件时，按下CANCEL键。如果是将数据库恢复到某个SCN或时间点上，你无需像进行基于取消的恢复那样，与恢复过程进行交互。

你希望应用所有的重做日志，直到包含了DROP TABLESPACE命令的那个日志。连续应用重做日志，直到你希望终止的那个重做日志出现。只要看到了不希望应用的那个文件，就按CANCEL键。在第5章中，我们会更详细地讨论基于取消的恢复。

任务5：以重置日志选项（Reset Logs）打开数据库

完成了不完全恢复后，必须重新设置联机重做日志。当使用RESETLOGS选项打开数据库时，所有的数据文件都获得一个新的RESETLOGS SCN和时间戳。已归档的重做日志在它们的头部也有这两个值。Oracle软件通过为数据库提供与重做日志文件相匹配的RESETLOGS SCN和时间戳，来防止用旧的归档日志破坏数据文件。

为了打开数据库并重新设置日志，可以使用以下命令：

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

在使用此命令时，会发现该命令的执行需要一些时间。联机重做日志文件被还原，每个数据文件的头部都被更新，控制文件也被更新。所有这些工作完成之后，数据库打开。注意每个联机重做日志文件的新序号。

```
SQL> SELECT group#, sequence#, archived, status
2 FROM v$log;
```

```
GROUP# SEQUENCE# ARCHIVED STATUS
```

```
-----
1          0 YES      UNUSED
2          1 NO       CURRENT
3          0 YES      UNUSED
```

现在重做日志文件再次开始计数。你已经创建了PRACTICE数据库的一个新实体。查看告警日志文件，浏览下面信息：

```
Mon Aug 27 06:54:49 2001
```

```
RESETLOGS after complete recovery through change 64300
```

对日志的重新设置出现在某个特定的SCN上。在SCN 64300以后的任何重做信息都没有被应用于数据文件。

任务6：确认数据库恢复

在恢复成功之后，可以确认恢复过程是否保留了你所期望的数据库对象。借助v\$tablespace或dba_tablespace视图查看数据库的新实体中是否包括INDX表空间。确实看到INDX表空间在列表中了。从TINA.DATE_LOG中提取出表中最新的日期/时间值。假如在数据库打开之后没有向这个表中插入任何行，那么会得到一个时间值，该值比删除表空间命令出现的时间要小。丢失了删除表空间以后向TINA.DATE_LOG表中插入的那些行。

恭喜！你已经成功地完成了一次不完全恢复。

注意 在使用RESETLOGS选项打开数据库之后，必须做一个完整的数据库备份。要还原一个以前的数据库并将它前滚到RESETLOGS执行后的某个时间几乎是不可能的。

4.3 疑难解答

在进行本章的练习时，你可能会遇到一些错误可以参考以下帮助。

4.3.1 Alter Database Open

在数据库启动时，可能会遇到下面一些问题：

1) ORA-00313: open failed for members of log group string of thread string.

日志文件保存在控制文件认为它应该保存的地方。查看控制文件所期望的地方是否有这样一个重做日志。还要检查是否有正确的日志文件可用。如果这些都没问题,就不得不实施不完全恢复,直至到达这个错误日志文件。使用RESETLOGS选项打开数据库可以重新创建任何丢失的日志文件。

2) ORA-00314: log string of thread string, expected sequence# string doesn't match string.

打开的日志文件超出了序列。有可能在还原之后未能从备份中复制联机重做日志。

3) ORA-01157: cannot identify/lock data file string—see DBWR trace file.

打开数据库时, Oracle无法找到某个数据文件。还原该文件并恢复数据库。

4.3.2 Startup Mount

ORA-00205: error in identifying controlfile, check alert log for more info.

当数据库启动时,它会寻找控制文件,并设置该控制文件。如果数据库实例找到参数文件中列出的控制文件,会看到这条消息。请确认控制文件保存在参数文件指定的地方。

4.3.3 Alter System Switch Log File

Hangs: 请确认包含归档日志文件的磁盘没有被填满。

4.4 小结

在本章中,学习了利用冷备份和归档日志进行用户管理物理数据库还原和恢复的基本原理和方法,还学习了恢复数据库的一些关键命令,并借助动态v\$视图创建一个冷备份脚本。

下一章,将学习如何在数据库打开时对数据库文件进行备份。利用这些文件,可以对PRACTICE数据库实现完全和不完全恢复。

习题

以下是一些问题,用于复习本章中的一些重要概念。

1. 为什么要将数据库设置在ARCHIVELOG模式?

- A. 可以完全恢复数据库或将其恢复到某一特定的时间点上
- B. 服务器有额外的磁盘空间需要填充
- C. 控制文件需要保存以便安全存放
- D. 数据文件在联机时可以被压缩

2. 哪个v\$视图将为你提供归档重做日志文件的序号?

- A. v\$datafile
- B. v\$log
- C. v\$log_history
- D. v\$database

3. 当数据库未打开时,如何知道哪个文件需要恢复?

- A. 从v\$recover_file中选择
 - B. 查看告警日志文件
 - C. 检查参数文件
 - D. 从v\$broken_datafile中选择
4. 哪个SQL*Plus命令使你能够快速浏览数据库的当前归档状态?
- A. ARCHIVE STATUS
 - B. SHOW PARAMETER ARCHIVE
 - C. ALTER SYSTEM SWITCH LOGFILE
 - D. ARCHIVE LOG LIST
5. 选出不完全恢复的三种类型。
- A. 基于取消的恢复
 - B. 基于时间的恢复
 - C. 基于错误的恢复
 - D. 基于变更的恢复

答案

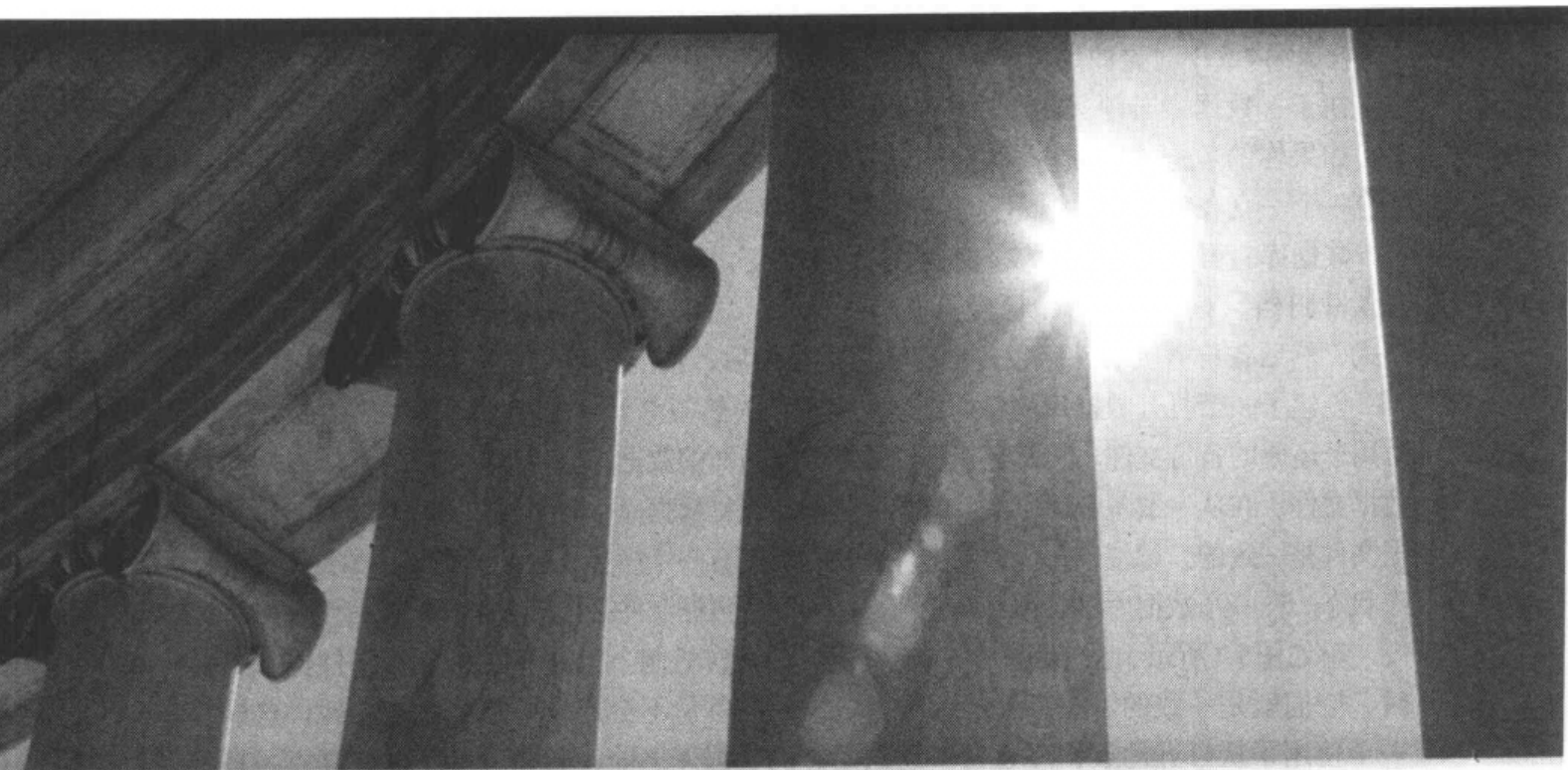
1. A。当数据库处于ARCHIVELOG模式时，重做日志文件内容被拷贝到指定的地方。通过把这些文件应用于数据库的还原部分，可以重做数据库以前所完成的工作。

2. C。v\$log_history给出了归档重做日志文件的一个详细列表、更改号的历史记录、用于日志切换的日志序号。

3. A。虽然告警日志确实列出某个文件需要恢复，但它不象v\$recover_file那样列出所有需要恢复的文件。

4. D。Archive Log List给出了数据库的归档模式、归档目的路径、最早的以及下一个联机重做日志群组，以及归档是手工还是自动模式。

5. A、B、D。基于错误的恢复不是一种指定恢复停止点的方法。



第5章 从打开的数据库备份与恢复

就像第4章中讨论的那样，可以从一个一致备份中还原并恢复一个数据库。一致备份的严重弊端在于：数据库必须关闭。由于用户以及应用程序一直在使用数据库，因此不可能关闭数据库。如果你的数据库必须一直保持可用状态，那么关闭数据库备份就不可取了。那么怎样才能备份一个一直在使用的数据库呢？答案是采用热备份方式。

数据库打开时，数据库文件的状态一直在变化。数据文件和控制文件被更新，重做日志被写入并归档。对于备份一个已经打开的数据库，用户管理的办法是把每一个表空间都置于备份模式，然后备份数据文件。在备份完数据文件之后，再把表空间恢复到正常状态。

创建了一个打开数据库的备份之后（也被称为热备份），就拥有了一个可以在数据库出现故障时用于还原的备份文件。假设必须修复某个遭到损坏的数据库，可以从备份路径下拷贝一个或多个数据文件。在从一致备份还原文件时，如果没有恢复数据库，将无法打开该数据库。原因在于：数据文件是一致的。记住，每个数据文件在其首部都有一个序号，该序号给出了数据文件的最新数据库状态。在一个数据库可以被打开之前，所有联机数据文件必须拥有同样的系统更改号（SCN）。

正如图5-1所示，SCN是Oracle自身相关的记时机制。当这些联机数据文件拥有相同的SCN时，它们就是一致的。在打开的数据库上进行的备份不会产生一致的数据文件。这样，在打开一个使用了从打开的数据库备份中还原文件的数据库之前，必须进行介质恢复。一旦所有必需的重做信息都被应用而保持SCN一致时，就可以打开数据库了。在一个打开的数据库上实施备份可以允许用户以及应用程序在整个备份操作期间使用整个数据库。

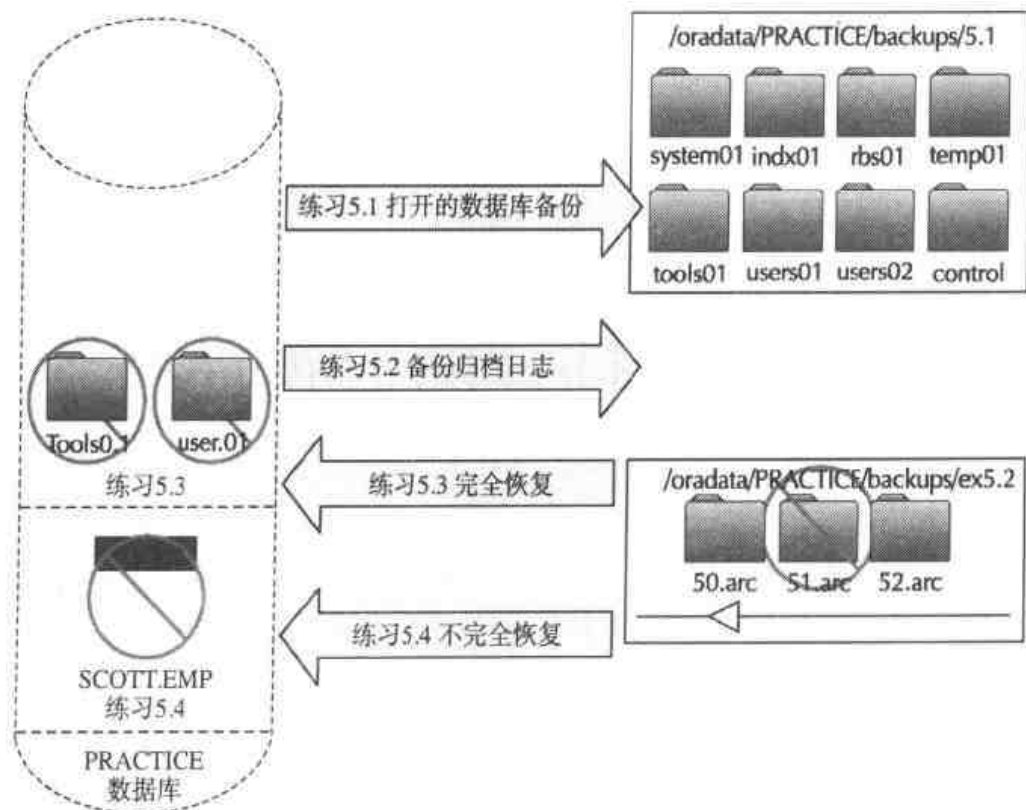


图5-1 打开的数据库的备份、还原和恢复操作

数据文件首部的SCN值

在数据文件的首部，有多个SCN值。当创建数据文件时，就分配一个创建SCN。当文件在经历其最后一个检查点时，检查点SCN标示出数据库或数据文件的状态。这是个重要的SCN值，用于确保在启动时数据文件与控制文件保持一致。在数据文件脱机时，脱机SCN描绘了数据文件的状态；当数据文件返回到联机状态时，联机SCN将被更新。备份SCN则记录下何时表空间被置于热备份模式，重置日志SCN用于记录何时数据库是使用RESETLOGS参数打开的。控制文件中类似的SCN值能确保在数据库启动和恢复时，所有的数据文件与控制文件一致，而且数据文件之间也相互一致。

5.1 打开的数据库的整体备份

为了在数据库打开时进行整体备份，所有的数据文件都被拷贝到其他位置（通常是磁盘或磁带）。在数据文件被拷贝期间，对该数据文件的改动不断地进行着。当拷贝完成后，数据库会继续改动已拷贝的数据文件。Oracle怎样做到数据文件在被改动时仍然能够被进行用于备份的拷贝？方法是在拷贝数据文件之前，将该文件的表空间置于备份模式。将一个表空间置于备份模式即是通知Oracle更新该表空间的数据文件的首部。Oracle同时也修改其对应这些数据文件的重做日志记录。当表空间处于备份模式时，会发生以下事件：

- 每个文件首部的标记都被设置，指出即将进行热备份。
- 表空间数据文件执行某一检查点。内存中所有“脏”数据块都被写入该文件中。检查点的SCN被写入数据文件的首部和控制文件中，这个开始备份SCN标识出对文件所做的最新改动。这时SCN结构对于文件中的任何变化都是冻结的。
- 为告警日志文件添加一个开始备份记录。
- 在数据文件中的任何块被首次改变之前，有关每个已变化块的一个块映像将被拷贝到重做日志中。然后针对该块上的变动生成一个标准的重做向量。接下来对同一块所做的所有改变将产生正常的重做更改向量。

表空间处于备份模式时，可以使用操作系统的复制（拷贝）命令，将数据文件拷贝到其他地方。如果多个用户正在更新这些表空间，则热备份期间产生的重做日志的数量会明显增加，因为重做日志中必须保留每个已变化数据块的一个拷贝（仅对第一次变化）。因此，应在数据库活动较少时进行打开数据库的备份。

数据文件拷贝完毕后，将表空间脱离备份模式。当标志一个表空间备份模式终止时，会发生以下事件：

- 热备份标志被清除，说明备份已经结束。
- 在重做流中记录下终止备份SCN，作为一个重做矢量。Oracle利用这个重做记录，可以知道在数据文件恢复过程中何时将表空间脱离热备份。
- 数据文件的检查点结构解除冻结，并与数据库的其他部分相匹配。
- 重做生成返回到正常方式，就像表空间被设置为备份模式之前的那样。

由于备份期间所做的更改必须保存在重做流中，而且不能丢失，因此数据库必须处于归档

日志模式，以便用于打开数据库备份。

注意 不能将一个只读表空间设置为备份模式，因为该表空间无法被数据库更改。也不能将一个临时本地管理的表空间设置为备份模式。要还原/恢复一个本地管理的临时表空间，只需重新创建即可。

练习5.1：备份打开的数据库

本练习中，将在PRACTICE数据库打开时，创建一个该数据库的完整备份。为此，使用ALTER TABLESPACE...BEGIN BACKUP命令，对表空间中的数据文件实施一个打开的备份。使用ALTER TABLESPACE...END BACKUP或ALTER DATABASE END BACKUP命令，使表空间脱离热备份模式。当每个表空间都处于备份模式时，拷贝该表空间对应的数据文件。除了拷贝每个联机数据以外，一个完整的数据库打开备份还包括对当前控制文件的备份拷贝。

任务描述	时间（分钟）
1. 产生数据库活动	5
2. 创建备份脚本	15
3. 运行热备份	5
4. 检查备份命令	5
总计时间	30

在确信数据库已经有了一些操作活动之后，创建一个脚本文件，其中包含了完成一个合法的完整打开数据库备份所需的所有命令。运行备份脚本，然后再检查已完成的工作。

任务1：产生数据库活动

向Tina的DATE_LOG表中插入一些行。如果用的是还原，请确信恢复已经结束。确信借助任务系统，一些行被插入到表中。可以将任务的时间间隔设为每分钟一次，以增加数据库的操作活动。

任务2：创建热备份脚本

创建数据库中所有文件的一个热备份。在nutshell中，准备生成的打开的数据库备份脚本将完成以下工作：

- 1) 切换日志文件，确保在备份前所做的所有更改都已被归档。
- 2) 将一个表空间设置为热备份模式。
- 3) 用操作系统命令拷贝该表空间中的文件。
- 4) 将该表空间脱离热备份模式。
- 5) 对于每个表空间，重复步骤2)~4)。
- 6) 切换日志文件，确保备份后所做的所有更改都已被归档。
- 7) 备份当前控制文件。

在运行热备份脚本时，使用SQL、SQL*Plus命令以及PL/SQL来创建一个有关SQL命令和操作系统命令的可执行文件，该文件（我们称之为备份命令文件）还将创建一个文件用来显示输出结果，用到的文件有：

- 备份脚本文件（open_backup.sql） 这是一个在文本编辑器（后面列出）中创建的文件。

- 备份命令文件 (open_backup_commands.sql) 该文件是在open_backup.sql中创建并执行的。
- 备份命令文件输出文件 (open_backup_output.sql) 该文件捕捉open_backup_commands.sql执行的那个文件的输出结果。

查看open_backup.sql的内容, 注意其中有些行的注释。可以将下面列出的这些命令键入文件中, 或从Osborne的Web站点下载:

```

1.  set feedback off pagesize 0 heading off verify off
linesize 100 trimspool on
2.  define dir = '/oradata/PRACTICE/backup/ch5'
3.  define fil = '/tmp/open_backup_commands.sql'
4.  define spo = '&dir/open_backup_output.lst'
5.  prompt *** Spooling to &fil
6.  set serveroutput on
7.  spool &fil
8.  prompt spool &spo
9.  prompt archive log list;;
10. prompt alter system switch logfile;;
11. DECLARE
12.  CURSOR cur_tablespace IS
13.  SELECT tablespace_name FROM dba_tablespaces
      WHERE status <> 'READ ONLY';
14.  CURSOR cur_datafile (tn VARCHAR) IS
15.  SELECT file_name
16.  FROM dba_data_files
17.  WHERE tablespace_name = tn;
18.  BEGIN
19.  FOR ct IN cur_tablespace LOOP
20.  dbms_output.put_line ('alter tablespace
'|ct.tablespace_name||' begin backup;');
21.  FOR cd IN cur_datafile (ct.tablespace_name) LOOP
22.  dbms_output.put_line ('host cp '|cd.file_name||' &dir');
23.  END LOOP;
24.  dbms_output.put_line ('alter tablespace
'|ct.tablespace_name||' end backup;');
25.  END LOOP;
26.  END;
27.  /
28.  prompt alter system switch logfile;;
29.  prompt 'alter database backup controlfile to '&dir\backup.ctl''
REUSE;'
30.  prompt archive log list;;
31.  prompt spool off;;
32.  spool off;
33.  @&fil

```

这个脚本将创建运行完整的打开的数据库备份所需的所有命令。第一行设置了一些

SQL*Plus变量，正如以前脚本中所做的那样（feedback=off, pagesize=0, heading=off, verify=off, linesize=100, trimspool=on）。现在让我们分析一下该脚本中的部分行，选择一些要点给予解释：

- 第2-4行设置了将被用于脚本中的用户变量，这些变量包括了包含有备份命令的文件、拷贝文件的目录以及用于重要脚本输出的一个文件。真正的备份命令被写到一个临时目录下的一个文件中。对于一个已经存在的数据库，并不希望运行备份脚本或使用备份命令时出错。请注意第4行，通过引用一个用户变量的内容，定义了一个用户变量。
- 第6行通知PL/SQL代码调用dbms_output存储过程，把输出结果显示在屏幕上。这些屏幕文本被写到命令文件中。Dbms_output包是Oracle提供的众多的PL/SQL包之一。
- 第7行开始将所有来自SQL、SQL*Plus命令和PL/SQL代码的输出结果记录在一个文件中，该文件是在前面用户变量中已经定义了的。
- 第8行向spool文件写入一条spool命令。当通过/tmp/open_backup_commands.sql中创建的脚本运行备份命令时，希望该脚本的输出结果能够提供一些重要的归档日志序号。
- 第9行通知命令文件列出当前正在归档的信息，提供给命令文件的spool文件。当从这个备份恢复数据库时，需要了解当前日志文件的序号。
- 第10行将日志文件从当前日志文件切换到下一个日志文件。这将强制对所有的联机数据文件实施一个检查点，并创建一个新的归档日志文件。
- 第11-27行包括了那些将各个表空间置于备份模式、调用所有的数据文件到某个备份目的路径以及将表空间脱离备份模式的PL/SQL代码。这些代码中有两个游标：所有数据库表空间的一个列表和表空间中所有数据文件的一个列表。（此处没有包括只读表空间，因为Oracle不允许将这种表空间置于备份模式。）表空间游标在第一个For循环中打开，dbms_output.put_line存储过程调用显示出备份命令脚本中所需的命令文本。在使用当前表空间名称打开数据文件游标之前，每个表空间都被置于备份模式，每个数据文件都使用操作系统的拷贝命令进行拷贝（对于Linux，使用CP命令；对于Windows，使用Copy命令）。一旦该表空间内所有的数据文件都被读取，并且拷贝命令已经创建，表空间就被解除备份模式。这个循环过程对数据库中的所有表空间连续进行。此处选择备份脚本中的PL/SQL，这样就可以使用PL/SQL循环机制来复制每个表空间内的所有文件。可以修改这个PL/SQL块，只对联机表空间或只对特定表空间创建备份。为此，可以向这个表空间游标中添加WHERE条件，类似于前面的READ ONLY。
- 第28行将日志文件从当前日志文件切换到下一个日志文件。强制对所有的数据文件实施一个检查点，并创建一个新的归档日志文件。这同时保证了热备份过程中创建的重做条目得以归档。
- 第29行在数据文件备份完成之后立即创建控制文件的一个拷贝。当使用这个备份进行恢复时，可能会需要这个控制文件的一个拷贝。备份控制文件的主要方法是使用一个SQL语句生成该文件的一个准确的二进制拷贝。可以把这个备份控制文件用于采用了这些备份数据文件的数据库恢复。如果文件已经存在，那么脚本中的REUSE关键字将会覆盖这个文件。
- 第30-32行指示open_backuo_commands.sql文件将归档信息列出到备份脚本中的spool文件中。如果已经将这个打开数据库备份与备份归档日志组合，将会得到归档序号的一个记录。

也可以使用与备份实施相近的时间, 来确定哪个归档信息是恢复所需的。最后, 命令SQL文件终止假脱机。

- 第33行执行创建的备份SQL命令文件。

警告 当将原始设备用于Oracle数据库时, 必须使用“CP”选项; “dd”命令是一个合适的替代选项。

只要将所有这些命令键入一个名为open_backup.sql的文件, 就可以用SYSDBA角色(或类似的特权用户), 以SYS连接数据库, 在SQL*Plus命令提示符下运行该文件。

任务3: 运行备份脚本

在SQL*Plus命令提示符下启动刚才创建的脚本, 当脚本运行时, 表空间和数据文件被置于备份模式。在备份过程中, 可以从另一个SQL*Plus会话中查看v\$backup视图, 查看哪个文件现在处于备份模式。在脚本结束之后, 还可以将用户表空间置于备份模式, 并查看所有联机数据文件的备份状态:

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
SQL> SELECT d.tablespace_name tablespace, b.file#,
2         b.status, b.change#, b.time
3   FROM dba_data_files d, v$backup b
4  WHERE b.file#=d.file_id
5  ORDER BY tablespace_name;
TABLESPACE FILE# STATUS CHANGE# TIME
-----
INDX      6 NOT ACTIVE      84972 05-JAN-02 09:30
RBS       2 NOT ACTIVE      85446 05-JAN-02 09:35
SYSTEM    1 NOT ACTIVE      85458 05-JAN-02 09:42
TEMP      4 NOT ACTIVE      85712 05-JAN-02 09:38
TOOLS     5 NOT ACTIVE      85452 05-JAN-02 09:40
USERS     3 ACTIVE          85722 05-JAN-02 11:59
USERS     7 ACTIVE          85722 05-JAN-02 11:59
SQL> ALTER TABLESPACE users END BACKUP;
```

用户表空间中有两个数据文件, 文件序号分别为3和7。这些文件的开始备份检查点SCN是85722, 开始备份命令是1月5日上午11点59分提交的。

在本练习中, 创建并运行了一个脚本文件。DBA实施的大多数备份操作都是通过脚本完成的。为了帮助你熟悉脚本操作, 请逐行输入任务2中的命令, 观察会发生什么事情。

技巧 可以使用Linux的Compress工具或Windows的Zip工具压缩备份的数据库文件, 这样可以节省空间。切记不要在复制操作结束之前压缩一个文件, 而且要注意使用压缩工具可能会给文件造成损害。

打开备份运行之后, 查看告警日志文件, 将得到如下的备份模式记录:

```
Wed Jan 05 09:30:00 2001
alter tablespace SYSTEM begin backup
Wed Jan 05 09:30:05 2001
Completed: alter tablespace SYSTEM begin backup
```

```
...
Wed Jan 05 09:31:00 2001
alter tablespace SYSTEM end backup
Wed Jan 05 09:31:05 2001
Completed: alter tablespace SYSTEM end backup
```

这个告警日志显示了一些有关SYSTEM表空间打开备份操作的时间信息。上午9点30分，脚本生成的命令文件提交命令，将SYSTEM表空间置于备份模式。5秒钟后，SYSTEM表空间真正处于备份模式。像前面讨论的那样，Oracle软件必须在表空间进入备份模式之前完成一些工作。复制一个SYSTEM数据文件所需的时间略少于55秒。上午9点31分，提交命令，将SYSTEM表空间脱离备份模式。5秒后，SYSTEM表空间恢复为正常模式。

警告 应该标识表空间备份的起始点和结束点。如果在一个打开的数据库中备份数据文件时没有将表空间置于备份模式，对恢复而言，备份文件就不会起什么作用。所期望的恢复操作会引起文件错误，并阻止你打开数据库，原因是Oracle无法确保这些数据文件的一致性。另外，如果将某个表空间停留于备份模式，就无法正常关闭数据库。数据库将会报告，在正常关闭数据库过程中出现了错误。

本练习中的命令实现了一个连续的打开备份。所有的表空间都是一次性同时备份。可以同时多个表空间置于备份模式。并行备份联机表空间可以减少完成备份所需的时间。Oracle公司推荐使用连续备份选项，因为它可以使ALTER TABLESPACE...BEGIN/END BACKUP语句之间的时间间隔减至最少。忙碌的表空间的并行热备份会产生庞大的重做信息。另外，如果数据库在热备份过程中崩溃了，那么结束一个表空间中部分数据文件的备份要比结束对于包括众多数据文件的多个表空间的备份要快得多、容易得多。

在备份数据库时，可能还希望备份相关的文件，例如口令文件、告警日志、NET8文件以及参数文件。数据库以外的文件可以使用操作系统的备份工具进行备份。在Oracle 9i中，如果采用了新的服务器参数文件特性，则可以使用如下CREATE PFILE命令创建服务器参数文件的一个拷贝：

```
SQL> CREATE PFILE = '/tmp/init.ora' FROM SPFILE;
```

此命令中的init.ora包含一个列出所有服务器参数的文本。

任务4：检查备份命令

查看脚本创建的文件，实际上该文件执行备份命令。在任务2中运行的脚本创建了一个名为/tmp/open_backup_commands.sql的文件。在通过ALTER SYSTEM命令备份数据文件之前，留意提交给重做日志的“注意（attention）”。接下来，文件包含了用于每个表空间的备份命令。在PRACTICE数据库中，除USERS表空间外，其他所有的表空间都只有一个数据文件。当USERS表空间置于备份模式时，需要备份这个表空间的所有数据文件。通过备份表空间中所有的数据文件，可以减少为了各数据文件而将表空间置于或脱离备份模式的工作量。还有，所有的数据文件都拥有同样的备份模式SCN，这使恢复更为简单，更易于管理：

```
alter system switch logfile;
alter system archive log all;
```

```

alter tablespace SYSTEM begin backup;
host cp /oradata/PRACTICE/system01.dbf /oradata/PRACTICE/backup/ch05
alter tablespace SYSTEM end backup;
...
alter tablespace USERS begin backup;
host cp /oradata/PRACTICE/users01.dbf /oradata/PRACTICE/backup/ch05
host cp /oradata/PRACTICE/users02.dbf /oradata/PRACTICE/backup/ch05
alter tablespace USERS end backup;
...
alter database backup controlfile to '/oradata/PRACTICE/backup/ch05/backup.ctl' REUSE;

```

一旦所有数据文件的备份结束，备份命令文件会拷贝一份控制文件。如果丢失了所有的控制文件，则可以使用备份控制文件。

练习5.2：备份归档日志文件

现在，PRACTICE数据库已经正确地创建了许多归档日志文件。请不要忽视这些归档日志文件。当数据库处理事务时，会产生重做日志并归档这些重做日志。经过一段时间，这些重做日志会累积起来，并填满归档路径。重做日志无法归档时数据库活动会中断，这通常都是由于缺乏磁盘空间所致，归档路径必须定期清理。大多数的数据库安装会将归档文件转移至一个磁带备份设备或一台远程机器上，然后磁带脱机存储。如果由于环境灾难（洪水、火灾、地震等）而导致机器损毁，那么配合脱机数据文件和控制文件备份，脱机归档文件可以用于恢复数据库。

本练习中，将把归档文件从当前路径转移到另一路径下。在实际的Oracle数据库配置中，将把文件从一个或多个文件系统移到另一个文件系统、磁带或机器上。

任务描述	时间（分钟）
1. 寻找归档文件	5
2. 创建归档文件脚本	10
3. 运行归档文件脚本	5
4. 确认归档文件备份	5
总计时间	25

首先，找到归档日志文件；然后，创建并运行删除这些文件的一个脚本；最后，确认这个脚本按照期望的那样工作了，并把文件放在了备份路径下。

本练习将验证对一个单独归档目的路径的备份。可能会存在有多个归档目的路径、多个archive_log_dest_n的情况，此处的n是介于1~5之间的数字。如果使用了多个归档目的路径，必须确定每个路径都已经被清空了。从多个目的路径转移（备份）归档日志文件的脚本可以从Osborne Web站点上找到。

任务1：寻找归档文件

归档文件将被存放数据库初始化文件中配置的log_archive_dest参数所定义的路径下。为了找到该参数的值，可以查看参数文件或从v\$archive_dest视图中提取该值。回顾一下第4章中练习4.3的启动归档日志配置详细资料的归档部分，找到重做日志归档的路径定位，打开命令对话框，确认归档重做文件存在。这个路径下的文件应与v\$sarchived_log视图中name列的内容相符。

任务2：创建归档文件脚本

把已经归档的文件从归档路径转移到其他地方，可以使用操作系统的移动（move）命令来完成。在名为archive_backup.sql的文件中，创建所需的命令，生成一个备份命令文件，对于选定的一组归档日志文件，使用移动命令，然后运行命令文件。该文件的重点是：找出v\$archive_log视图中昨日创建的各个归档日志，并用移动命令将这些归档日志转移到一个备份路径下。

```

1.   define dir = '/oradata/PRACTICE/backup/ch5'
2.   define fil = '/tmp/archive_backup_commands.sql'
3.   spool &fil
4.   prompt archive log next;;
5.   SELECT 'host mv '||name||' &dir'
6.     FROM v$sarchived_log
7.    WHERE completion_time >= trunc(sysdate)-1
8.          AND completion_time < trunc(sysdate);
9.   spool off;
10.  @&fil

```

这个脚本文件和我以前给出的类似，这里我再强调几点：

- 第1-2行设置了脚本中后面要使用的SQL*Plus用户变量。注意归档日志将被拷贝到的那个路径保存在dir用户变量中。
- 第3行和第9行分别开始和终止将SQL输出记录到一个文件中。第10行运行spool文件的内容条目。
- 第4行通知服务器对任何尚未归档的联机重做日志文件进行归档。无论配置的是手工归档还是自动归档，这个命令都很重要。PRACTICE数据库已经被配置了自动归档日志文件。如果由于重做活动繁多，需要控制归档，ARCHIVE LOG NEXT命令可以确保在返回到SQL*Plus命令提示符控制之前，所有需要归档的联机重做日志得到归档。请留意该行的末尾处的两个分号，这可以在已被归档的文件中产生一个分号。
- 第5-8行找出控制文件知道的多个已归档日志文件名称，这些文件名称与v\$sarchived_log视图中给出的一样。利用各个文件名称，对新路径会生成一个move命令。这个脚本将转移昨天创建的文件。TRUNC功能删除掉SYSDATE变量的时间（time）部分。这个脚本会转移从昨天00:00:00至23:59:59之间24小时里创建的所有归档日志，由于大多数的恢复将只需要近期的归档文件，所以需要安全地保留当天的归档日志文件。

在典型的产品数据库环境下，归档文件是使用操作系统工具来备份到磁带上的，然后从磁盘上删除这些文件。这些磁带可以脱机保存。只要这些文件安全地保留在磁带上，就可以从磁盘上删除这些文件了。

任务3：运行归档文件脚本

在输入了archive_backup.sql中的内容之后，可以在PRACTICE数据库上以SYS用户身份，从SQL*Plus运行这个脚本。如果不只一次地运行了这个脚本，会得到消息提示：移动命令没有找到文件。这是正常的，因为归档文件早已经被转移了。

注意 你可能希望把归档路径下的所有文件复制到一个备份设备上。在这样一个操作过

程中，会在创建的进程中拷贝一个归档文件。这个局部归档日志文件不能用于恢复，而且还会阻止完全恢复，除非拥有该日志文件的另一个合法拷贝。请留意归档日志文件转移脚本，确定所有的文件都已被拷贝了。

任务4：确认归档文件备份

在命令状态下，检查归档路径，查看哪个文件还保留着。找出归档备份路径，确认文件已经被转移到了合适的位置。查看是否已经拥有了所期望的所有序号值。即使只丢失了一个归档日志文件，也将无法完整恢复数据库。

现在你已经有了一个完整的数据库备份，而且归档文件已被转移了。我们可以检查一下，在出现数据库故障时能否用这个备份进行恢复。

5.2 完全和不完全恢复

在出现故障时，可以用打开的数据库备份并应用重做信息，来完全恢复数据库。当使用用户管理的热备份修复损坏了的数据库时，常常不得不恢复已经还原的数据文件。

练习5.3：从打开的数据库备份中完全恢复

在本练习中，将要删除两个数据文件，并恢复它们，这样数据库就被完全恢复了。本练习看起来与上一章中的第一次恢复很相似，不过，这一次，当破坏并恢复数据库时，将要在数据库打开时完成所有的操作。

任务描述	时间（分钟）
1. 破坏数据库	10
2. 还原丢失的数据文件	10
3. 恢复还原的数据文件	10
4. 确认数据库恢复	5
总计时间	35

这次练习中，将要在数据库打开时对其进行恢复。这个恢复将是完整恢复，不会丢失任何数据库的工作。

任务1：破坏数据库

数据库打开时，在操作系统命令提示符下删除tools.dbf和users01.dbf文件。在Linux环境中，可以在数据文件联机时删除文件；在Windows环境中，需要将数据文件脱机以解除操作系统对文件的锁定。如下所示使用ALTER DATABASE命令将这些文件脱机：

```
SQL> alter database datafile '/oradata/PRACTICE/tools01.dbf' offline;
SQL> alter database datafile '/oradata/PRACTICE/users01.dbf' offline;
```

假如在表空间尚处于联机时就删除了数据文件，那么在数据文件被删除之后的某些时刻，数据库实例会检测到这个（些）文件丢失了。在遇到错误之前，数据文件可能会丢失多次。当这个文件必须被数据库访问时，会产生错误信息“ORA-01157: cannot identify/lock data file”。在Linux环境下，可以尝试从TINA.DATE_LOG表中提取数据文件，如果所请求表的数据块保存在内存中，则SQL语句不会产生什么错误。即使是对表进行更新，短时间内在Linux的数据库上

也不会产生错误，原因是Oracle将数据块放置（暂存）在内存中了。如果所需的数据块事先并不在内存中，系统将从磁盘上读取数据。对于更改过的数据块也是如此。当向磁盘刷新数据块时，数据库实例会发现文件已经不存在了。在Linux环境中，对于丢失的一个数据文件，日志切换引起的检查点会产生这种错误；Windows中的日志切换不会产生错误，因为这两个丢失了的文件不是联机的，在Windows环境中，将不会看到任何错误，除非将文件恢复到联机状态。

```
SQL> alter database datafile '/oradata/PRACTICE/tools01.dbf' online;
SQL> alter database datafile '/oradata/PRACTICE/users01.dbf' online;
```

如果这时试图将数据库正常关闭，则Oracle软件是不允许这样做的。在成功地正常关闭之前，shutdown进程必须成功地完成对所有联机数据文件的一个检查点。如果决定关闭一个缺少了数据文件的数据库，首先必须将这些数据文件脱机或者异常中断这个数据库实例。（我并不推荐使用异常中断数据库实例，数据库异常中断只能作为最后的办法。）当在打开数据库中遇到错误时，请查看告警日志文件，在这个文件中，可以看到与屏幕上类似的这些错误信息：

```
Errors in file /app/oracle/admin/PRACTICE/bdump/practiceDBW0.TRC:
ORA-01157: cannot identify/lock data file 5 - see DBWR trace file
ORA-01110: data file 5: '/oradata/PRACTICE/tools01.dbf'
ORA-27041: unable to open file
OSD-04002: unable to open file
O/S-Error: (OS 2) The system cannot find the file specified.
```

当数据文件不存在时，可以在数据库打开时恢复它。只有在所有所需的归档重做日志可用时，才可以前滚这个数据文件，使之与当前数据库一致。

恢复丢失的数据文件将在本练习其他任务中给出。

任务2：还原丢失的数据文件

使用数据字典，检查是哪个文件需要恢复。可以查看两个视图，确定哪个文件必须恢复：

```
SQL> select * from v$recover_file;
SQL> select name, status from v$datafile;
```

这两个视图都显示哪些文件需要恢复。v\$recover_file视图还让你了解将要碰到的特别错误。此时，ERROR列会显示值为“FILE NOT FOUND”。

当数据文件需要恢复时，数据文件必须脱机。将一个数据文件脱机时，没有任何脱机选项。也可以通过将其表空间脱机的方式把一个数据文件脱机。在将一个表空间脱机时，有四种选择：

- 正常（normal）——当将表空间正常脱机时，它的数据文件经历一个检查点，所有的数据块都从缓冲区流入表空间，数据文件以及文件的头部都以一个检查点SCN被更新。当将这个数据文件恢复到联机时，介质恢复就并不必要了。
- 临时（temporary）——将表空间临时脱机，Oracle对这个表空间内的所有联机数据文件执行一个检查点，但并不确保所有文件都可写。在将表空间恢复到联机时，任何联机文件都可能需要介质恢复。
- 立即（immediate）——立即将表空间脱机，Oracle并不确保表空间文件可用，也不执行一个检查点。在将表空间恢复到联机之前，必须在这个表空间上执行介质恢复。
- 用于恢复（for recover）——选择FOR RECOVER，将恢复设置中的产品数据库表空间脱

机，以便表空间时间点恢复。在第9章中将讨论这一选项。

警告 不能将并非运行于归档日志模式的数据库中的数据文件脱机。必须脱机删除这个文件，然后，还要还原这个表空间。

从打开的数据库备份脚本所使用的备份目的路径下复制users01.dbf和tools01.dbf文件。可以从/oradata/PRACTICE/backup/ch5目录下拷贝文件。在把这两个文件复制回它们原来的数据库位置时，检查v\$recover_file中的内容。注意该视图中的值已经改变，change#列现在有了一个值。对视图的提取过程已经读取了刚刚还原的文件，而且发现了这个检查点SCN。检查点SCN的值就在视图中。将这些数字与v\$datafile视图中的CHECKPOINT_CHANGE#作比较。联机数据文件将拥有比还原的文件更大的检查点数值。V\$datafile视图中包含的检查点信息从控制文件中读取，这与数据文件本身不同。

任务3：恢复还原的数据文件

为了恢复一个表空间或一个数据文件，需要实施完全恢复。首先，通过恢复数据文件恢复TOOLS表空间。使用RECOVER DATAFILE命令，从归档和联机文件向还原数据文件实施重做。可以指示Oracle根据文件号或文件名进行恢复。例如：

```
SQL> recover datafile 5;
```

对于PRACTICE数据库，文件5就是tools01.dbf文件。恢复命令找出还原文件的检查点SCN，将此数值与重做日志文件中的SCN范围进行比较，确定哪个文件需要首先恢复。然后那个文件被从归档目的路径中读出，并被应用于数据文件。可能会看到类似下面的返回信息：

```
ORA-00279: change 83584 generated at 01/05/2002 12:51:17 needed for thread 1
ORA-00289: suggestion : /oradata/PRACTICE/archive/51.arc
ORA-00280: change 83584 for thread 1 is in sequence #51
```

恢复命令确定出恢复所需的重做信息在归档日志序号221~51里。在上一练习中，文件5被转移到备份路径下了。为此，恢复命令必须找出它所需的归档文件。可以将此文件复制到你初始位置，或者可以告诉恢复命令去查看另一个路径。无需复制归档文件，只需指定恢复命令从何处获得归档日志文件：

```
SQL> RECOVER FROM '/oradata/PRACTICE/backup/ch5' DATAFILE 5;
```

如果决定将归档文件复制回归档文件目的路径，不一定要在RECOVER命令中包括FROM子句。对于包含还原文件的整个表空间，可以这样使用RECOVER命令：

```
SQL> RECOVER FROM '/oradata/PRACTICE/backup/ch5' TABLESPACE TOOLS;
```

也可以在同一命令中恢复多个的数据文件或表空间。在RECOVER命令后面写上期望恢复的各个数据文件或表空间，并用逗号分隔：

```
SQL> RECOVER FROM '/oradata/PRACTICE/backup/ch5'
2      DATAFILE '/oradata/PRACTICE/users01.dbf',
3      '/oradata/PRACTICE/tools01.dbf';
```

数据文件恢复完成后，会看到这样一条消息：

```
Log applied.
Media recovery complete.
```


检查告警日志，可以看到恢复过程的时间基线。

```
ALTER DATABASE RECOVER tablespace tools
Wed Jan 05 16:54:55 2002
Media Recovery Tablespace: TOOLS
Media Recovery Start
Media Recovery Log
ALTER DATABASE RECOVER FROM '/oradata/PRACTICE/backup/ch5' CONTINUE DEFAULT
Media Recovery Log /oradata/PRACTICE/backup/ch5/51.arc
...
Recovery of Online Redo Log: Thread 1 Group 1 Seq 14 Reading mem 0
  Mem# 0 errs 0: /oradata/PRACTICE/redo01.log
Media Recovery Complete
Completed: ALTER DATABASE RECOVER tablespace tools
Wed Jan 02 16:55:09 2002
alter tablespace tools online
Wed Jan 02 16:55:09 2002
Completed: alter tablespace tools online
```

在继续最后一个任务之前，试验一下能否跟踪应用于恢复的重做日志文件的痕迹。当数据文件还原后，每个数据文件都包含一个检查点SCN，这个检查点SCN是数据文件备份时记录下来的。这个数字确定了截止到何时，所有已更改的数据块都已被写入了那些文件。这样，在这个SCN之前的任何重做信息对于这个文件来说都不是必需的。每个重做日志文件都包括了一定范围的SCN值，并被分配了一个惟一的序号。含有还原数据文件检查点SCN的重做文件变更，直到那个数据文件必须应用的当前联机重做日志文件，这样重做文件就成为当前的了。

任务4：确认数据库恢复

只要恢复了各个数据文件，就可以使用ALTER DATABASE命令将数据文件置回到联机状态，正如前面讲过的那样。如果数据文件回到了联机状态，就可以确定已经成功地恢复了这些数据文件。除非与控制文件相匹配，否则数据文件不会回到联机状态。

也可以在TINA.DATE_LOG中检查最近的数据库时间。这个时间至少应该是删除TOOLS表空间数据文件之前的时间。最近的时间记录可能会更接近一些，因为Oracle插入行的操作会在表中插入更多的时间项。

技巧 临时表空间无需恢复，因为这个表空间只是作为一个整理空间，可以删除这个表空间，然后另外重新创建一个。除此，不能将临时表空间脱机，但可以将临时表空间中的数据文件脱机。

继续努力吧！你已经从打开的数据库中的热备份中恢复了两个数据文件。

练习5.4：从打开的数据库备份进行不完全恢复

本章最后一个练习将会很有帮助。利用练习5.1中所实施的备份，在“不慎”删除了PRACTICE数据库中的一个表格之后，可以实施不完全恢复。这次练习将重温学过的那些概念，为你提供更多的经验，并让你了解其他一些重要的恢复概念。

任务描述	时间 (分钟)
1. 删除一个表	5
2. 重新命名一个归档日志	5
3. 检查恢复	10
4. 还原数据库	10
5. 将数据库恢复到某一点上	10
6. 验证数据库恢复	5
总共用时	30

本练习中，将删除属于用户SCOTT的一个表，然后从先前的打开数据库备份中还原所有的文件，并利用备份控制文件恢复数据库。

任务1：删除一个表

这个任务中，要删除SCOTT.EMP表。在删除该表之前，记下当前数据库状态，以便此后进行不完全恢复。同时还要记下TINA.DATE_LOG中的最新的date/time值，或向表中插入一个日期，这个日期值在后面的恢复中要能够查到，恢复数据库时，TINA.DATE_LOG中的date/time将是这个日期/时间值（或与之相近的一个值）。此外，切换日志文件，并从v\$datafile中的change#列中提取出数据文件的检查点SCN，记下来。在记下所有这些值之后，就可以用DROP命令来删除表了：

```
SQL> DROP TABLE scott.emp;
```

有些SQL命令可以借助Oracle的回退特性予以撤销。但DROP命令不能被撤销，因为数据定义语句（DDL）已经彻底提交了。一旦某个表被删除了，它就不再存在了。

在开始恢复之前，记下TINA.DATE_LOG中当前的date/time值。一定要在表中插入更多行。甚至可以往表中插入未来的某个时间值。例如向表中插入一年后的一个时间值：

```
SQL> INSERT INTO tina.date_log VALUES (sysdate+(365*5)); SQL> COMMIT;
```

一旦不完全恢复了PRACTICE数据库，将只能看到表中等于或早于删除表之前记录的日期/时间的那些记录。

任务2：重新命名一个归档日志

知道了表被删除的时间，同时日志是在删除操作之前切换的。在最后一章中，不完全恢复是基于时间的；在本章中，练习的是基于取消的一个恢复。当某个归档日志文件损坏或丢失时，常常使用基于取消的恢复。

查询v\$log动态视图，确定出当前日志文件：

```
SQL> SELECT sequence# FROM v$log WHERE status = 'CURRENT';
```

查看归档路径，找出序号小于前面查询结果返回的序号的那些归档日志文件。例如，假设当前日志文件序号为52，应找出/oradata/PRACTICE/archive目录下的一个名为51.arc的文件，将这个文件重命名为51.arc.backup。当在任务5中恢复数据库时，找不到/oradata/PRACTICE/archive/51.arc时，将撤销恢复。

技巧 可以模拟一个被破坏了的归档文件，方法是将任何文件复制到/oradata/PRACTICE/archive/51.arc中（比如：某个字处理文档）。当恢复进程试图读取该归档文件时，将得

到一个Oracle错误信息。

任务3：检查恢复

需要应用各个重做日志，直到发生故障时的日志。考虑一个问题：发生故障时哪个重做日志文件是当前的呢？在上一章中，是从一个删除表空间命令进行恢复的。当表空间被删除后，告警日志会给出删除行为发生的准确时间。删除表命令并不向告警日志中写入记录，因此将不得不估计故障发生的时间。利用这个时间值，可以近似得到删除表时的日志文件序号。

```
SQL> SELECT sequence#, first_change#,  
2         to_char(first_time,'DD-MON-YY HH24:MI:SS') first_time  
3     FROM v$log_history;
```

找出date/time值在故障发生之前的日志序号。

注意 确定数据库故障（用户失误）的时间并非绝对科学。常常无需掌握执行错误DDL命令的准确时间。可能需要从某个近似时间开始着手，在成功之前，可能需要进行多次还原/恢复过程。

任务4：还原数据库

虽然可以使用来自一个打开数据库的文件进行恢复，但整个数据库的不完全恢复必须在一个关闭的数据库上实施，原因是，只要结束了一个不完全恢复，数据文件就会停留在它们原来的某个状态上。当已经还原了一个旧的数据库，而这个数据库需要更新时，DBA的惟一选择就是利用日志文件将数据库前滚。不可以还原一个新的备份，然后再将其回退到某一个未提交的事务。

因此，为了重新获得SCOTT.EMP表，必须用删除表之前所做的拷贝取代所有的数据文件。然后把日志应用于所有的数据文件，直到表被删除前的某一时间点。还要注意的是不要从当前数据库联机重做日志文件上所做的备份中复制日志文件，联机重做文件可能会含有在不完全恢复期间希望应用的事务。

在还原数据文件之前，首先要正常关闭数据库。在实际的恢复情况下，如果时间允许，则实施整个数据库的一个关闭备份（以防恢复过程不能按要求实施）。还原练习5.1中所创建的所有数据文件，并将这些文件存放在当前数据库目录下。当数据库打开时，将用所做的备份文件覆盖PRACTICE数据库的数据文件。

任务5：将数据库恢复至某一时间点

在数据文件上实施恢复操作，当数据库在使用时，查看v\$log_file中的内容。将看到所有的数据文件都需要恢复。注意，除了USER表空间的两个文件外，其他所有的文件都拥有不同的更改号，原因是这些数据文件是在打开数据库备份期间拷贝的。这些表空间被置于备份模式，并在不同的时间被拷贝。它们都需要从其当前的SCN恢复到将要应用的最新重做日志的结尾SCN。

注意 当使用热备份进行不完全备份时，必须确保将要恢复到的时间涵盖了最近热备份的结束标记。否则，将试图去打开一个拥有不一致的数据文件的数据库，而这时无法打开这个数据库。最近终止备份命令的时间应当被记录在alert.log中。

在此处的恢复中，删除表之前最近的当前日志文件序号是50。重做日志文件中的更改号在

83821~83850之间。当我实施恢复时，我会持续应用恢复命令所建议的日志，直到到达日志序号51，然后撤销操作。

```
SQL> SET LOGSOURCE "/oradata/PRACTICE/backup/ch5";
```

此前，指定了恢复命令使用某个非缺省路径。也可以在RECOVER命令之前用SET LOGSOURCE命令指定归档日志的位置。

开始撤销恢复。保留所应用的重做日志，直到最后一个日志也被应用了。使用重置日志选项打开数据库：

```
/oradata/PRACTICE/backup/ch5/51.arc
ORA-00280: change 83851 for thread 1 is in sequence #51
ORA-00278: log file '/oradata/PRACTICE/backup/ch5/50.arc'
no longer needed for this recovery
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

在此命令提示符下，输入“CANCEL”一词。现在从另一个独立的SQL*Plus会话框查看v\$recover_file表。注意每个数据文件的change#和time列都被提升了。通过应用重做日志文件，已经将所有数据文件的内容都提升到某个一致的时间和更改号上。所有这些数据文件的更改号以及时间都将相同，并等于所应用的最近日志文件的最新更改号和时间。

为了打开数据库，必须在ALTER DATABASE OPEN命令中指定RESETLOGS。联机重做日志文件将以数字1为序号开头。除此，当数据库开始归档时，归档上的序号将从1重新开始，这是因为这些归档文件是联机重做日志的普通文件拷贝。

注意 由于归档日志文件现在使用的序号是从1开始计数的，所以应当在打开数据库之前清除当前归档日志目的路径。这样能够避免来自不同数据库实体的文件保存在同一目录下时产生冲突。

警告 不能使用以前的备份，然后通过RESETLOGS打开方式将其前滚。当数据库是以RESETLOGS方式打开时，最近的当前联机重做日志中未提交的数据会丢失。因此，恢复过程也有可能丢失重做信息。必须在数据库采用RESETLOGS参数打开时对其进行备份。这个备份将成为将来所有恢复的最基本的起点。

任务6：确认恢复

实施这个恢复的前提是替换丢失了的表。查看并确认SCOTT.EMP表存在。从DATE_LOG中提取出表被删除前的最近的时间与日期。如果在EMP表被删除之后插入未来5年后的某一个日期/时间值，这个日期/时间值不会保存在该表中。

5.3 疑难解答

通读本章，掌握一些技巧，这些技巧可以应付打开的数据库备份、归档操作和恢复过程中遇到的那些可能的错误和情况。

5.3.1 归档命令

当进行配置或使用归档时，可能会遇到下面这些问题：

归档日志列表 (Archive Log List)

当运行这一命令时，可能会遇到如下错误：

```
SQL> archive log list  
ORA-01031: insufficient privileges
```

必须以SYSDBA或SYSOPER身份连接数据库才能使用该命令。

归档位置 (Archive Locations)

在Windows中，可能会注意到归档日志文件并不存在于你认为它们应该在的地方。例如，可能已经设置了log_archive_format = %s.arc和log_archive_dest_1=“location=D:\oradata\PRACTICE\archive”，但已归档日志文件却在d:\oradata\PRACTICE目录下，名称为archive%s.arc。请确认已经在d:\oradata\PRACTICE目录下创建了归档路径。如果没有，那是因为文件与路径联在一起了。

5.3.2 启动或关闭命令

在数据库启动时，可能会遇到以下这些可题。

1) ORA-01113: file 3 needs media recovery

可能有某个文件与数据库并不同步。除此，也可能是在某个表空间处于备份模式时数据库崩溃了。尝试使用recover database命令来恢复数据库。恢复进程将把那个表空间脱离备份模式。

2) ORA-01123: cannot start online backup; media recovery hot enabled

必须在归档日志模式中，才能运行打开数据库备份。

3) ORA-01142: cannot end online backup -none of the files arc in backup mode

如果试图将一个并非处于备份模式的表空间脱离备份模式时，会出现这个错误。

4) ORA-01078: failure in processing system parameters

查看参数文件中请求的字符串是不是没有以“\”结尾。在Windows环境中“\”符号是作为目录分隔符使用的，但是在Oracle中却是作为字符串扩展符使用的。

在数据库关闭时，可能会遇到下面这个错误。

5) ORA-01149: cannot shut down-file 3 has online backup set

如果某个表空间尚处于备份模式时就试图关闭数据库，将会出现这个错误。请在关闭数据库之前将表空间脱离备份模式。

5.4 小结

在数据库打开时，可以对Oracle数据库实施部分或完整的备份。在此过程中，拷贝表空间中的数据文件之前，将表空间置于备份模式。一旦完成复制，就将表空间从备份模式还原出来。在连同热备份一起备份控制文件之前，一定要切换日志文件。利用这些备份文件，可以恢复数据库、表空间或数据文件。

测试备份可用性的最佳途径是将它们在另一个独立的主机上还原，并尝试打开数据库，如有必要再实施介质恢复。这种测试策略要求有一台独立的计算机用于还原和恢复数据库。

在本书后面，将看到Recovery Manager（恢复管理器）是如何简洁、高效地控制数据库热备

份的。当数据库打开时利用Recovery Manager创建一个备份，无需将表空间设置为热备份模式。

要想获得关于打开数据库备份和恢复的更多信息，请查阅《Oracle 8i Backup and Recovery Guide》的第3章到第6章以及《Oracle 8i Administrators Guide》的第3、6、7章。

习题

以下的问题用于帮助复习本章中的一些重要概念。

1. 某个处于备份模式的联机表空间中的数据文件并不发生变化，或由数据库写进程用更改过的数据块予以更新。

- A. True
- B. False

2. RECOVER命令是如何知道哪个重做日志文件要被用于恢复的？（选择所有正确答案）

- A. 恢复是从含有最小的SCN的数据文件开始的。
- B. Oracle从控制文件获知重做日志文件的变更范围。
- C. Oracle从控制文件获知所需的重做日志文件的位置。
- D. 不清楚。

3. 为什么要清理归档日志目的路径中已经归档的重做日志文件？（选择所有正确答案）

- A. 这些已经归档的重做日志文件与数据库参数文件冲突了。
- B. 将它们脱机以防不测。
- C. 这样归档目的路径有空间容纳更多的归档文件。
- D. 防止一个新数据库的文件覆盖已有的归档文件。

4. 利用哪个视图可以确定需要恢复什么数据文件？

- A. v\$logfile
- B. v\$datafix
- C. v\$controlfile
- D. v\$recover_file

5. 应用哪个命令可以将TOOLS表空间置于备份模式，从而可以在操作系统拷贝表空间中的文件？

- A. ALTER TABLESPACE TOOLS YOU CAN COPY IT NOW
- B. ALTER TABLESPACE TOOLS SET BACKUP
- C. ALTER TABLESPACE TOOLS BEGIN BACKUP
- D. ALTER TABLESPACE TOOLS BACKUP BEGIN

答案

1. B。在热备份模式中，数据文件块被持续不断地读出和写入，数据文件首部和重做内容的控制不一样。

2. A、B、C。RECOVER DATAFILE命令找出拥有最早的检查点SCN的数据文件，利用

这个SCN，RECOVER DATAFILE命令找出包含了变更范围的重做日志文件名称，并开始将这些重做日志应用于数据文件。控制文件对重做日志文件变更号、时间、序号以及文件名称保持跟踪。

3. B、C、D。当数据库产生重做信息并予以归档时，归档目的路径将会塞满。当归档目的路径塞满时，数据库操作中止。假如丢失了数据库计算机，可以用脱机备份（包括重做日志）恢复数据库。最后，假如保持归档文件的目的路径和格式一样的话，新产生的归档信息会覆盖任何已有的归档日志文件。

4. D。v\$logrecover_file中说明哪个数据文件与其他的数据文件以及控制文件不一致。

5. C。当使用ALTER TABLESPACE TOOLS BEGIN BACKUP命令时，TOOLS表空间经历一个检查点。对该表空间中文件所做的所有更改都被作为一个块映像记录在重做文件中。



第6章 复制数据库

当系统停机和用户正在等待时，常常就不必进行数据库恢复。事实上一些恢复可以预先做好计划。我认为规划一个恢复工作有三点原因：

- 为了检验备份是完整的。
- 为了锻炼数据库管理技能。
- 为了创建一个已有数据库的复制副本。

在本章中，我们将讨论创建一个复制数据库。

复制数据库可以被称为副本或克隆数据库。数据库的副本名称与原数据库相同，克隆数据库则与原数据库名称不同。当同时创建一个副本和一个克隆时，数据库将被物理地复制到相同的机器或不同的机器上。运用前面已经讲过的还原和恢复技术，可以很简单地复制一个数据库。

通过实施数据库复制操作，能够为自己以及开发小组带来以下好处：

- 开发小组拥有一个用于编程和测试的模拟数据库。
- 可以验证当前数据库备份的质量。
- 可以衡量、验证备份和恢复存储过程。
- 可以练习数据库恢复技能。

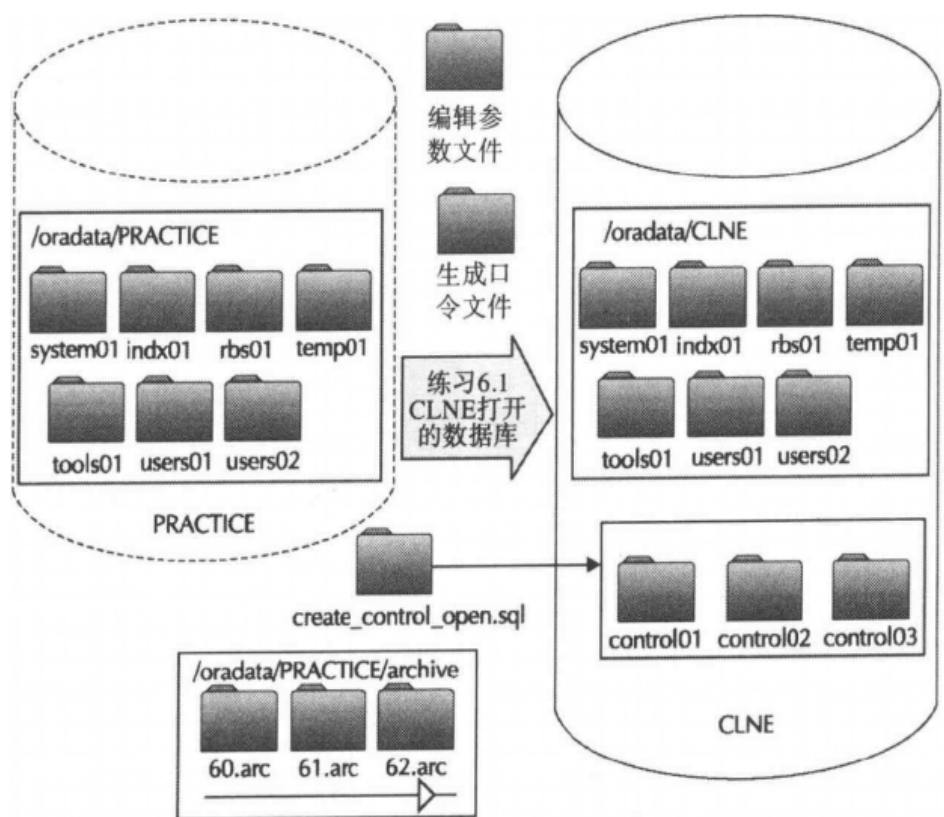


图6-1 数据库克隆练习

利用Oracle的工具，可以用三种方式复制或克隆数据库。首先，可以用操作系统的文件拷贝进行物理数据库的还原和恢复，并重新创建数据库控制文件。其次，可以用Oracle Export和

Import工具将原数据库逻辑还原到一个新的数据库（参见第8章）。最后，可以用Oracle的恢复管理器进行数据文件和控制文件的一个物理文件还原（参见第15章）。在本章中，将在同一台机器上运用一个打开数据库备份来克隆PRACTICE数据库（参见图6-1）。然后我们再修改章节练习中介绍的存储过程，以实践其他的复制场景。

复制数据库看上去很简单，就像复制一组字处理文档一样。但不幸的是，实际上并没有那么简单。切记一个Oracle数据库是由三个文件类型的组合构成的。这三个类型（数据文件、重做日志文件和控制文件）必须协调一致。当克隆一个数据库时，将数据文件复制到新的数据库位置，创建新的控制文件，并重新设置重做日志。前面已经学习了如何备份（拷贝）数据文件和重置重做日志。

复制数据库可以从一个主数据库创建，无论这个数据库是打开的还是关闭的。在第4、5章中，学习了关闭（冷）备份和打开（热）备份。为了创建一个复制数据库，来自打开和关闭备份的备份文件可被用作数据文件副本。关闭备份的数据文件备份不需要恢复。打开备份的数据文件需要恢复，因为各个文件之间相互不一致。当克隆数据库时，将必须创建一个新的控制文件。让我们了解一下CREATE CONTROLFILE命令的细节情况。

数据库控制文件可以用CREATE CONTROLFILE命令来创建。很多情况下，这个命令都是有用的或者必要的，列举如下：

- 1) 丢失了所有的控制文件。
- 2) 需要更改重做日志成员或组的最大设置。
- 3) 必须更改数据文件或实例的最大序号。
- 4) 希望更改数据库文件的名称和位置（当然也可以在数据库打开时通过其他办法实现）。
- 5) 希望更改数据库的名称。

在给出CREATE CONTROLFILE命令语句之后，Oracle根据语句中给定的信息创建一个新的控制文件。

CREATE CONTROLFILE与CREATE DATABASE语句有点类似。可以参考表6-1中关于该命令中部分关键字和子句的描述。我已经标出了适于数据库复制的参数。

表6-1 对于创建新的控制文件比较重要的关键字

关键字	是否需要	描 述
LOGFILE	是	指定所有将被数据库使用的重做日志成员和群组。如果计划重新设置重做日志，这些文件就无需存在了，它们将被创建。如果计划再利用已有的重做日志，必须正确指定每个重做日志文件
DATAFILE	是	列出数据库中指定的每个数据文件的文件名称
REUSE	否	当创建一个控制文件，同时又有与它名称相同的文件，会得到一个错误信息。利用REUSE参数来覆盖已有的控制文件。控制文件的位置由数据库参数文件中的control_files参数来确定
SET	否	定义出所希望的数据库名称。当重新为数据库命名时需要该选项
RESETLOGS	否*	指出正在忽略和取代LOGFILE关键字后面跟着的任何日志文件。如果在创建控制文件时使用这一选项，必须使用RESETLOGS选项打开数据库
NORESETLOGS	否*	指定从数据库打开时起的重做日志文件将被使用。LOGFILE参数指定的重做日志文件必须存在，创建的控制文件将参考当前的数据文件

* RESETLOGS或NORESETLOGS必须被指定。

创建控制文件的注意事项

- 1) 数据库实例必须启动, 但没有加载。如果执行成功, CREATE CONTROLFILE语句则加载新创建的控制文件。
- 2) 运行该命令的用户必须被赋予OSDBA角色。
- 3) 当创建一个新的控制文件时, 会丢失所保存的历史资料, 例如归档日志历史和RMAN备份。

警告 当数据库是以RESETLOGS选项打开时, 重做日志文件将被重新设置, 而且数据库被认为是一个新的实体。实体意味着以新的形式出现、刷新, 或重新生成。每次数据库以重新设置日志方式打开时, 都会建立数据库的另一个实体。请记住打开数据库和重新设置日志时需要慎重考虑很多因素。

不能用一个以前的实体备份和后来的实体重做日志来恢复数据库。尽量避免重新设置日志(创建一个新的数据库实体)。如果必须重新设置日志, 那么在数据库打开时就备份数据库。

6.1 数据库复制

本章中有一个练习, 将要在同一机器上克隆PRACTICE数据库。将学习如何使用CREATE CONTROLFILE命令克隆数据库。对PRACTICE数据库的克隆将在数据库打开时创建。完成此练习后, 我们将学习其他一些可以实施的数据库复制方案。在本章学习的基础上可以完成实际的复制任务。

练习6.1: 克隆打开的PRACTICE数据库

为了创建打开的PRACTICE数据库的一个克隆, 遵循以下步骤。完成任务的时间可能会比这里给出的少。对于每一步我都给出了充足的时间, 并假定这个时间是完成各步的首次时间, 应尽力在这些时间内完成任务。

任务描述	时间(分钟)
1. 准备克隆数据库	10
2. 备份打开的PRACTICE数据库	10
3. 配置控制文件脚本	10
4. 运行控制文件脚本	5
5. 恢复克隆的数据库	10
6. 打开克隆的数据库	5
总计时间	50

首先, 将为一个名为CLNE的新数据库准备路径、参数文件和口令文件(以及Windows服务)。接下来, 把PRACTICE数据文件复制到CLNE数据库的数据文件目录下。然后将生成为CLNE数据库创建一个新的控制文件所需的命令脚本。最后, 运行这个命令脚本完成CLNE数据库。

任务1: 准备CLNE数据库

创建数据库路径、参数文件和以及口令文件（以及Windows服务），准备好CLNE数据库。如果在第3章中使用了Database Configuration Assistant（数据库配置助手）来创建PRACTICE数据库，那么下面也将使用这个工具完成部分任务。

为准备建立CLNE数据库创建所有路径：

```
export ORACLE_SID=CLNE
export ORACLE_BASE=/app/oracle
export ORACLE_HOME=/app/oracle/product/8.1.7
export ORACLE_DATA=/oradata/$ORACLE_SID
export ORACLE_ADMIN=$ORACLE_BASE/admin/$ORACLE_SID
mkdir $ORACLE_ADMIN
mkdir $ORACLE_ADMIN/pfile
mkdir $ORACLE_ADMIN/bdump
mkdir $ORACLE_ADMIN/cdump
mkdir $ORACLE_ADMIN/udump
mkdir $ORACLE_ADMIN/create
mkdir $ORACLE_DATA/
mkdir $ORACLE_DATA/archive
```

可以用这些命令在以后的章节中创建数据库路径结构，只需改变ORACLE_SID环境变量即可。

接下来，为CLNE数据库创建一个参数文件。把PRACTICE数据库的参数文件拷贝到\$ORACLE_BASE/admin/CLNE/pfile目录下。同时，添加一个与\$ORACLE_HOME/dbs目录下的默认参数文件的符号链接。在Linux上可以这样使用下列命令：

```
LINUX> cp $ORACLE_BASE/admin/PRACTICE/pfile/initPRACTICE.ora
$ORACLE_BASE/admin/CLNE/pfile/initCLNE.ora
LINUX> ln -s $ORACLE_BASE/admin/CLNE/pfile/initCLNE.ora
$ORACLE_HOME/dbs/initCLNE.ora
```

\$ORACLE_HOME/dbs路径中的符号链接向OFA（Oracle Flexible Architecture—Oracle柔性结构）定义的pfile位置指出默认参数文件的所在位置。在Windows环境中，没有Linux环境中的这种符号链接。可以在默认参数文件名中包括另外一个参数文件。在Windows中，把PRACTICE数据库的参数文件拷贝到CLNE所在的位置，并在默认数据库中创建一个新的文件，包括IFILE关键字，命令如下：

```
WINNT> copy %ORACLE_BASE%\admin\PRACTICE\pfile\initPRACTICE.ora
%ORACLE_BASE%\admin\CLNE\pfile\initCLNE.ora
WINNT > notepad $ORACLE_HOME\dbs\initCLNE.ora
> IFILE=d:\app\oracle\admin\CLNE\pfile\initCLNE.ora
```

对CLNE数据库设置了参数文件后，打开pfile目录下的文件。将文件中出现的所有“PRACTICE”换成“CLNE”一词。当进行这项全面搜索和替换时，将改变数据库的名称、服务名称、控制文件位置，以及参数文件的许多目录。

由于改变了数据库名称，可能还希望将这个文件中的其他参数从旧的数据库名PRACTICE改变为新的数据库名CLNE。更改方法如下：

```

db_name = CLNE
instance_name = CLNE
service_names = CLNE
log_archive_dest_1 = "location=/oradata/CLNE/archive"
background_dump_dest = /app/oracle/admin/CLNE/bdump
core_dump_dest = /app/oracle/admin/CLNE/cdump
user_dump_dest = /app/oracle/admin/CLNE/udump
control_files = ("/oradata/CLNE/control01.ctl",
                 "/oradata/CLNE/control02.ctl",
                 "/oradata/CLNE/control03.ctl")

```

在改变控制文件所在位置时要非常小心。你并不希望覆盖当前的控制文件。CREATE CONTROLFILE命令会在CONTROL_FILE参数指定的文件目录下创建控制文件。

如果试图用一个已有的文件名创建一个控制文件，CREATE CONTROLFILE命令将不会执行。这可以防止错误地覆盖一个已经存在的控制文件。如果就是要求这样做，可以在创建控制文件时添加REUSE关键字以避免这种错误。注意：如果没有改变init.ora文件，这样做会覆盖某个控制文件。

创建一个控制文件时，可能会创建同一个控制文件的多个副本。在本示例中，将要在/oradata/CLNE目录下创建3个控制文件。在实际情况下，这3个文件应当保存在不同的物理空间上。

创建一个口令文件，这样就可以在SQL*Plus中以SYSDBA角色的SYS连接数据库。这个口令文件还用于远程管理安全。命令如下：

```
LINUX> orapwd file=$ORACLE_HOME/dbs/orapwCLNE password=CLNE entries=4
```

在Windows中，需要为CLNE数据库创建一个新的数据库服务。这会创建一个名为OracleServiceCLNE的Windows服务，使用的是CLNE内部口令。

```
WINNT> oradim -new -sid CLNE -intpwd CLNE
```

任务2：备份打开的PRACTICE数据库

还记得在第5章的练习5.1中所建立的备份脚本吗？你可以再次使用这个脚本，在PRACTICE数据库打开时，把数据文件拷贝到CLNE数据目录下。找到名为open_backup.sql的文件，如下所示，做部分精细调整：

1) 将dir用户变量从当前的值改变为/oradata/CLNE。当运行打开数据库备份时，将把所有的数据文件拷贝到复制数据库的目录下。

2) 删除用于创建控制文件复制命令的行。当复制数据库时，将用新的数据库名称创建一个新的控制文件。

编辑好open_backup.sql数据库文件之后，在PRACTICE数据库上以SYSDBA角色的SYS运行这个文件。如果执行完成，/oradata/CLNE目录下将有PRACTICE数据库中所有数据文件的一个非一致副本。

在这个简单练习中，只有一个路径。在一个包括许多大文件的实际系统中，可能需要编辑复制脚本，以便将文件存放在不同的路径或磁带上。

任务3：配置控制文件脚本

输入CREATE CONTROLFILE命令所需的全部文件需要一定的工作量。但是有一条捷径可

以方便地完成该工作。可以让数据库产生创建一个新的控制文件所需的所有指令，用alter database命令完成这些工作。打开一个SQL*Plus对话框，以SYS用户登录PRACTICE数据库：

```
SQL > ALTER DATABASE BACKUP CONTROLFILE TO TRACE RESETLOGS;
```

使用这个命令时，Oracle服务器会在服务器机器上产生一个包含了创建控制文件所需的全部命令的文件。文件的位置由user_dump_dest初始化参数确定。

事实上，这个文件包含了一个完整的脚本，可以修改这个脚本来复制数据库。为了寻找能够找到跟踪文件的路径，使用以下这些命令：

```
SQL > SELECT value FROM v$parameter
2 WHERE name = 'user_dump_dest';
SQL > SHOW PARAMETER user_dump_dest;
```

这个文件可能以oral23.trc之类的名称命名。寻找一个日期/时间值与提交跟踪命令时刻相同的文件。当看到这个跟踪文件时，会发现创建一个控制文件和启动数据库所需的全部命令。这个文件包括了STARTUP、CREATE CONTROLFILE、RECOVER DATABASE以及ALTER DATABASE OPEN等命令。

如上所述，在user_dump_dest目录下找到该文件。找到该文件后，将它重新命名为create_control.sql，并把它转移到\$ORACLE_BASE/admin/CLNE/create目录下。文件名为create_control.sql。

这个跟踪文件具有创建新的CLNE数据库的一个新控制文件所需的全部命令。只需做少量改动，就可以运行这个脚本来终止数据库克隆进程。改动如下：

1) 删除脚本中的注释行。文件的顶部包括了创建控制文件和打开数据库都不需要的跟踪信息。还有，删除以#符号开头的行。

2) 将CREATE CONTROLFILE命令中的REUSE关键字换成SET关键字。改变数据库的名称，而不保留原来的数据库名称。

3) 将CREATE CONTROLFILE命令中的“PRACTICE”换成“CLNE”。所创建的控制文件将被用于名为CLNE的数据库。

4) 将每个数据文件和重做日志的所在位置从/oradata/PRACTICE目录下换到/oradata/CLNE目录下。

完成以上修改之后，文件看起来可能会如下所示：

```
1.  STARTUP NOMOUNT
2.  CREATE CONTROLFILE SET DATABASE "CLNE" RESETLOGS ARCHIVELOG
3.      MAXLOGFILES 16
4.      MAXLOGMEMBERS 2
5.      MAXDATAFILES 30
6.      MAXINSTANCES 1
7.      MAXLOGHISTORY 226
8.  LOGFILE
9.      GROUP 1 '/oradata/CLNE/redo03.log' SIZE 1M,
10.     GROUP 2 '/oradata/CLNE/redo02.log' SIZE 1M,
11.     GROUP 3 '/oradata/CLNE/redo01.log' SIZE 1M
```

```

12.  DATAFILE
13.      '/oradata/CLNE/system01.dbf',
14.      '/oradata/CLNE/tools01.dbf',
15.      '/oradata/CLNE/rbs01.dbf',
16.      '/oradata/CLNE/temp01.dbf',
17.      '/oradata/CLNE/users01.dbf',
18.      '/oradata/CLNE/indx01.dbf',
19.      '/oradata/CLNE/users02.dbf'
20.  CHARACTER SET WE8ISO8859P1;

```

新的控制文件需要知道各个数据库文件都被拷贝到什么地方了。CREATE CONTROLFILE 命令并不关心每个数据库文件的所在位置。由于控制文件只需找到各个文件，它无需关心文件名称或文件的所在位置。

- 第1行启动数据库实例，并打开控制文件。可以用PFILE（PFILE代表参数文件）选项指定数据库参数文件。如果不指定参数文件，Oracle服务器会根据ORACLE_SID设置以及操作系统打开某个默认位置下的一个文件。在NOMOUNT启动状态时，只需阅读init.ora文件，并建立Oracle所需的内存结构。
- 第2行创建控制文件。当创建自己的控制文件时，将会重新命名数据库。控制文件中的数据库名称最初与各个数据文件中的数据库名称不同。由于不能改变重做日志中的数据库名称，因此这些重做日志必须重新创建。只要重新设置日志，使之包括了新的数据库名称，原数据库的联机重做日志和归档重做日志就都不能再使用了。为了改变控制文件中的数据库名称，可以在CREATE CONTROLFILE命令中使用SET和RESETLOGS关键字。SET关键字指出控制文件将拥有一个与当前名称不同的名称。本示例中指定的数据库名称为CLNE，而不考虑数据文件首部会认为它们属于哪个数据库。控制文件中的RESETLOGS选项意味着对重做日志的描述将被用于创建新的重做日志。
- 第8~11行用数据库名称创建控制文件、创建重做日志文件和群组、创建数据文件以及一个字符集。联机重做日志文件由LOGFILE关键字后面的文本决定。当数据库RESETLOGS选项打开时，这些文件将被创建。
- 第12~20行定义了组成数据库的那些数据文件。

你已经删除了恢复命令和打开数据库的命令。现在只能实施基于取消的恢复，这种恢复过程必须交互完成，接下来会看到具体的过程。

任务4：运行控制文件脚本

在SQL*Plus中连接数据库，并运行\$ORACLE_BASE/admin/CLNE/create目录下的create_control_open.sql脚本，实际创建一个新的控制文件：

```

LINUX> export ORACLE_SID=CLNE
LINUX> cd $ORACLE_BASE/admin/$ORACLE_SID/create
LINUX> sqlplus /nolog
SQL> connect sys/CLNE as sysdba;
SQL> @create_control_open.sql

```

注意：CREATE CONTROLFILE脚本可以在任何地方通过一台客户机上的Net8 SQL*Plus会

话运行。

任务5：恢复克隆数据库

现在，已经为CLNE数据库创建了一个新的控制文件，对于从打开的PRACTICE数据库复制而来的各个数据文件都需要应用重做信息。用于CLNE数据库的数据文件需要恢复，因为它们是在PRACTICE数据库打开时拷贝的。当表空间被置于备份模式时，数据文件的检查点SCN是不同的，因此，每个数据文件都必须用来自PRACTICE数据库的归档日志进行恢复。

从关闭备份克隆数据库

如果从一个关闭备份克隆数据库，在创建新的控制文件之后和打开数据库之前，将无需恢复数据文件，原因是当数据库关闭时，所备份的数据文件是相互一致的，因此，没必要进行恢复。如果选择从关闭备份中克隆数据库，可遵照本练习中除恢复任务以外的所有步骤。

当应用重做日志时，可以实施完全恢复或不完全恢复。无论哪种恢复，都需要使PRACTICE数据库的归档日志对于CLNE数据库可用，使用下面的命令：

```
SQL> SET LOGSOURCE /oradata/PRACTICE/archive;
```

这个命令通知CLNE实例在PRACTICE数据库的归档目的路径中寻找用于恢复的归档日志。

准备开始恢复了。采取基于取消的恢复是明智的选择。如果有许多归档日志文件需要恢复，可以采取基于SCN或基于时间的恢复，同时应用各个重做日志文件，以节省时间和精力。可以逐一提供归档日志文件，直至撤销或介质恢复完成。

```
SQL> RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL;
```

在提示符下，应用至少一个归档日志文件。每个数据文件的结束备份重做记录可能会被包含在该重做文件中。在应用了这个重做文件的基础上，所有的数据文件都将被提升到同样的SCN。只要应用了足够的重做信息将能把数据库前滚到所期望的时间点上，按下CANCEL键。为了打开CLNE数据库，应用所请求的所有归档日志，包括含有来自打开数据库备份的结束热备份记录的日志。如果希望在打开CLNE数据库之前完全恢复它，将不得不用PRACTICE数据库的当前联机重做日志进行恢复。实施恢复，直到用完了要应用的归档重做日志。一旦来自PRACTICE数据库的所有归档重做日志都已被应用，就有如下两个选择：

1) 在PRACTICE数据库上强制日志切换，并将新归档的日志应用于CLNE数据库。在应用完后，将会把日志切换之前对PRACTICE数据库所做的全部改变应用到CLNE数据库上。在CLNE数据库恢复提示中，按CANCEL键。这样就可以打开CLNE数据库了。

2) 关闭PRACTICE数据库，把PRACTICE数据库的当前重做日志应用到CLNE数据库上。在PRACTICE数据库关闭时，在CLNE数据库恢复提示中键入/oradata/PRACTICE/redo01.dbf（假设redo01.dbf是当前联机文件）。还可以把当前联机重做文件复制到为CLNE数据库准备的一个地方以便读取。如果已经完全恢复了CLNE数据库，将看到消息“MEDIA RECOVERY COMPLETE”。由于数据库是不断地被写入当前联机日志的，所以在PRACTICE数据库打开时不要尝试应用PRACTICE数据库的当前重做日志。

额外的信任

对于数据文件，在DATAFILE子句中有许多灵活选择，列举如下：

- 1) 文件名可以连同文件路径一起改变。例如，文件INDX01.dbf可以被拷贝给INDX01.ora（或选择的其他任何名称）。
- 2) 对于CREATE CONTROLFILE命令，数据文件和重做日志没有必要保存在同一路径下。
- 3) DATAFILE关键字后面的文件顺序可以改变。例如，tools01.dbf文件可以在第一位，而system01.dbf放在最后。

为什么可以这样呢？每个数据库文件在其首部都有惟一的数据库信息，可以用于识别该文件。CREATE CONTROLFILE命令确认该文件存在以及数据库名称与新控制文件的数据库名统一。每个数据文件头部都被读取并存入新的控制文件中。这就是为什么你可以改变文件名、路径以及文件名顺序的原因。

对于额外的信用，可以更改文件名称、路径和顺序。注意，一旦打开数据库，从v\$datafile字典视图获得的文件号在复制数据库和原数据库中是一样的。（我不建议改变数据文件的顺序，但可以试一试，以帮助你更好地理解复制进程。）

任务6：打开克隆数据库

在看到介质恢复被取消或完成的消息之后，准备打开数据库，使用RESETLOGS选项。为了使这个命令工作，所有的数据文件必须一致，在每个文件的头部拥有相同的检查点SCN：

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

好极了！没有关闭PRACTICE数据库就创建了一个克隆。

6.2 复制数据库场景

在完成本章的练习后，就可以准备其他类型的数据库复制。在结束本章之前，我想提供一些技巧和窍门，以完善刚刚学到的内容。

6.2.1 克隆到其他机器上

典型情况下，数据库复制并不出现于同一机器上。例如，可能创建了一个TEST数据库作为PRODUCTION数据库的一个副本。你可以遵循以下步骤，但请记住文件必须从源数据库机器上转到复制数据库机器上。应该谨记：

- 1) 在不同的机器之间克隆数据库时数据库版本必须相同。如果Oracle版本以及补丁版本都相同的话（8.1.7.x.x），可以减少出现问题的可能性。
- 2) 数据库复制只能在使用同样类型和版本的操作系统的机器之间进行。例如，把Linux中的数据库复制到Windows机器上是行不通的。
- 3) 当在其他机器上克隆数据库时，某些任务将在原来的数据库上执行，另外的任务则要在克隆数据库机器上完成。在源数据库机器上，创建一个控制文件跟踪脚本，将数据文件备份到磁盘或磁带上，并把数据文件、参数文件、控制文件脚本和归档日志拷贝到克隆

数据库机器上。在克隆数据库机器上,准备好数据库实例(创建路径、编辑参数文件、建立窗体服务、创建口令文件),然后将数据文件复制到指定的路径下,编辑控制文件创建脚本以指向那些文件的所在位置。最后,创建一个新的控制文件,用复制的归档日志恢复数据库。

- 4) 当创建了一个备份并需要转移文件时,必须在不同的机器之间传输文件。FTP工具是将文件从一台机器拷贝到另一台机器的通用方法。FTP(文件传输协议)是主要在互联网上使用的文件传输机制。Linux和UNIX机器都将FTP作为标准工具。记住,将传输模式设为二进制。Windows NT/2000上也有FTP可用。可以在两台NT/Windows 2000服务器之间设立一个网络驱动。用于传输文件的其他UNIX命令还有RCP(远程复制)。
- 5) 任何数据库复制都需要为复制机器配置Net8。你并不希望在连接复制数据库时错误地连接了源数据库。通过查询v\$database和v\$instance视图,可以确认连接的数据库和实例。
- 6) 如果数据库文件非常庞大,而网络连接又比较慢,可以在源数据库机器上压缩数据库文件,转移这些文件,然后在复制数据库机器上解压缩这些文件。这样可以节省大量时间。UNIX文件压缩工具包括compress/uncompress和gzip。Windows下的文件压缩工具有PkZip和WinZip。Oracle不支持任何形式的文件压缩。在文件复制操作中采用压缩技术时,可能会导致数据块损坏。

6.2.2 从磁带备份克隆

由于数据库可能备份到磁带上,因此可以从同一磁带上还原数据库。在源数据库备份中创建的备份集可以用于复制数据库。利用当前备份进行复制的好处就在于能够确保磁带备份处于良好状态。如果利用磁带备份复制数据库但失败了,说明应该在备份中寻找问题。

6.2.3 向其他机器拷贝数据库

当从一台机器向另一台机器上拷贝一个数据库时,对于复制数据库,可能使用(或不使用)现有的控制文件。没有重新创建控制文件而拷贝一个数据库是可能的。如果是以下情况则可以实现这一点:

- 1) 数据库副本将使用同样的名称,但不在同一机器上。
- 2) 把所有的数据文件、联机重做文件和临时文件复制到了同一路径和文件系统名下,或者重新命名数据文件,并加载了拷贝的控制文件。

为了拷贝一个数据库而不重新创建控制文件,可以在目标机器上创建一个与源机器上一样的目录结构。将所有的数据库文件从源机器上复制到目标机器上。在目标机器上把每个文件都放置在与源机器上相同的目录下,使用与之相同的名称。使用源数据库参数文件的副本,打开拷贝的数据库。这样就可以了!

如果复制数据库文件无法放在与源数据库中同样的位置,可以拷贝数据库,并加载控制文件。然后在已加载的控制文件中,针对每一个被拷贝到不同目录位置的文件,用ALTER DATABASE RENAME FILE命令改变其位置。

比如,需要将tools01.dbf文件从/u01文件系统移到/u02文件系统,可以用以下命令来完成:

```
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE RENAME FILE '/u01/PRACTICE/tools01.dbf'
      2      TO '/u02/PRACTICE/tools01.dbf';
SQL> HOST mv /u01/PRACTICE/tools01.dbf /u02/PRACTICE/tools01.dbf
SQL> Remark Repeat datafile moves for other datafiles
SQL> ALTER DATABASE OPEN;
```

可以改变文件的名称及其位置，将已有的控制文件指向新的位置。

注意 也可以在数据库打开时转移数据文件。将表空间或数据文件脱机，重新命名数据文件，在操作系统中移动数据文件，再将表空间或数据文件重新置于联机状态。

如果选择在拷贝数据库时创建一个新的控制文件，除了以下情况外，应完成本章练习中所有的克隆任务：

- 1) 当创建一个控制文件时，在创建控制文件脚本中的CREATE CONTROLFILE关键字后面使用了REUSE关键字，而不是SET关键字。
- 2) 因为保留了数据库的名称，所以无需重新设置日志。如果决定重新利用已有的源数据库的日志，在创建控制文件时把RESETLOGS换成NORESETLOGS。还必须从源数据库中复制联机日志，因为你不会为此重新设置日志。然后，打开数据库时，不使用重新设置日志。可以使用ALTER DATABASE OPEN或ALTER DATABASE OPEN NORESETLOGS命令。

技巧 通过创建一个新的控制文件或保留当前控制文件，可以在同一机器上移动一个数据库。如果创建一个新的控制文件，可以按照本章中给出的过程进行。如果保留当前控制文件，加载该控制文件，转移数据库中的所有数据文件，再用“ALTER DATABASE RENAME FILE”命令重新命名每个数据文件。

6.2.4 复制部分数据库

当复制数据库时（克隆或拷贝），不需要复制所有的数据文件。如果只需要少量的表空间，可以只拷贝这些表空间的数据文件和所有的SYSTEM（系统）表空间文件（以及可能应用的回退表空间，因为在打开数据库时需要这些表空间，同时能够避免重新创建回退段）。可以打开使用了一个新创建的控制文件的复制数据库。在数据库打开后，查看数据库中的数据文件，可能会注意到在Oracle的Home数据库中或dbs目录下的某些文件，这些文件是以字符“missing”开头的。如果在创建控制文件时不指定一个文件，数据库在打开时将没有文件。但是系统表空间中包含的数据字典认为这些文件存在。如果在克隆数据库时需要缺少的文件，可以再次执行复制过程，并确保下一次包含了这些文件。如果数据库并不需要这些丢失的文件，可以从数据库中删除它们。必须删除整个表空间，而不是单独一个文件。

6.3 疑难解答

当克隆数据库时，可能遇到各种问题。通常，这些问题在工程的某个任务中通过一个错误消息反映出来。继续阅读，了解一些可能遇到的常见错误消息。为了使错误易于发现，我把它

们集中在一起。这里集中这些错误的另一原因是，无论是拷贝还是克隆一个数据库，某些错误都会出现。

6.3.1 Connect命令

当试图以SYSDBA连接CLNE数据库时，可能会看到这个错误消息：

ORA-27101: shared memory realm does not exist

如果以SYSDBA身份连接一个数据库，必须拥有口令文件。否则，将不得不使用internal连接。在操作系统命令提示下利用orapwd工具创建一个口令文件。

6.3.2 CREATE CONTROLFILE命令

当试图创建一个控制文件时，会遇到许多错误。我在这里列出一些错误示例、出错的原因以及可能的解决办法。这里的每个错误都以ORA-01503-CREATE CONTROLFILE失败消息开始。伴随Oracle错误消息还有一个操作系统的出错信息，这将有助于查明问题的根源。

1) ORA-00200: controlfile could not be created -ORA-27038: skgfrcre: file exists

CREATE CONTROLFILE命令期望创建一个新文件。可以删除已有的控制文件，或在CREATE CONTROLFILE命令中使用REUSE选项。也可以在数据库参数文件中改变控制文件参数。当使用REUSE子句时要注意：不要覆盖了主数据库的控制文件。

2) ORA-01504: database name 'CLNE' does not match parameter db_name 'PRACTICE'
确保数据库名称与新的init.ora文件以及所创建的控制文件脚本中的一样。

3) ORA-01565: error in identifying file '/oradata/CLNE/system01.dbf'

CREATE CONTROLFILE验证DATAFILE关键字后面的每个数据文件是否存在。务必正确地输入了文件名称，以及正确地拷贝了文件。

6.3.3 RECOVER DATABASE命令

在恢复CLNE数据库时，Oracle可能会询问某个根本没有从PRACTICE数据库创建的归档日志。CLNE数据库可能需要当前PRACTICE数据库归档日志中的重做信息。在PRACTICE数据库上切换日志，使新归档的重做文件可应用于CLNE数据库。尝试将这个重做文件应用于CLNE数据库以完成恢复。可能遇到下面这个错误，这是因为在打开数据库备份完成之后没有切换日志文件：

ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error below.

必须至少在一个归档日志文件上实施完全恢复，以便成功地打开一个来自打开数据库备份的数据库。

6.3.4 ALTER DATABASE OPEN RESETLOGS命令

最后一步，同时也是数据库克隆是否有效的真实测试，那就是打开数据库的时候，可能会遇到这些错误：

ORA-00283: recovery session canceled due to errors

当打开数据库时，所有的数据文件都必须具有同样的当前文件检查点SCN，这个SCN是Oracle在打开数据库之前对文件头部进行有效性检查的一个序号。这个错误指出指定的数据文件需要恢复。遇到这个错误可能出于多种原因。如果遇到这一错误，并试图用一个新名称打开数据库，可能需要从最初的数据文件开始。原因列举如下：

- 1) 源数据库的副本是在数据库打开或异常中断时做的。可以查看所有数据文件的日期-时间值，确认这些文件拥有完全同样的日期和时间。这时必须用一个归档日志或源数据库的某个联机日志进行恢复。
- 2) 恢复命令应用在了某个非当前重做日志或名称不符的数据库的某个日志上。可能不得不重新拷贝所有文件。
- 3) Oracle软件版本与源数据库的版本不同。确保副本数据库运行在与源数据库同样的版本上。检查ORACLE_HOME的环境设置。
- 4) 当从一个打开数据库克隆时，可能没有应用足够的归档日志文件以涵盖对某个或多个数据文件的最后结束备份命令的时间。

6.4 小结

通过拷贝一个已有的数据库，创建了一个数据库。学习了如何通过重新创建控制文件，创建一个使用新名称的数据库。数据库克隆进程允许你测试备份，并快速提供可供选择的数据库环境用于测试和鉴定。同时还进行了一次恢复练习。你将发现这个存储过程在你整个Oracle DBA职业生涯中都会有用。记住，复制数据库的关键是重新创建控制文件。要想获得关于创建控制文件方面的更多信息，请查阅《Oracle 8i Backup and Recovery Guide》一书的第2章、《Oracle 8i SQL Reference》的第9章和《Oracle 8i Administrator's Guide》的第5章。也可以访问Oracle的Metalink站点（<http://metalink.oracle.com>）中关于复制数据库方面的专题。

习题

以下问题可以帮助你复习本章中的重要概念及其细微差别。

1. 通过重新创建一个控制文件，可以创建数据库的一个副本。以下哪个生成CREATE CONTROLFILE命令？
 - A. ALTER SYSTEM BUILD CONTROL FILE
 - B. ALTER DATABASE BACKUP CONTROL FILE TO TRACE
 - C. ALTER DATABASE RECOVER CONTROL FILE
 - D. ALTER SYSTEM BACKUP CONTROL FILE
2. 如果希望改变一个已有数据库的名称，将不得不重新创建控制文件。在CREATE CONTROLFILE命令中哪个关键字可以重新命名数据库？
 - A. SET DATABASE
 - B. RENAME DATABASE
 - C. CHANGE DATABASE
 - D. COPY DATABASE

3. 为什么在打开一个重新命名的数据库时要用ALTER DATABASE OPEN RESETLOGS命令?
- A. 任何已经存在的日志文件可重新命名。
 - B. 日志文件中的数据库名称必须改变。
 - C. 来自数据库备份的任何已有日志文件都不能使用新的数据库名称。
 - D. 日志文件包含了数据库数据文件名称, 日志文件必须改变。
4. 当克隆一个数据库并改变其名称时, 可以使用已有控制文件和重做日志的一个副本。
- A. True
 - B. False
5. 通过创建一个新的控制文件, 可以完成以下哪些任务? (选出所有正确答案)
- A. 改变数据文件的位置。
 - B. 改变重做日志文件的位置。
 - C. 改变Oracle数据库版本。
 - D. 重新命名一个数据库。

答案

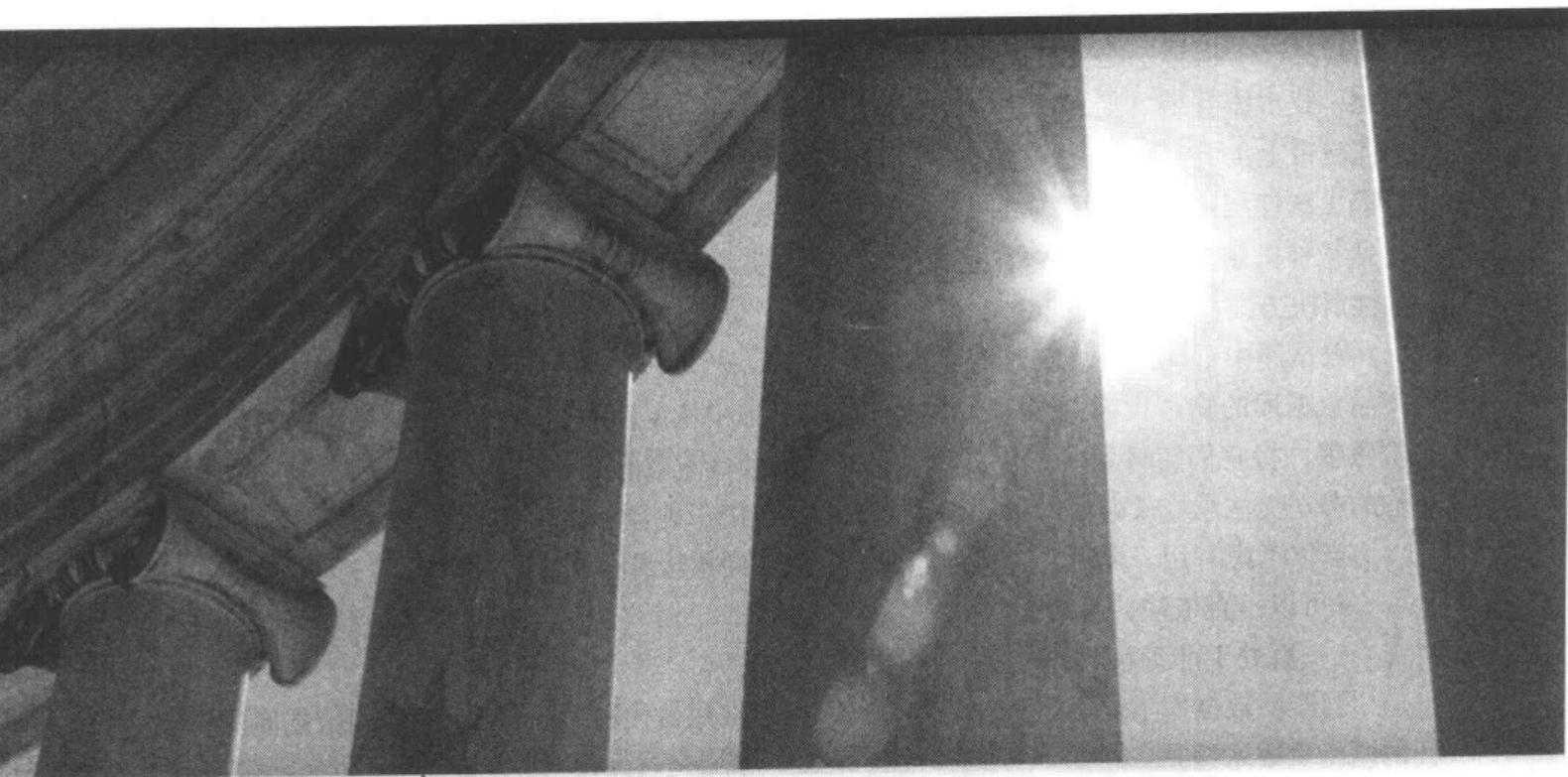
1. B。为了创建一个控制文件——包含CREATE CONTROLFILE命令的文本文件, 使用ALTER DATABASE BACKUP CONTROL FILE TO TRACE。文件将位于user_dump_dest目录下。

2. A。通过在新的控制文件中设置数据库名称, 可以将数据库名称换成别的名称。

3. C。当重新命名一个数据库时, 已经存在的日志文件是无效的, 因为它们包含了旧的数据库名称。

4. False。数据库名称 (尤其是DBID) 不能在已有的控制文件和重做日志中修改。因此, 如果改变数据库名称, 控制文件和重做日志必须重新创建。

5. A、B、D。可以用CREATE CONTROLFILE命令改变数据文件的位置。用CREATE CONTROLFILE命令并使用RESETLOGS选项, 可以重新命名一个数据库。



第7章 备用数据库

修复一个损坏了的数据库需要一些时间。在发现数据库出现问题之后，首先必须查清这个问题，找出哪些地方损坏了，应当如何修复。然后，为了修复数据库，找出备份数据库（通常是在磁带上），还原备份，实施恢复。所有这些操作需要耗费大量时间，这时你的用户和应用程序都无法使用这个数据库。如果长时间内无法应用工作数据库，你的公司会怎么办？所以必须找到某种机制，迅速提供一个可运行的产品数据库。面对这种挑战的数据库解决方案叫做高可用性解决方案。

数据库中心遇到环境灾难时会怎样？洪水、火灾、龙卷风、连续断电，或爆炸都会使承载产品数据库的物理设施瘫痪。从类似这些的灾难中恢复叫做灾难恢复。从离线保存库中找出备份磁带，将它们转移到另一台机器上，并还原和恢复数据库，其间会持续数小时或数天。如果你的公司无法承受长时间的停顿期，可以考虑使用一种包括Oracle备用数据库特性的灾难恢复。

当你考虑备用一词时，会想到什么？备用一词意味着：

- 可以一直依赖，包括紧急情况。
- 一直处于准备状态，将被作为代用品使用。

当产品数据库不能使用时，备用数据库可以被用做代用品。在Oracle备用数据库配置中应用的两个重要术语是：

• **主数据库**：数据库源是主数据库。这个数据库也可被称为源、产品或主动数据库。在本章中，PRACTICE数据库实例将被作为主数据库。

• **备用数据库**：数据库副本是备用数据库。这个数据库也可被称为目标数据库或恢复数据库。这个数据库常常拥有与主数据库相同的名称和实例名称。如果备用数据库与主数据库保存在同一服务器上（仅用于测试），实例名称将不同。在本章中，备用实例的名称将是STANDBY。

当部署一个备用数据库时，创建主数据库的一个副本，并将这个副本数据库保存在某个备用位置。典型的做法是存放到位位于别的物理位置的另一台主机上。为了使备用数据库与主数据库保持一致，归档重做日志被加载到备用数据库上并被应用。如果主数据库出现故障，几分钟内备用数据库就可以被激活，作为新的主数据库使用。这可以避免长期的宕机时间，这些宕机时间也是时间和金钱上的损失。

警告 当我说备用数据库与主数据库一致时，备用数据库只是与所应用的最新归档日志一致。这个归档重做上的延迟即是主数据库与备用数据库的“一致”中的差别。

在开始实现备用数据库之前，还需要指出以下问题：

- 为了真正有用，主数据库和备用数据库必须保存在不同物理位置的远程系统上。
- 存放主数据库和备用数据库的服务器必须拥有相同的机器结构和操作系统。
- 在两个数据库上的Oracle数据库版本必须相同。
- 为了确保主服务器故障时能够维持性能，硬件特性也必须复制。备用主机只能被用做备用服务器，不能兼顾他用。
- 对归档重做日志文件的加载取决于Net8，而这完全依赖于网络。如果网络出现问题，在主

数据库和备用数据库之间将存在更大的时间延迟。

- 进行彻底的错误检查，这样，在归档信息传播和应用遇到问题时（例如，在网络传输中归档日志文件损坏了或归档日志文件应用时出错），才能迅速进行调查和处理。
- 客户机和中间层服务应当做好准备，一旦出现故障，能够适应主数据库向另一台服务器上的转移。这可以通过采用不同的TNS服务名称、改变DNS名称或Net8失效恢复工具来实现。
- 数据库结构和对象操作上的许多变化不会被传播给备用数据库。
- 主数据库必须运行于ARCHIVELOG模式。这些日志文件被用于恢复备用数据库。
- 一定要验证备份备用数据库的存储过程。一旦出现故障，备用数据库将成为主数据库，备份存储过程将要保护新的主数据库。

7.1 Oracle的备用数据库

Oracle的备用数据库（见图7-1）是从Oracle 7.3版本开始引入的。随着8.0版本中RMAN的出现，一种全新的备份和恢复数据库的方法产生了，其中包括一些在创建备用数据库时可以用到的新方法（将在第16章中讨论）。版本8i包括了备用数据库方面包括更多的新特性，它允许Oracle自动完成部分维护任务。使用8i，可以只读方式打开一个备用数据库，然后把它退回到恢复状态。在Oracle 9i中备用数据库特性变得更简单，并被重新命名为Data Guard（数据卫士）。

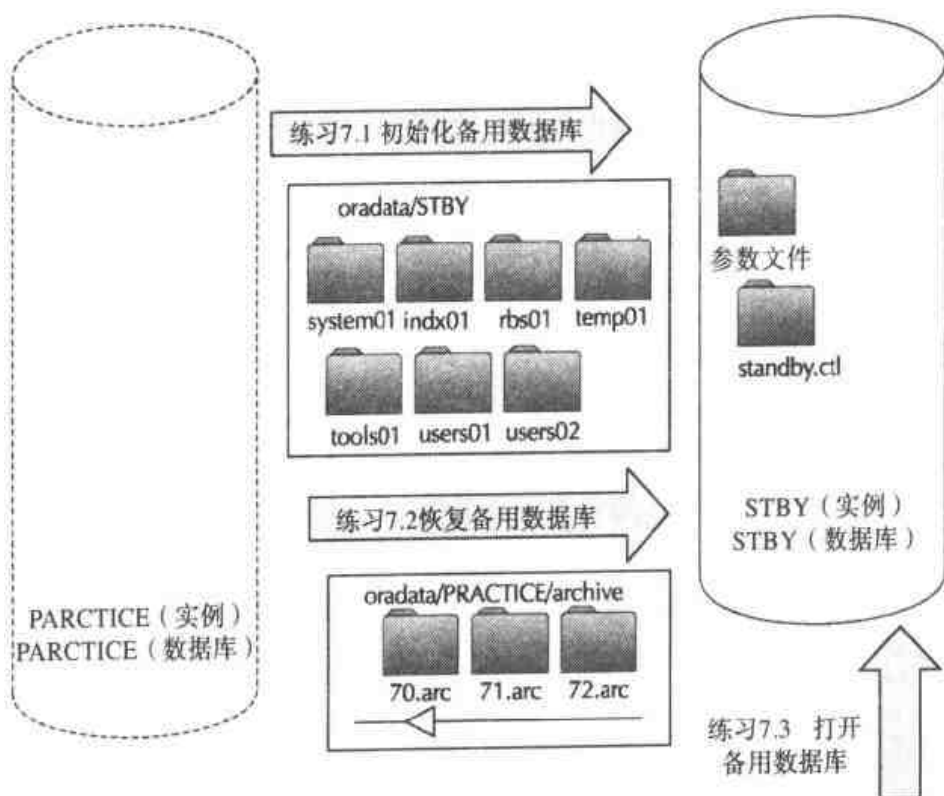


图7-1 PRACTICE数据库的备用数据库

在本章的四个练习中，将初始化一个备用数据库，传送重做信息，打开数据库以便查询，然后激活备用数据库。在我介绍各个练习的任务时，我将就有关备用部署的命令、细节和约束进行简要讨论。

注意 备用数据库的复杂性来源于繁忙的产品系统的维护和自动错误检查。

练习7.1: 初始化备用数据库

在本练习中, 将建立一个名为STANDBY的数据库, 作为PRACTICE数据库的备用数据库。

任务描述	时间 (分钟)
1. 为备用数据库做准备	5
2. 制作打开数据库的备份	10
3. 创建备用控制文件	5
4. 配置备用参数文件	10
5. 加载备用数据库	5
总计时间	35

设置STANDBY数据库是从创建PRACTICE数据库副本的目录 (以及Windows上的服务) 开始的。接下来, 将把PRACTICE数据库的数据文件备份到STANDBY目录结构下。STANDBY数据库将加载其自身特有的控制文件, 这个文件是由PRACTICE数据库生成的。在配置完备用数据库参数文件之后, 就可以把新的数据库加载为备用数据库了。

任务1: 为备用数据库做准备

创建STANDBY数据库时, 需要创建存放数据库管理文件、数据库文件和归档日志的目录, 还需要拷贝和配置一个参数文件, 创建一个口令文件。在Windows环境中还要创建Windows服务, 以此准备备用数据库实例。参照第6章中练习6-1任务1中给出的步骤细节。在本练习中, 要创建的一个数据库实例名为STANDBY, 而不是CLONE。

注意 如果使用一个远程备用数据库, 如果目录结构与主服务器上的相同, 那么将更易于管理。

任务2: 制作打开数据库的备份

利用创建的目录, 可以创建PRACTICE数据库中所有数据文件的备份。创建备用数据库时采用冷备份 (数据库是关闭的) 或热备份 (数据库是打开的) 都可以。当数据库打开时, 制作数据文件的一个副本。回顾第5章中, 创建了一个名为open_backup.sql的脚本。编辑这个脚本, 改变数据文件备份的目的路径。无需把文件拷贝到名为oradata/PRACTICE/backup/user/ch5的备份目录下, 只需改变dir用户变量的值, 使之指向STANDBY数据库目录:

```
define dir = '/oradata/STANDBY'
```

再看下面这一行:

```
prompt 'alter database backup controlfile to '&dir\control.ctl'' REUSE;;
```

将其改为:

```
prompt ALTER DATABASE CREATE STANDBY CONTROLFILE AS '  
/oralog/oradata/STANDBY/standby.ctl' REUSE;;
```

这个命令创建了备用数据库必须使用的专用控制命令。备用数据库不能使用主数据库控制文件的一个普通备份来创建。

现在运行这个脚本，将会创建主数据库的一个备用副本。

注意 如果正在使用一个远程备用数据库，那么需要备份文件，然后用诸如FTP（Unix）或网络映射驱动器（NT/Windows2000）这样的工具来传送。如果使用FTP，请确保传输模式设置为二进制。

任务3：配置备用参数文件

与其他数据库一样，备用数据库需要一个参数文件。该文件中的一些参数需要特别注意，因为这个数据库是个备用的。在任务1中拷贝了这个参数文件之后，用编辑器打开这个文件，做部分改动。针对STANDBY数据库及其路径修改相应的参数如下：

```
db_name = PRACTICE
instance_name = STANDBY
service_names = STANDBY
control_files = ("/oradata/STANDBY/standby.ctl")
log_archive_dest_1 = 'location=/oradata/STANDBY/archive'
standby_archive_dest = "/oradata/STANDBY/archive"
background_dump_dest = /app/oracle/admin/STANDBY/bdump
user_dump_dest = /app/oracle/admin/STANDBY/udump
db_file_name_convert = "/oradata/PRACTICE", "/oradata/STANDBY"
log_file_name_convert = "/oradata/PRACTICE", "/oradata/STANDBY"
lock_name_space = STANDBY
```

以下对该文件进行简要说明：

- **db_name** 备用数据库必须拥有与主数据库绝对相同的名称。这里，数据库名称是 PRACTICE。
- **instance_name** 与内存结构以及后台进程链接的名称是由实例名称参数定义的。实例名称不需要与数据库名称相同。
- **service_names** 连接一个或多个数据库服务的某个实例是由服务名称参数确定的。服务名称 STANDBY 将被用于归档日志在 PRACTICE 数据库和 STANDBY 数据库之间的传输。
- **control_files** 控制文件是由 control_files 参数定义的实例打开的。STANDBY 数据库的 control_file 参数与 PRACTICE 数据库上的控制文件不同。
- **log_archive_dest_1** 在手工恢复期间，这个路径被 Oracle 用来应用归档日志。
- **standby_archive_dest_1** Oracle 利用这个路径来应用那些从主数据库（PRACTICE）上被自动传输/加载的归档日志。这被称为管理恢复。
- **background_dump_dest** 用于后台进程的 alert.log 和调试跟踪文件的路径。
- **user_dump_dest** 用于用户进程跟踪文件的路径。
- **db_file_name_convert** 控制文件是来自 PRACTICE 数据库的一个特别副本，但它仍然认为数据文件是保存在 /oradata/PRACTICE 目录下。这个参数告诉实例在 /oradata/STANDBY 目录下寻找那些数据文件。因为备用数据库是在与主数据库相同的机器上，所以必须将数据文件保存在一个不同的目录下。如果主数据库和备用数据库使用不同的主机，可以将所有的数据文件放置在相同的目录结构下。这是推荐使用的一种做法，这样，就无需定义这个参数。这个参数只能定义一次，但可以使用目录名称的子目录，例如，如果数据文件保存

在/u01/oradata/PRACTICE和/u02/oradata/PRACTICE目录下，那么它们需要被保存在/u01/oradata/STANDBY和/u02/oradata/STANDBY目录下。参数Db_file_name_convert可能如下所示：

```
db_file_name_convert = "/oradata/PRACTICE", "/oradata/STANDBY"
```

- log_file_name_convert 一旦STANDBY数据库被激活，其控制文件将认为联机日志文件在/oradata/PRACTICE目录下。激活数据库时，这个参数通知数据库实例在/oradata/STANDBY目录下创建联机日志文件。
- lock_name_space 这个参数必须使用，以确保两个名称一样的数据库能够保存在同一机器上。分配给这个参数的名称空间值被用于命名该实例的内存结构。这个参数经常被用于允许两个同名数据库共存于同一机器上。

任务4：加载备用数据库

以STANDBY数据库为例，启动实例并加载备用控制文件。使用两个命令来初始化新的备用数据库：将ORACLE_SID环境变量设置为STANDBY，并以SYS在SQL*Plus中连接数据库：

```
LINUX> export ORACLE_SID=STANDBY;
LINUX> sqlplus /nolog
SQL> CONNECT sys/standby AS SYSDBA;
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

如果为STANDBY数据库正确地配置了初始化文件，启动命令就会成功执行。如果该命令未能成功执行，请检查是否如上所述配置了参数。查询v\$datafile，查看数据文件的目录是否已经被db_file_name_convert参数正确地转换了。

好了，这样就初始化了一个名为STANDBY的备用数据库。

练习7.2：恢复备用数据库

现在有了一个备用数据库，如何使这个数据库与主数据库保持一致呢？在PRACTICE数据库上一直发生着的变化必须在STANDBY数据库中反映出来。这些变化将被应用于STANDBY数据库，利用的是来自PRACTICE数据库的归档重做日志。在本练习中，将把归档文件从主数据库传输到备用数据库并应用这些归档文件。在这些归档文件被应用之后，STANDBY将在最后应用的文件上与PRACTICE数据库一致。

任务描述	时间（分钟）
1. 手工恢复——传送归档日志文件	15
2. 手工恢复——应用归档日志文件	5
3. 启动管理恢复	15
4. 确认管理恢复	10
总计时间	45

从主数据库向备用数据库恢复和应用归档日志文件有两种方法，Oracle把这两种方法称为手工恢复和管理恢复。

手工恢复，正如其名，是一种无需Oracle处理的归档日志应用方法。DBA必须建立一个自动

脚本或手工输入指令，将文件从主数据库传输到备用位置，然后在备用数据库上使用恢复命令。这是在Oracle 8i版本之前进行备用恢复的惟一方法。如果没有选用手工恢复，还需要解决日志序列中的不连续问题。在某个日志文件丢失，而且曾经被还原，必须被应用时，会出现这种情况。

管理恢复是相对手工恢复而言的。Oracle将利用Net8监控归档日志文件从主数据库位置向备用数据库位置的传输。这些日志文件在备用数据库上的应用（恢复）也受到管理。这是通过一些特殊的init.ora参数设置和一些恢复命令选项来配置的。备用数据库产生出一个名叫远程文件服务器（RFS）的后台进程，从net8传输中创建归档文件，并保存在standby_arch_dest_1目录下。

任务1：手工恢复——传送归档日志文件

使备用数据库与主数据库保持一致的关键在于有效地传输并应用归档日志文件，在我们这个备用练习任务中，需要完成的就是将归档文件从一个路径拷贝到另一个路径下，以模拟归档日志文件的传输。

注意 如果正在使用一个远程备用数据库，日志文件将需要在网络上传输。如果使用FTP，传输模式必须设置是Binary（二进制）。

为了将PRACTICE数据库的变化传递到STANDBY数据库上，拷贝自STANDBY数据库创建以来所创建的归档日志。在操作系统提示符下，将归档文件从PRACTICE归档目的路径拷贝到STANDBY归档目的路径下：

```
LINUX>cp /oradata/PRACTICE/backup/archive/* /oradata/STANDBY/archive
```

文件拷贝完了，可以准备恢复和应用这些文件了。

任务2：手工恢复——应用归档日志文件

将STANDBY数据库设置为备用数据库后，利用如下RECOVER命令在STANDBY数据库上应用所需的归档文件：

```
SQL> RECOVER STANDBY DATABASE;
```

RECOVER命令将提示归档文件。

接受推荐的文件名称，直到已经应用了所有的归档文件。这个恢复就像在第5章中进行的基于取消的恢复那样：

```
ORA-00279: change 615858460 generated at 01/07/2002 06:51:49 needed for thread 1
ORA-00289: suggestion : /oralog/oradata/STANDBY/archive/70.arc
ORA-00280: change 615858460 for thread 1 is in sequence #70
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

按下ENTER键，应用推荐的日志。可以保留正在应用的日志，直到产生下面这个消息：

```
ORA-00308: cannot open archived log
'/oradata/STANDBY/archive/72.arc'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
```

这个消息说明应用了某个并未产生或拷贝的归档日志。这不会造成什么损害，恢复过程就此取消。如果知道拷贝的最后一个归档日志，就可以在这个最后的文件被应用之后键入cancel以避免出现这个消息。查看STANDBY文件的告警日志，会看到所应用的归档文件：

```
ORA-279 signalled during: ALTER DATABASE RECOVER standby database ...
Fri Jan 07 06:51:49 2002
ALTER DATABASE RECOVER CONTINUE DEFAULT
Fri Jan 07 06:51:49 2002
Media Recovery Log /oralog/oradata/STANDBY/archive/70.arc
```

技巧 使用日志文件的自动传送可以减少管理的工作量。如果选择使用手工恢复来部署一个备用数据库，就需要对日志文件的传输和应用给予密切关注。典型情况下，脚本中的这些进程的运行是无人监视的，并自动检测错误。

任务3：启动管理恢复

为了启动管理恢复，需要配置Net8，这样主服务器上的归档进程可以与备份服务器上的RFS进程通信。配置了Net8之后，除本地路径以外，还必须通知主归档进程将归档日志文件写到一个Net8服务中。最后，备用数据库被配置为以管理模式恢复。归档文件从PRACTICE数据库发出，并被自动应用于STANDBY数据库，直到取消恢复。

利用Net8助手配置Net8配置文件。针对STANDBY数据库的listener.ora和tnsnames.ora各建立一个条目。对于监听者，利用IPC协议创建一个名为STANDBY的新数据库服务（使用IPC协议是因为数据库在本机上）。为一个名为STANDBY的新服务配置tnsnames.ora文件。目的是允许PRACTICE数据库通过监听者与STANDBY实例连接。每当一个归档日志从主数据库发送给备用数据库时，这个连接就被创建和使用。重新启动监听者，这样，做的改动就会生效。

注意 如果使用远程备用，备用服务器的tnsnames.ora将使用IPC与STANDBY通信。在主服务器上，tnsnames.ora文件将使用TCP，因为归档日志文件将在网络上传输。

连接配置好之后，命令备用数据库开始管理恢复。以SYS和STANDBY数据库连接，开始管理恢复。

```
SQL> RECOVER MANAGED STANDBY DATABASE;
```

当提交这个命令之后，可能会认为这个命令的执行被堵塞了。得不到任何反馈信息，SQL提示符也不出现。这是正常的。STANDBY数据库已经启动了RFS后台进程接收任何从PRACTICE数据库传送来的归档日志。当发出一个归档日志时，它是从Net8传输中被拷贝到/oradata/STANDBY/archive目录下。RFS同时还用新的归档日志信息更新备用控制文件。一旦归档日志在STANDBY归档目的路径下重新生成，该文件就被应用于STANDBY数据库了。恢复进程每隔15秒检查一次新的归档日志。重新生成归档日志并应用它们的工作持续进行，直到管理恢复被取消。

```
SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
```

上面这个取消命令必须在STANDBY数据库上的另一个SQL*Plus会话中提交。

最后，通知主数据库开始将归档日志写入已经配置的新服务中。在PRACTICE数据库的参数文件中添加第二归档日志目的路径，并重新启动实例。由于LOG_ARCHIVE_DEST_N可以动态设置，以SYS身份连接PRACTICE数据库实例，用ALTER SYSTEM命令开始向Net8服务写入。只要某个数据库参数是按照这种方式设置的，就需要把这个参数写进init.ora文件中，以确保在数据库重新启动后保留了设置。

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = "MANDATORY service=STANDBY reopen=30"
```

归档进程也将向一个Net8服务中写入归档文件。关键字“mandatory”要求在联机重做可以被写覆盖之前传输必须成功。“Reopen”关键字以秒为单位给出了归档进程返回错误时重试传输操作的时间。归档文件现在将被写入名为STANDBY的Net8服务中。

通过设置LOG_ARCHIVE_DEST_STATE_N动态参数，可以关闭和开启归档日志传输过程。如果归档日志的传输由于某种原因失败了（例如网络问题）可以关闭向备用数据库的文件传输，命令如下：

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = DEFER;
```

推迟归档传输可以允许主数据库继续向本地磁盘上写入归档文件。你并不希望数据库因为归档进程无法向一个目的的路径写入而停止。只要备用归档目的路径已经重新建立，在延迟时间内产生的归档日志文件将不得不手工拷贝到备用服务器上。然后可以通过Net8重新启动归档文件的传送功能：

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = ENABLE;
```

任务4：确认管理恢复

如何知道管理归档传递进程是否正确运行了呢？可以查看以下三个地方：

- 1) Archive files 在STANDBY数据库的归档目的路径中查找从主数据库传输来的归档日志。PRACTICE数据库上新的归档日志文件将在/oradata/STANDBY/archive目录下重新生成。
- 2) Standby alert log 在STANDBY的alert.log中查看归档日志应用记录。如果并未见到任何活动，在主数据库上做几次日志切换。等待几分钟，查看是否有新的归档日志被传输和应用的迹象。

```
Media Recovery Log /oradata/STANDBY/archive/71.arc
Media Recovery Waiting for thread 1 seq# 72
```

- 3) Log history 第3种也是最后一种测试方法就是同时查询主数据库和备用数据库上的v\$log_history。以下查询应当返回相同的数值，注意备用数据库的响应会延迟几秒钟：

```
SQL> SELECT MAX(sequence#) FROM v$log_history;
```

练习7.3：以只读方式打开备用数据库

目前备用数据库已经设置，并与主数据库保持了一致。从版本8i开始可以只读模式打开备用数据库。为什么要这么做呢？因为这样可以允许DBA查看备用数据库是否正确工作了（因为备用数据库可以打开并从中查询数据了）。备用数据库也可被用作是一个发布数据库，以减少对主数据库的影响。当备用数据库以只读模式打开时，不能改动数据。归档日志文件源源不断的从主数据库传输过来，但并不被应用。只有在备用数据库回到恢复模式时归档日志才能被应用。

注意 当备用数据库以只读模式打开时，来自主数据库的归档日志文件将不被应用。因此，长时间的使备用数据库处于打开状态，如果主数据库失效，会增加激活备用数据库所需的时间。

任务描述	时间 (分钟)
1. 产生数据库活动	5
2. 以只读模式打开备用数据库	15
3. 将备用数据库返回到恢复模式	10
总计时间	30

在本练习中将要将在PRACTICE数据库上产生一些数据库活动，停止备用数据库的恢复进程，以只读模式打开备用数据库，检查一下新的数据库活动是否已经被传播了。最后，将备用数据库返回到恢复模式。

任务1：产生数据库活动

连接PRACTICE数据库实例，在TINA.DATE_LOG中插入一些行，这些行的时间值是未来某个时刻。当打开STANDBY数据库时可以在其中查找这一行。插入命令如下：

```
SQL> INSERT INTO tina.date_log VALUES (SYSDATE+365*7);
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

切换日志文件，这样会创建一个新的归档日志文件，并在STANDBY数据库上被应用。

任务2：以只读模式打开备用数据库

使STANDBY数据库保持在管理恢复模式，直到新的归档日志被传输和应用。这些新的归档日志将包含上一任务中插入的新行。取消STANDBY数据库上的恢复：

```
SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
```

打开STANDBY数据库，这样就可以从TINA.DATE_LOG表中提取行了：

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

查看在PRACTICE数据库上插入的行是否出现在STANDBY数据库中。

为了测试来自PRACTICE数据库的日志文件的连续传输，像任务1中那样，在表TINA.DATE_LOG中插入更多的行。在提交了SWITCH LOGFILE命令之后，归档日志文件应当出现在/oradata/STANDBY/archive目录下。

注意 如果要在备用数据库上运行那种所需排序空间超过SORT_AREA_SIZE定义大小的大型查询，则必须在备用数据库中加入一个临时表空间。通过使用一个临时表空间和一个本地管理的临时数据文件，备用数据库的只读特性不会受到影响。这方面的内容请参看《Oracle 8i Standby Database Concepts and Administration Guide》。

任务3：将备用数据库返回到恢复模式

为了使STANDBY数据库从只读打开模式返回到恢复模式，首先必须关闭数据库：

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
SQL> RECOVER MANAGED STANDBY DATABASE;
```

提交恢复命令将开始应用正在等待的归档日志文件，这些归档日志文件是在备用数据库以只读模式打开期间从主数据库传输过来的。

练习7.4：激活备用数据库

当主数据库失效时，可以激活备用数据库成为新的主数据库。当备用数据库激活时，它就是一个打开的数据库，像主数据库那样。这种激活过程无法练习、取消或重试。在激活过程中，联机重做日志被重新设置（类似于以RESETLOGS选项打开数据库），同时来自主数据库的日志文件不再被应用于备用数据库。不必激活备用数据库来查看变动是否已被传播，只需以只读模式打开备用数据库就可以了。如果备用数据库被错误地激活了，这个备用数据库必须重新创建。如果备用数据库被不必要的激活了，主数据库将不再拥有备用数据库了，除非重新创建另一个。

只要打开新激活的备用数据库，就一定要为它做备份（热备份或冷备份）。在本练习中没有第二备用数据库，当然对于一个更为复杂的环境时这很容易配置。如果没有做备份而备用数据库也失效了，就不得不利用还原和恢复了。

任务描述	时间（分钟）
1. 停止备用数据库的恢复进程	5
2. 激活备用数据库	5
总计时间	10

本练习将指导你完成激活备用数据库必需的简单步骤。

任务1：停止备用数据库的恢复进程

你可以关闭PRACTICE数据库实例来模拟主数据库失效，并激活STANDBY数据库实例。此时备用数据库还处于恢复模式，因此需要关闭备用数据库，然后再激活备用数据库。在停止其恢复进程之前，请确保被传输到备用归档日志目的路径的所有归档日志文件都已经被正确地应用了。可以用“确认管理恢复”（练习7.2中任务4）介绍的三种方法中的任一种实现这一点，然后正常关闭备用数据库：

```
SQL> RECOVER MANAGED STANDBY DATABASE CANCEL;
```

任务2：激活备用数据库

这是一个没有回复的过程。在产品环境中，这必然是一次真实的突发事件。一旦备用数据库被激活，它与其他任何Oracle数据库没有什么区别。为了激活备用数据库，首先确保连接了正确的服务器（在本练习中主机是相同的），ORACLE_SID也设置正确了，同时以SYSDBA用户身份连接数据库。恢复进程刚被取消，因此备用数据库正处于加载状态。

```
SQL> ALTER DATABASE ACTIVATE STANDBY DATABASE;
SQL> SHUTDOWN;
SQL> STARTUP
```

联机重做日志文件将创建在/oradata/STANDBY目录下，控制文件现在也成为了主控制文件。为了使备用数据库在激活后可用，必须关闭该数据库，然后重新启动它。

注意 在产品环境中，完成两件事很重要：备份新的主数据库，创建一个新的备用数据库。不完成这两件事，你将面临在另一次事故时丢失数据库的危险。

好的，你现在已经建立、管理、打开和激活了一个备用数据库！

7.2 几点建议

本章中的练习教你如何部署备用数据库。这个备用数据库并不像实际的产品环境中那样完整。实现一个真正的备用站点包括使用位于不同位置上的其他机器。如果选择使用Oracle备用特性，可以借鉴下面这些经验：

- 直接加载那些在主数据库上曾进行过的操作时需要引起特别注意。例如，如果使用SQL*Loader来加载数据时采用direct选项，这些操作在重做日志中记载会有所不同，这会导致备用数据库上受到影响的数据块无效。
- 对主数据库所做的许多物理改变需要在备用数据库上做手工响应。例如，如果在数据库上添加一个数据文件，必须手工在备用数据库上也添加这个数据文件。
- 考虑在两个数据库服务器之间试验switchover和switchback。这将模拟事故发生后备用数据库成为主数据库的情景。Switchback包括将主数据库转移回主服务器上。
- 注意Data Guard特性的部署，这种特性在Oracle 9i以及某些平台上的8i中可用。Data Guard提供了许多新特性，能够帮助杜绝或减少由于人为错误、传播问题、灾难以及数据毁坏面引起的损失。

7.3 疑难解答

在备用部署过程中，可能会遇到很多错误。下面列出了一些可能出现的错误：

1) ORA-00283: recovery session canceled due to errors.

当从一个SQL*Plus会话启动管理备用恢复时，必须从另外一个SQL*Plus会话取消管理恢复。一旦取消命令从第一个会话完成，第一个会话可能不会立即反应，这时如果第一个SQL*Plus会话中按下ENTER键，就会看到这个错误。这个消息所指的错误是对管理恢复的取消。这个消息属于正常的，除非同时还有其他许多错误消息。

2) Managed recovery doesn't seem to be working.

如果没有在备用数据库的alert.log中看到介质恢复消息，可以查看其他一些信息：

- 查看两个数据库上的v\$archive_dest，确信在主数据库上产生的归档日志被送到了备用数据库进行读取的地方。
- 查看备用数据库所期望的归档日志的序号是否正确。通过检查备用数据库的alert.log，可以跟踪归档日志序号，查看哪个日志文件是所期望的。比较这个日志文件和从主数据库传输来的日志文件。
- 用tnsping工具查看与STANDBY服务的网络连接。如果主数据库与备用数据库之间的网络链路不可用或备用监听者停机了，日志就无法传输了。
- 在主数据库的alert.log中查找归档错误消息。

7.4 小结

现在我们已经体验了备用数据库的部署过程。在Oracle服务器上创建了PRACTICE数据库的一个备用数据库。学习了备用数据库的重要参数文件设置。还学习了包括RECOVER、OPEN

READ ONLY和 ACTIVATE在内的重要的备用数据库命令。我们只是简要学习了那些可能成为先进解决方案的技术，但我希望你已经获得了一些自信，并对备用实现的重要细节有所熟悉。在实施实际的备用解决方案时，可以运用在这些练习中获得的经验。通读《Oracle 8i Standby Database Concepts and Administration Guide》一书，了解有关成功实现一个Oracle备用数据库的全部细节。也可以在Oracle Metalink站点上找到备用数据库白皮书。

习题

作为复习，请回答以下问题，巩固本章中的概念和练习。

1. 在设置备用数据库时用哪个命令？
 - A. ALTER DATABASE MOUNT STANDBY;
 - B. ALTER SYSTEM MOUNT STANDBY DATABASE;
 - C. ALTER DATABASE MOUNT STANDBY DATABASE;
 - D. ALTER DATABASE OPEN STANDBY DATABASE.
2. 为什么要恢复备用数据库？（选出所有正确答案）
 - A. 备用数据库崩溃了。
 - B. 来自主数据库归档日志的变化必须被应用到备用数据库上。
 - C. 备用数据库的联机重做日志已经重新设置了。
 - D. 在备用数据库上进行的恢复使之与主数据库保持一致。
3. 当备用数据库打开以备查询时，可以继续恢复该数据库。
 - A. True
 - B. False
4. 管理备用恢复能自动保持备用数据库与主数据库的一致。管理恢复如何保持备用数据库的一致？（选出所有正确答案）
 - A. 备用数据库产生一个后台进程来重新生成归档日志。
 - B. 归档日志从主数据库上通过Net8发送。
 - C. 归档日志在备用数据库上自动被应用。
 - D. 当你在打高尔夫时管理员实施恢复。
5. 当激活备用数据库时，应重新加载备用数据库上的联机重做日志。
 - A. True
 - B. False

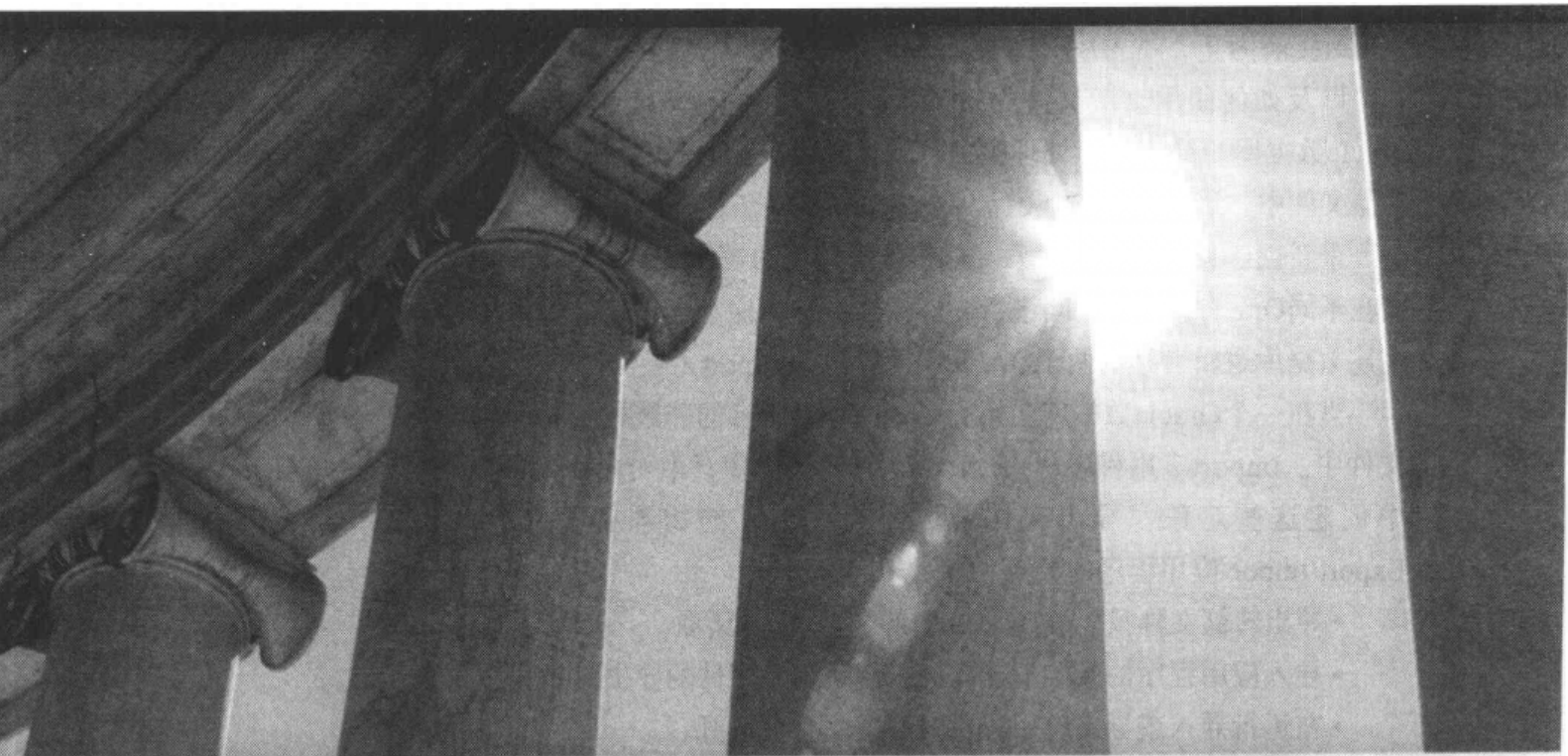
答案

1. C。改变数据库并将其设置为一个备用数据库。
2. B, D。在主数据库上发生的变化必须传播给备用数据库。这些变化包含在主数据库的归档日志中。当这些日志被加载应用到备用数据库上时，变化就体现在备用数据库上了。在备用数据库上发生的归档日志应用是通过RECOVER命令完成的。

3. False。备用恢复和打开状态是互斥的。Oracle并不推荐把备用数据库长时间的作为查询使用，因为归档日志不能被应用于备用数据库。那么，备用数据库就无法与主数据库保持一致。

4. A, B, C。在手工恢复中，必须从主数据库向备用数据库发送归档文件，并应用这些文件。在管理恢复时，归档文件是从主数据库通过Net8发送给备用数据库上的后台进程的。这些传输的归档文件周期性地被应用于备用数据库。管理员什么都不用做。

5. True。除非激活，否则备用数据库一直在没有联机重做日志的情况下运行。激活时，联机日志被重新加载。这个数据库就不能再被用作主数据库的备用了。



第8章 导出与导入

到目前为止，本书中介绍的备份和恢复活动都是物理操作。前面学习了组成数据库的物理文件以及如何使用这些文件保护数据库。如果只希望把数据库的数据和对象拷入或拷出一个Oracle数据库，又该如何处理呢？可以把一个数据库的逻辑内容拷贝到一个Oracle二进制格式的转储文件中，该文件保存在磁盘或磁带上。这个二进制文件的内容可以被读入一个Oracle数据库中以重新创建其中包含的对象。这种数据对象的逻辑转移可以在同一Oracle数据库上进行，也可以在不同Oracle数据库之间进行，即使这些数据库位于硬件和软件配置不同的平台上。用于完成上述数据库逻辑转移的两个Oracle应用程序是Export（导出）和Import（导入）。

当在一个Oracle数据库上运行Export时，所有的非数据字典对象（例如表）都被提取到一个文件中。Import应用程序从一个Export转储文件中读取对象定义以及表数据，并在Oracle数据库中创建这些对象。导出文件可以作为除正常的物理备份之外的备份。请注意以下有关Export/Import应用程序的要点：

- 导出转储文件只能被Oracle的导入应用程序读取。
- 导入应用程序的版本不能比用于创建转储文件的导出应用程序的版本老。
- 在运行导入或导出应用程序时数据库必须打开。
- 导出和导入应用程序可以在任何Net8客户机上运行；所处理的文件常常存放在客户机上。Net8导出或导入会引起额外的网络通信开销。

当运行Export/Import应用程序时，有四种模式，分别代表提取或插入数据库内容时所涉及的工作范围：

- **完整数据库模式 (Full database mode)**：在导出时，除少数内容外，整个数据库的内容都被写入一个文件中。某些用户对象（包括SYS、ORDSYS、CTXSYS、MDSYS、ORDPLUGINS）并不被导出。但数据库结构信息，例如表空间定义和回退段，包含在内。定义完整数据库模式的导出参数为FULL。在导入时，导出文件中包含的所有数据库对象都在数据库中创建。定义用户模式的导入参数是FULL=Y。
- **用户模式 (User mode)**：在导出过程中，属于特定用户的所有数据库对象被写到一个文件中。可以用OWNER参数来指定这些用户。属于该用户的所有表、索引、视图、触发器、同义名、数据库链接、对象、存储代码等等，都被写到导出文件中。导入过程中，在导出文件中属于指定用户的所有数据库对象都在数据库中被创建。导入用户是在FROMUSER参数中定义的。
- **表模式 (Table mode)**：在导出过程中，单个表以及相关的对象（例如：索引、约束、触发器、授权）被写到一个文件中。每个表必须用TABLES参数命名。只有被授权的用户才可以导出属于其他用户的表。在导入过程中，表从导出文件中被读出，并创建在数据库中。定义表模式的导入参数也为TABLES。
- **表空间模式 (Tablespace mode)**：在导出过程中，对应于所选表空间以及这些表空间中

包含的所有对象的元数据被写入一个文件中。实际的表数据（行）并不写入导出文件中。产生的导出文件连同表空间数据文件一起被从源数据库拷贝到目标数据库。在导入过程中，表空间和对象元数据被添加到目标数据库上。在8i版本中这种模式的导出只在将一个表空间从一个数据库传输到另一个数据库时才用（可传输的表空间）。表空间导出将在第9章中介绍。

导出应用程序执行时创建文件的模式与该文件被导入时的模式是互斥的。例如，可以以表模式导入一个用完整数据库模式创建的导出文件。

运行导出应用程序的用户的特权决定了这位用户可以采用哪种模式。拥有CREATE SESSION授权的任何用户都可以按其模式执行表或所有者模式导出。只有被授权的DBA用户，以DBA、SYSDBA或EXP_FULL_DATABASE身份，才可以按照他人模式执行表或所有者模式导出。除此，只有被授予了EXP_FULL_DATABASE系统特权的授权用户才可以实施完整数据库模式导出和表空间模式导出。

运行导入应用程序的用户的授权与导出应用程序的不同。具有IMP_FULL_DATABASE身份的用户可以从一个导出文件进行导入，即使他并未创建这个导出文件。当进行导入时，连接的用户必须拥有执行导出文件中指定命令的权限。例如，如果导入的运行必须执行导出文件中的CREATE TABLE命令，那么运行导入应用程序的用户必须拥有使用该命令的授权，才能使该命令起作用。

Oracle导出和导入应用程序由许多参数控制。有一些参数是两者共有的。表8-1给出了这些共用参数的一个列表以及简要说明。

表8-1 导出与导入共用的一些重要参数

参数	描 述
Userid	userid参数为运行导出/导入应用程序的用户提供数据库用户的id和口令。如果以SCOTT身份连接数据库，应该输入： Userid=scott/tiger 如果准备在Net8上连接，应该输入： userid=scott/tiger@practice 如果必须以SYSDBA身份运行导出或导入应用程序，应该输入： userid=" scott/tiger as SYSDBA"
File	参数file定义了导出应用程序将要创建的或导入应用程序将要读取的文件名称。可以完整的给出路径名称。如果只提供文件名称，文件将被创建到当前目录下或从当前目录下读取。如果没有提供文件名称，这两种应用程序都将在当前工作目录下寻找expdat.dmp文件
Log	导出/导入的屏幕输出可以被捕捉到用参数log定义的一个文件中。可以完整地给出路径名称。如果仅提供文件名称，文件将在当前目录下创建
Help	在命令提示符下，可以键入help=y，来获得导出和导入应用程序的所有参数的一个简要列表。当用HELP参数列出导出的参数时，部分参数在简单的描述后面会给出默认值。例如： ROWS Export Data Rows(Y) 这表示名为ROWS的参数在默认情况下是导出数据行
Parfile	导出/导入应用程序的参数可以从一个被称为参数文件的文件中读取。为了提供一个参数文件，在parfile参数中给出一个文件名称。这个文件应当包含参数及其相应值的一个列表

(续)

参数	描 述
Tables	利用tables参数,在导出时提取表列表中的那些表,或导入那些包含在导出文件中的表
Rows	利用row参数指示导出应用程序提取出每个导出表中的行,或指示导入应用程序插入每个导入表中的行

导出和导入应用程序都分别包含了许多其他的参数。其中一些参数将在本章的练习中介绍。少数参数在导出和导入之间是共用的,但意义和内涵不同,列举如下:

- **Full** 完整导出意味着提取整个数据库的所有内容。完整导入则意味着读取某个导出文件中的所有内容,并创建它们。
- **Owner 和 Fromuser/Touser** Owner参数在导出时提取属于指定拥有者的所有对象。如果导出文件中对象的拥有者在导入时需要更改,Fromuser将指定导出文件中包含的原对象拥有者。Touser指定创建和拥有导入对象的新模式。例如,我可以导出Stephan的对象,并希望把它们按照Kenny的模式导入,这时可以这样使用这些导入参数:FROMUSER=STEPHAN TOUSER=KENNY。

Oracle的导出/导入应用程序有许多有价值的功能和特性,包括:

- **备份和恢复** 导出和导入对于应用开发、数据转移和可迁移表空间是有用的。它们可以方便的用于除质量备份和恢复策略之外的各种DBA任务。导出最多不过是表的一个快照,因为这些表存在于某一时间点上。如果丢失了一个表或数据文件,假如表本质上是动态的,那么用导出文件和导入应用程序来替换所有的数据将是十分困难的。不能对导入的表应用重做信息。但是,或许导出对于扩展实际的备份和恢复有用,例如,在新的数据已经被加载到表之后,可以导出关键的静态表。
- **数据块损坏** 可以导出整个数据库或关键的表,以此发现表中损坏的数据块。导出过程会对被导出的表进行完全扫描,强制读取每个数据块,检查介质损坏情况。
- **数据库版本交叉** 可以将某一版本的Oracle源数据库上的模式和数据拷贝到不同版本的一个目标Oracle数据库上。当从源数据库导出数据时,使用两个数据库中较早的版本进行导出。向目标数据库导入数据时,使用目标数据库的导入版本。
- **操作系统交叉** 可以用导出/导入将数据从某个操作系统上的一个Oracle数据库转移到相同或不同操作系统上的另一个Oracle数据库中。(Oracle 8i的可迁移表空间不能在不同操作系统上的数据库之间迁移。)
- **字符集支持** Oracle的NLS(本地语言支持)特性提供了适合支持本地语言的字符、数字、符号、日期等。如果在导出或导入过程中看到POSSIBLE CHARSET CONVERSION字样时应当留心。请注意导出和导入客户机的字符集值,必要时要调整NLS_LANG环境变量设置。

本章,将要进行一系列简单的导出/导入操作,如图8-1中所示。我将说明如何对整个数据库(数据库模式)、针对一个或多个用户(用户模式)和针对一个或多个表(表模式)执行导出/导入,并练习交互运行导出/导入应用程序,也将讨论如何通过命令行参数和参数文件来控制这些应用程序。本章中大多数练习都使用一个参数文件。下一章中,我将示范如何针对可迁移表空间使用导出/导入应用程序。

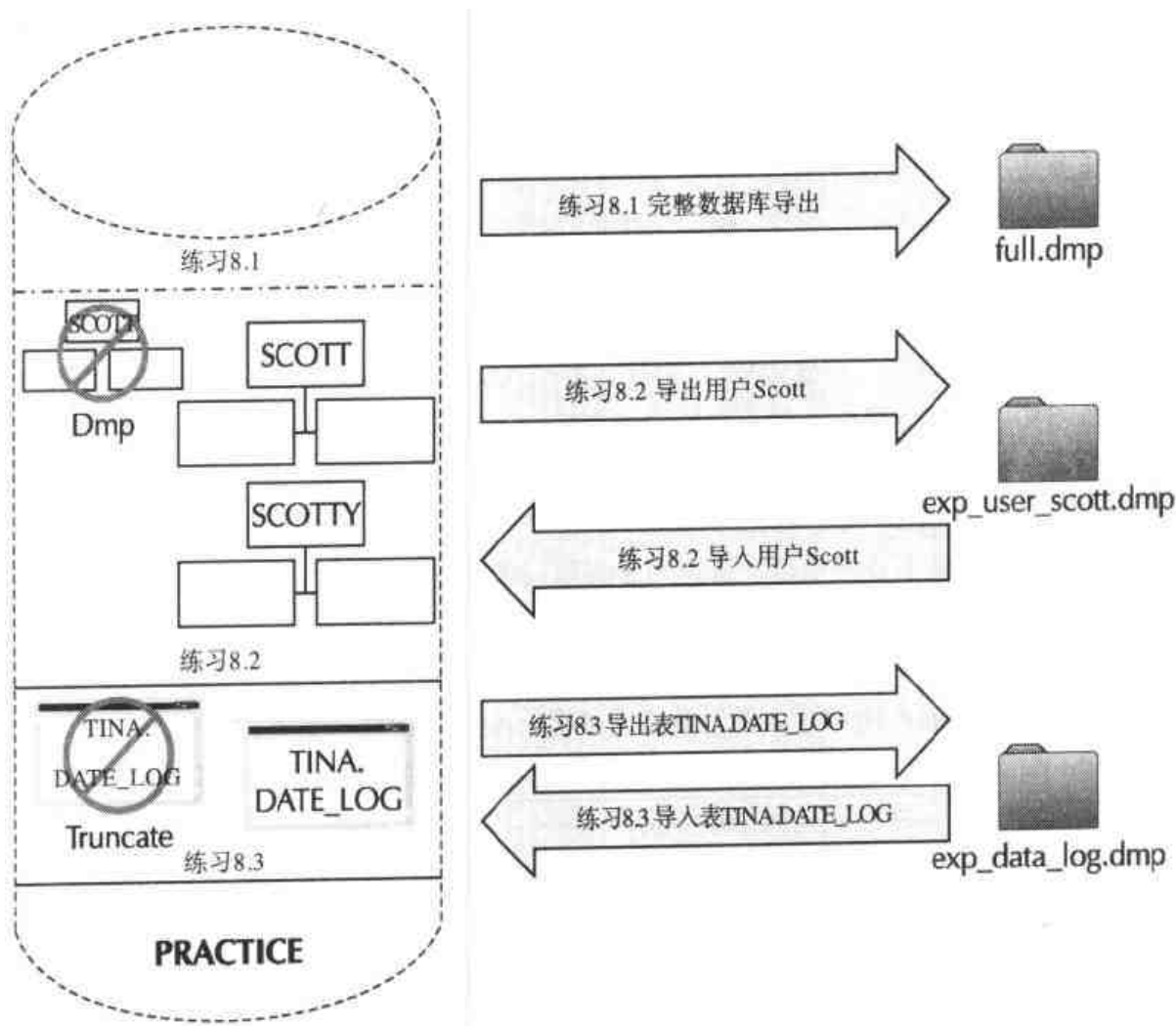


图8-1 导出和导入PRACTICE数据库

8.1 数据库模式的导出和导入

首先进行的导出/导入模式是完整数据库模式。在第一个练习中的导出操作将在整个数据库上执行，导入操作将在整个导出文件上完成。

练习8.1：导出完整的数据库

在本练习中，将创建PRACTICE数据库的一个完整数据库导出。

任务描述	时间（分钟）
1. 交互导出完整的数据库	10
2. 通过命令行导出完整的数据库	10
3. 通过参数文件导出完整的数据库	10
4. 以直接模式导出完整的数据库	10
5. 导入整个数据库	10
总计时间	50

将要完成多次同样的完整数据库导出，举例说明在调用和控制导出应用程序中的选项。在最后一个任务中，将用导入应用程序来浏览导出文件中的内容。

任务1：交互导出完整的数据库

在此任务中，可以通过交互运行导出应用程序创建一个完整数据库导出文件。正如在导出应用程序的提示中看到的，可以学习用于控制导出的关键字。在操作系统的提示符下，键入exp启动导出程序。只要按下ENTER键，就会提出一大串问题：

- 1) username 指将要执行导出的用户的名称。任何用户都可以导出他们自己的模式（要导出并不属于你的对象，需要以EXP_FULL_DATABASE或DBA的用户身份连接数据库）。在此练习中，以SYS身份连接。如果希望在Net8上连接数据库，请在服务名（sys@practice）后面添加用户名。
- 2) password 给出刚才输入的用户口令。
- 3) buffer size 以字节为单位，指定用于提取行的缓冲区的大小。接受默认值。当缓冲区写满时，其内容会被自动写入导出文件中。
- 4) export file 导出应用程序创建一个二进制文件，其中包含了版本信息、运行细节、对象创建命令和表插入语句等。接受默认文件名expdat.dmp。
- 5) entire database 定义导出的范围。可以导出整个数据库、一个或多个用户，或一个/多个表。选择“1”，进行完整数据库导出。
- 6) grants 决定导出应用程序是否导出对象授权。完整数据库导出时，所有的对象授权都应被导出。选择“Yes”。
- 7) table data 导出表时可以带表数据，也可以不带表数据。因为我们希望整个数据库内容都包含在内，因此选择“Yes”。
- 8) extents 如果压缩域，那么在导入时建立的对象创建命令会将所有的数据合并到一个初始域中。如果没有压缩域，对象将按照它们在导出中已有的域设置被创建。选择“No”。

当导出开始运行时，屏幕会显示正在进行的工作。当导出结束时，在导出命令运行的目录下将创建一个名为expdat.dmp的文件。该文件包含了除SYS、ORDSYS、CTXSYS、MDSYS和ORDPLUGINS以外的每位用户所拥有的全部模式对象的一个副本。

任务2：通过命令行导出完整的数据库

通常，可能希望在命令行中或通过一个命令文件提供所有的导出参数。可以把命令行参数放在一行中，以提供导出应用程序所需的全部信息。如果未能给出所需的某个参数，导出应用程序会提示输入参数值。为了用命令行参数将整个数据库导出到一个指定文件中，给出USERID、FILE和FULL参数如下：

```
LINUX> exp USERID=sys/practice
FILE=/oradata/PRACTICE/backup/ch08/full.dmp FULL=Y
```

每个关键字后面跟着一个或一组值。可以省略userid关键字，这时user和password将采用第一个值。

```
LINUX> exp sys/practice FILE=/oradata/PRACTICE/backup/ch08/full.dmp
FULL=Y
```

由该命令创建的文件被命名为 /oradata/PRACTICE/backup/ch08/full.dmp。

任务3：通过参数文件导出完整的数据库

如果不希望以命令行参数的形式提供关键字的值，可以提供一个导出应用程序能够读取的参数文件。可以进行与上一个任务中完全相同的导出，只需将参数写到一个文件中。为了向导出提供一个参数文件名，在parfile关键字后面添加包含了命令的一个文件名：

```
LINUX> exp parfile=export_full.par
```

文件export_full.par中的内容如下所示：

```
# Export the entire database
USERID = sys/practice
FILE = /oradata/PRACTICE/backup/ch08/full.dmp
LOG = /oradata/PRACTICE/backup/ch08/full.log
FULL = Y
ROWS = N
BUFFER = 10000
COMPRESS = N
```

关键字USERID、FILE和FULL与前一个任务命令行中的一样，同时这里还添加了一些有用的参数：

- Log 导出执行的全部输出结果将被写入一个名为/oradata/PRACTICE/backup/ch08/full.log的文件中。
- Buffer 用于提取行的缓冲区大小为10 000字节。适当的设置这个参数将有助于大型导出更快地运行。这里取值10 000字节作为示范。
- Rows 数据行成为导出文件内容的主要部分。设置ROWS=N，导出应用程序将只导出对象定义，而不导出数据行。
- Compress 压缩会导致导入时创建的各个对象使用初始域，这个域大小足以容纳整个对象。这并不意味着采用某种压缩算法压缩导出文件。

请注意有关完整导出的以下几个要点：

- 1) 导出会覆盖原先的导出文件。如果导出文件已经存在，系统不会给出警告。
- 2) 可以在命令行和参数文件中指定参数。如果在这两者中都指定了同一个参数，则命令行参数优先。

任务4：以直接模式导出完整的数据库

常规路径导出，即缺省的方式，是用SQL SELECT语句从表中提取数据。为此，当以手工方式对一个表提交提取命令时，随之是同样的进程。数据块被从磁盘上的数据文件中读取出来，经由数据库缓冲池，计算出数据的量后，数据被写入导出文件中。

直接路径导出几乎完全抛开了缓冲池。数据是直接从数据文件中读取，避开了数据库缓冲池。某些数据块，例如段的头部以及可能的一致读取数据将仍要经过缓冲池。因此在大型表上直接路径导出的运行速度更快。

对于小型的PRACTICE数据库，通过直接路径导出速度上不会有显著提高。但对于大型数据库，直接路径导出比常规路径导出要快好几倍。

用类似于下面给出的这个参数文件运行完整的数据库导出：

```
# Export the entire database with direct option
USERID   = sys/practice
FILE     = /oradata/PRACTICE/backup/ch08/full.dmp
LOG      = /oradata/PRACTICE/backup/ch08/full.log
FULL     = Y
COMPRESS = N
DIRECT  = Y
```

技巧 如果需要导出大量的数据，可以为每个导出文件指定最大文件大小，给出多个需要创建的文件名。例如：如果操作系统的文件大小限制为2GB，可以指定FILESIZE=2000M，并在FILE参数中给出一列文件名（FILE=exp1.dmp, exp2.dmp, exp3.dmp）。每个文件最大可为2GB，但是如果导出的数据不到2GB，最后一个文件则可能会小些。

任务5：导入整个数据库用于显示

导入应用程序利用导出文件把对象和数据拷贝到数据库中。也可以用导入应用程序来显示一个导出文件中的内容。创建输出且不改变数据库的两个参数是SHOW和INDEXFILE。在本任务中，可以运行导入应用程序，查看导出转储文件中的内容。

利用show参数来显示导入的输出结果，如下所示。创建另外一个名为import_full.par的参数文件，并在这个参数文件中设置以下参数值：

```
USERID = sys/practice
FILE   = /oradata/PRACTICE/backup/ch08/full.dmp
LOG    = /oradata/PRACTICE/backup/ch08/full.dmp
FULL   = Y
SHOW  = Y
```

可以在Linux提示符下用名为imp的可执行程序运行这个文件。

```
LINUX> imp parfile=import_full.par
```

导出文件（也被称为卸载文件）的内容在屏幕上飞速滚动，同时它们被写到由LOG参数指定的文件中。利用编辑器，打开屏幕输出的日志文件，查看其内容：

```
. importing TINA's objects into TINA
"ALTER SCHEMA = "TINA""
"CREATE TABLE "DATE_LOG" ("CREATE_DATE" DATE) PCTFREE 10 PCTUSED 40 INITRAN"
"S 1 MAXTRANS 255 LOGGING STORAGE(INITIAL 196608 NEXT 65536 MINEXTENTS 1 MAX"
"EXTENTS 4096 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT"
"T) TABLESPACE "TOOLS""
. . skipping table "DATE_LOG"
...
. importing TINA's objects into TINA
"ALTER SCHEMA = "TINA""
"CREATE PROCEDURE INSERT_DATE_LOG_ROW"
"IS"
"-- Purpose: Insert a row with the current time into DATE_LOG."
"BEGIN"
```

```

" INSERT INTO DATE_LOG (create_date) VALUES (SYSDATE);"
"END;"
"ALTER PROCEDURE "INSERT_DATE_LOG_ROW" COMPILE TIMESTAMP '2001-09-12:09:36:29'"
...
"CREATE UNIQUE INDEX "CREATE_DATE_PK" ON "DATE_LOG" ("CREATE_DATE" ) PCTFRE"
"E 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 393216 NEXT 65536 MINEXTENTS 1"
" MAXEXTENTS 4096 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DE"
"FAULT) TABLESPACE "INDX" LOGGING"
"ALTER TABLE "DATE_LOG" ADD CONSTRAINT "CREATE_DATE_PK" PRIMARY KEY ("INSE"
"RT_TIME") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 393"
"216 NEXT 65536 MINEXTENTS 1 MAXEXTENTS 4096 PCTINCREASE 0 FREELISTS 1 FREEL"
"IST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "INDX" ENABLE "
...
. importing TINA's objects into TINA
"ALTER SCHEMA = "TINA"
"BEGIN sys.dbms_ijob.submit(job=>1,luser=>'TINA',puser=>'TINA',cuser=>'TIN"
"A',next_date=>to_date('2001-10-01:09:48:15','YYYY-MM-DD:HH24:MI:SS'),interv"
"al=>'(SYSDATE + 1/(24*60))',broken=>FALSE,what=>'tina.insert_date_log_row;',nls"
"env=>'NLS_LANGUAGE='AMERICAN' NLS_TERRITORY='AMERICA' NLS_CURRENCY='$'"
"' NLS_ISO_CURRENCY='AMERICA' NLS_NUMERIC_CHARACTERS='.,' NLS_DATE_FORMA"
"T='DD-MON-RR' NLS_DATE_LANGUAGE='AMERICAN' NLS_SORT='BINARY'',env=>'0"
"102000200000000'); END;"

```

上面列出了文件的一些摘录,说明TINA模式部分是什么样的。(格式有些混乱。为了让这个输出结果在创建对象时可用,应当编辑这个文件,或整理成一个脚本。)在文件中首先看到了对TINA的表的定义,接下来存储PL/SQL语句的源代码。可以查看第3章中写入的INSERT_DATE_LOG_ROW存储过程的代码。最后,是Oracle作业的一条记录,用于调用存储过程。通过查看这个文件,已经浏览了导入进程。这个输出文件中有以下两点注意事项:

- 1) 数据库对象被导出和将被导入的顺序。表在索引之前被导入。存储过程在Oracle作业之前。
- 2) 日志文件中的文本定义被隐藏在字词中。利用显示文件输出重新创建存储代码需要有效编辑。

如果只希望在导出文件中包含表和索引定义,那么可以使用INDEXFILE参数,来创建一个只包含表和索引的文件,如下所示:

```

USERID = sys/practice
FILE   = /oradata/PRACTICE/backup/ch08/full.dmp
INDEXFILE = /oradata/PRACTICE/backup/ch08/full.idx
FULL   = Y

```

查看所得到的索引文件,注意,表的定义已经被注释了。可以删除注释部分,通过运行这个文件重新创建该表。

额外的信任

拷贝一个数据库,可以遵循第6章中介绍的步骤,也可以用导出/导入应用程序拷贝一个数据库:创建一个没有对象的新数据库,而不要使用默认对象(数据字典对象),然后实施

源数据库的一个完整导出。利用这个导出文件，将整个文件导入新建的数据库中。如果数据库是在不同的操作系统上，导出/导入应用程序也会运行良好。练习创建一个新的数据库，并导入PRACTICE数据库的完整导出。首先，创建一个新的空数据库，其中包含了与PRACTICE数据库中名称完全相同的SYSTEM、RBS、TEMP以及其他一些表空间。在这个新数据库上运行了全部所需的catalog.sql、catproc.sql以及其他一些数据字典脚本后，就做好了利用PRACTICE数据库导出文件向新数据库实施完整导入的准备。

8.2 用户模式的导出/导入

有时运行导出应用程序，并不需要导出整个数据库。可能只会选择拷贝一个或多个用户的逻辑内容到一个简单的导出文件中。导出用户的主要原因是保护应用系统的重要用户及其数据，创建测试环境，以及将模式拷贝到不同的数据库上。导出应用程序并不一定需要DBA来运行，开发人员可以对他们自己的开发数据做模式拷贝。

练习8.2：替换并克隆用户Scott

本练习中，将把用户Scott的数据库对象导出到一个单独文件中。利用这个文件，可以把该用户复制回数据库，然后用这个文件创建另一个用户。

任务描述	时间（分钟）
1. 导出用户Scott	10
2. 删除用户Scott	10
3. 导入用户Scott	10
4. 导入用户Scotty	10
总计时间	40

在这些任务中，你将获得一些导入操作的经验。首先，创建一个导出文件，该文件包含了用户Scott的所有数据库对象。删除用户Scott之后，将重新创建这个用户，并导入他所有的对象。最后，将把Scott拥有的数据库对象复制给名为Scotty的用户。

任务1：导出用户Scott

利用下面给出的这个参数文件，可以导出名为Scott的用户：

```
# Export user Scott
USERID = sys/practice
FILE   = /oradata/PRACTICE/backup/ch08/export_user_scott.dmp
LOG    = /oradata/PRACTICE/backup/ch08/export_user_scott.log
OWNER  = SCOTT
```

关键字“OWNER”决定了该文件包含属于Scott的数据对象。如果希望导出多个用户，在圆括号中列出这些所有者，并以逗号分隔。如果查看这个导出用LOG关键字定义的日志文件，将看到在导出过程中所创建的屏幕输出。

由于用户Scott具有与数据库的连接权限，他可能会创建自己的导出，而不是依靠DBA为他

完成这项工作。如果Scott确实自己创建了导出，他可以使用一个参数文件，如下所示：

```
# Export user Scott
USERID = scott/tiger
FILE = /oradata/PRACTICE/backup/ch08/export_user_scott.dmp
LOG = /oradata/PRACTICE/backup/ch08/export_user_scott.log
```

注意：当以Scott身份连接数据库运行导出时，无需提供OWNER参数。因为Scott只能导出他自己的数据对象，导出应用程序默认，如果不指定某个表，他就是导出自己的整个模式。

如果在以Scott连接时试图通过指定OWNER=TINA来导出用户TINA，将会得到以下错误信息：

```
EXP-00032: Non-DBAs may not export other users
```

如果希望导出某个用户的模式，可以以那位用户的身份来运行导出，并给出用户口令。如果需要导出一位或多位用户的模式，而又不知道用户的口令，可以以授权DBA用户的身份运行导出，使用口令OWNER。作为一个授权用户，可以一次导出多位用户模式。

任务2：删除用户Scott

可以用如下DROP USER命令来删除用户Scott：

```
SQL> SELECT username FROM DBA_USERS;
SQL> DROP USER SCOTT CASCADE;
SQL> SELECT username FROM DBA_USERS;
```

DROP USER命令删除一位用户和属于这位用户的所有对象。名为DBA_USERS的数据字典视图包含了数据库当前全部用户的一个列表。如果在使用删除命令之前提取该视图中的数据，将看到Scott是数据库的一位用户。在删除命令之后，在这个（或其他任何）数据字典视图中就看不到用户Scott了。

任务3：导入用户Scott

当导入Scott的模式时，Scott用户帐号必须已经存在于数据库中。用户模式的导入并不会创建被导入的用户，它只会创建用户对象：

```
SQL> GRANT CONNECT, RESOURCE TO SCOTT IDENTIFIED BY TIGER;
SQL> ALTER USER SCOTT DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP;
```

在创建了用户之后，可以导入对象。

控制导入应用程序的方式与控制导出应用程序的类似。可以交互运行导入应用程序，或用一个命令行，或用一个参数文件。为了查看可用的参数关键字，在命令提示符下键入help关键字：

```
LINUX> imp help=y
```

请注意有许多与导出中相同的参数，如USERID、FILE、LOG和PARFILE，但没有关键字OWNER。在导出时，使用OWNER关键字来执行用户模式。作为DBA，使用FROMUSER关键字执行用户模式的导入，如下所示：

```
# Import user Scott as a DBA
USERID = sys/practice
FILE = /oradata/PRACTICE/backup/ch08/export_user_scott.dmp
LOG = /oradata/PRACTICE/backup/ch08/import_user_scott.log
FROMUSER = SCOTT
```

作为用户Scott，可以将一个文件导入自己的模式中，而无需使用关键字FROMUSER，如下

所示：

```
# Import user Scott as Scott
USERID = scott/tiger
FILE = /oradata/PRACTICE/backup/ch08/export_user_scott.dmp
LOG = /oradata/PRACTICE/backup/ch08/import_user_scott.log
```

任务4：导入用户Scotty

用户Scott找到你，告诉你他需要一些空的表来测试新应用程序，他希望你将所有他的模式对象复制到一个名为Scotty的新模式中。为此，你可以从任务1的导出中把Scott的模式导入到一个新创建的用户中：

```
SQL> GRANT CONNECT, RESOURCE TO SCOTTY IDENTIFIED BY KITTY;
SQL> ALTER USER SCOTTY DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE
TEMP;
```

在创建了新的用户后，可以把Scott的对象导入到Scotty模式中。这一次，不需要导入数据，而只导入对象，如下所示：

```
# Import user Scotty
USERID = sys/practice
FILE = /oradata/PRACTICE/backup/ch08/export_user_scott.dmp
LOG = /oradata/PRACTICE/backup/ch08/import_user_SCOTTY.log
FROMUSER = SCOTT
TOUSER = SCOTTY
ROWS = N
```

请注意这个参数文件中的一些问题：

- Touser 用户Scott的内容被从导出文件导入到用户Scotty中。
- Rows Scott的对象在Scotty模式中创建，但没有任何数据行。

在导入日志文件中，将会有这样一条内容，解释了所有者的转换事件：

```
. importing SCOTT's objects into SCOTTY
```

额外的信任

有时，可能需要向一个导出文件中没有数据库对象表空间的数据库实施导入。导入文件的内容，如果所请求的表空间并不存在，则将在默认的用户表空间中创建对象。可以在一个不同的表空间中创建对象，然后导入行。参照以下步骤：将Scott的所有对象导入Scotty模式中，但将所有的对象放在INDEX表空间中。具体做法是：从导出文件的导入创建一个索引文件。编辑所创建的这个索引文件，删除注释部分，改变文本文件中所有对象的表空间。在SQL*Plus中以脚本方式运行这个索引文件，在INDX表空间中创建Scotty模式的所有数据库对象。一旦创建了对对象，使用关键字IGNORE=Y，导入文件。利用新创建的对象结构，数据行将被导入到它们新的表空间位置。

8.3 表模式的导出和导入

本章我们要讨论的最后一种模式是表模式。任何用户都可以在他自己的模式上实现这种最

细琐的数据对象转移机制。只有授权的DBA用户才能够导出/导入其他用户模式的表。导出或导入的每个表通常包含了表定义，同时也可能包含了表的索引、约束、触发器、授权等。可以将多个用户的多个表导出到一个导出文件中。当以表模式导入时，可以导入包含在一个导出文件中的多个用户的多个表。

练习8.3：导出一个数据库表

在本练习中，将要创建一个文件，该文件中包含了PRACTICE数据库中TINA模式的一个表。

任务描述	时间（分钟）
1. 导出表	10
2. 截短表	10
3. 导入表	10
总计时间	30

按照查询参数使用where子句来导出TINA.DATE_LOG表。查询参数允许从一个只包含满足查询子句中谓词的数据行的表进行导出。在截短TINA.DATE_LOG表之后，将从导出文件中导入这个表。最后，当从TINA.DATE_LOG中提取数据时，该表中将只有导出过程中创建的那些行。

任务1：导出表

在从PRACTICE数据库中导出部分表之前，向TINA.DATE_LOG表中插入一些将来时间的行。可以向该表中插入'SYSDATE+8'，这样表中将包含比今天更大的日期值。在插入之后给出提交（commit）命令。

利用一个参数文件创建三个表的导出，如下所示：

```

USERID    = sys/practice
FILE      = /oradata/PRACTICE/backup/ch08/export_date_log.dmp
LOG       = /oradata/PRACTICE/backup/ch08/export_date_log.log
TABLES    = (TINA.DATE_LOG)
QUERY     = "WHERE create_date < SYSDATE"
COMPRESS  = N
DIRECT    = N

```

导出文件将包含TINA.DATE_LOG的定义和表中满足where子句条件的行。

任务2：截短表

利用TRUNCATE TABLE命令，可以快速地从TINA.DATE_LOG表中删除所有数据。删除数据不会打断创建该表中行的作业和存储过程。

```

SQL> SELECT count(*) from TINA.DATE_LOG;
SQL> SELECT count(*) from TINA.DATE_LOG
      2  WHERE create_date > SYSDATE;
SQL> TRUNCATE TABLE TINA.DATE_LOG;
SQL> SELECT count(*) from TINA.DATE_LOG;

```

在截短表之后，在表中将看不到任何行。

技巧 如果有非常大的表需要导出，使用COMMIT参数，在每次缓冲池内容（由

BUFFER决定)写满并被写入表时强制执行提交(commit)。默认情况下,当每个表被完全导入时Oracle将提交,这样会导致回退段变得非常大。

任务3: 导入表

可以用导出文件中的数据来替换表TINA.DATE_LOG中的行。为了执行表的导入,使用“TABLES”关键字,指定想要插入的行,如下所示:

```
USERID    = sys/practice
FILE      = /oradata/PRACTICE/backup/ch08/export_date_log.dmp
LOG       = /oradata/PRACTICE/backup/ch08/import_date_log.log
TABLES    = (TINA.DATE_LOG)
IGNORE    = Y
```

关键字“IGNORE”指示导入程序向任何已经存在的表中插入数据行。如果不指定这个关键字,对任何已经存在的表的导入都会引起错误,同时该表的行也不会被导入。谨记,如果表中没有惟一约束限制,使用IGNORE=Y会引起重复的行被导入表中。

行被导入之后,查看确认这些行在表中,同时没有哪行的值大于SYSDATE。

```
SQL> SELECT count(*) from TINA.DATE_LOG;
SQL> SELECT count(*) from TINA.DATE_LOG
      2  WHERE create_date > SYSDATE;
```

现在表TINA.DATE_LOG中有大量的行,但没有一行的日期比今天大。注意,当使用查询参数创建了一个导出文件后,查询条件并不写入导出文件中。当导入这个文件时,无法判断被导入的文件是否包含整个表的内容。如果保留导出文件的一个导出日志文件,将能够查看创建文件时用到的参数。

8.4 Oracle 9i的新特性

以下是在Oracle 9i的导出/导入应用程序中增加的一些特性。如果想进一步深入了解这些技术特性,请参考《Oracle 9i Utilities Guide》。

- **Tablespace exports (表空间的导出):** 现在可以导出驻留在以TABLESPACE参数命名的一组表空间内的所有对象。该特性与可迁移表空间不同。不做检查以确保对象在各个表空间中是自容的。产生的导出文件与其他任何导出文件相同。导出文件的内容可以用FULL、TABLES或FROMUSER/TOUSER方式导入。
- **Table name pattern matching (表名格式匹配):** 表一级的导出得到增强,支持采用混合条件的表名用于命名导出。在Oracle 9i之前,所有的表都采用大写名称。现在的导入程序允许导出混合条件字符命名的表。
- **Resumable exports and imports (可持续的导出和导入):** 现在,当导出/导入遇到与存储管理相关的错误时,可以延缓导出或导入。这种与存储管理相关的问题包括用于导出文件的磁盘空间满、在被导入的表上超出最大域限制,或用于导入对象的数据文件空间耗尽。在解决了这些问题之后,导出或导入可以继续。如果问题在给定的时间间隔内没有解决,导出/导入将被取消。
- **Flashback exports (倒叙导出):** 使用Oracle 9i,可以运行一个查询语句,利用一个SCN

或时间值返回此前某一时间点上的结果。导出应用程序能够利用这种服务器能力。利用FLASHBACK_SCN或FLASHBACK_TIME参数, 可以按照在此前某一时间点上对象的情形将它们导出。为了使用这一特性, 数据库必须配置为使用Automatic Undo Management (自动取消管理), 并定义一个恰当的撤销保留期。

8.5 疑难解答

如果在执行导出/导入操作时遇到问题, 可以从下面找到一些解决办法:

1) EXP-00026: conflicting modes specified.

指定的导出模式相互冲突。例如, 参数FULL、TABLES和OWNER是互斥的。

2) IMP-00015: following statement failed because the object already exists.

在导入时, 不能创建一个已存在的对象。将IGNORE参数设为Y, 忽略此错误。

3) ORA-00001: unique constraint (SCOTT.PK_EMP) violated.

试图插入一个不满足主键的数据行。或者已经导入了这个导出转储文件中包含的数据。

4) Partial Import success and partial failure.

导入应用程序将执行它能够完成的任务。如果在创建某个表时导入遇到一个错误, 没有行会被插入到这个表中。如果导入在插入某行时遇到一个错误, 那一行将不被插入到表中, 但其他行还将继续被插入到表中。应查看日志文件, 检查哪些部分起作用了, 哪些部分没有。

8.6 小结

导出和导入应用程序提供了一种逻辑的、基于文件的机制, 用于在Oracle数据库和表空间(使用可迁移表空间特性)之间转移数据库对象。可以导出/导入整个数据库、数据库中的某个用户, 或数据库中的某个表。虽然有人将此工具作为一种逻辑备份和恢复策略, 但是将其作为一种数据传输机制更加有益。在下一章, 我将介绍如何利用导出和导入在数据库之间迁移表空间。如果想了解有关导出和导入应用程序的更多内容, 请阅读《Oracle 8i Utilities manual》的第1、2章。

习题

请回答以下问题, 加深对本章中概念和练习的理解。

1. 以下哪个参数指定数据库导出将提取一个或多个用户的数据对象? (选出所有正确答案)
 - A. SCHEMA
 - B. OWNER
 - C. USER
 - D. TABLES
2. 导出转储文件只能被Oracle 导入应用程序读取。
 - A. True
 - B. False
3. 在导入过程中, 以下哪些表数据对象首先被导入?

- A. 表索引
 - B. 完整性约束、视图、存储过程和触发器
 - C. 表定义
 - D. 表数据
4. 为了将属于某个用户的数据库对象在另一个用户中创建，为导入应用程序指定哪个关键字？（选出所有正确答案）
- A. TOUSER
 - B. OWNER
 - C. USER
 - D. FROMUSER
5. 可以在一个以TABLES模式关键字创建的导出转储文件上运行一个完整的导入。
- A. True
 - B. False

答案

1. B、D。当导出某个用户的数据对象时，在OWNER关键字后面定义出该用户。在一个Oracle数据库中，以他自己的模式拥有数据库对象的用户被称为这些对象的所有者。可以用表参数来指定多个用户表/对象（TABLES=SCOTTEMP, TINA.DATE_LOG）。

2. True。导出转储文件是一个Oracle二进制格式的卸载文件，只能被Oracle导入应用程序读取。

3. C。在索引、约束和数据可以被加入到表之前，必须创建表。

4. A、D。FROMUSER参数使你能够从包含了多个模式的某个导出文件中导入一个模式子集。TOUSER参数则指定一个用户名列表，这些用户的模式将成为导入的对象。当这些参数一起使用时，可以将用户对象从一个用户克隆到另一个用户上。如果指定了多个模式，模式名将成对出现。

5. True。导入的full关键字指定是否导入整个导出文件。由于导出文件只包含了表定义，导入应用程序将在连接的数据库上以连接用户的身份创建该表中的所有表定义。



第9章 表空间时间点恢复

我们已经学习了如何对整个数据库或部分数据库进行完全恢复，如何对整个数据库进行不完全恢复，以及如何在不完全恢复一个或多个表空间的同时保持数据库的其他部分仍处于当前状态。那么在什么情况下使用不完全表空间恢复选项呢？有以下几种情况：

- 当不慎删除或截短了某个表时；
- 当表中的数据被逻辑损坏时；
- 在数据库局部批处理任务运行不恰当时；
- 当某个SQL语句错误地改变了表数据时；
- 在对某个非常大的数据库的完全恢复可能会持续过长的时间，但却只需要数据库的某一个子集时。

可以恢复数据库的一个或多个表空间，使用一种叫做表空间时间点恢复（TSPITR）的特性，如图9-1所示。TSPITR支持将一个或多个非SYSTEM表空间恢复到与数据库其他部分不同的某个时间点上。为了实施表空间恢复，在一个辅助实例中还原源数据库的一部分。辅助实例用于将选择的表空间恢复到某个时间点上。最后，把恢复的表空间迁移到源数据库，或从恢复的表空间中导出所需的对象。在本章中，将了解如何在PRACTICE数据库上实施一个非常简单的TSPITR。可以利用这个例子，熟悉在实际数据库环境中如何完成TSPITR任务。

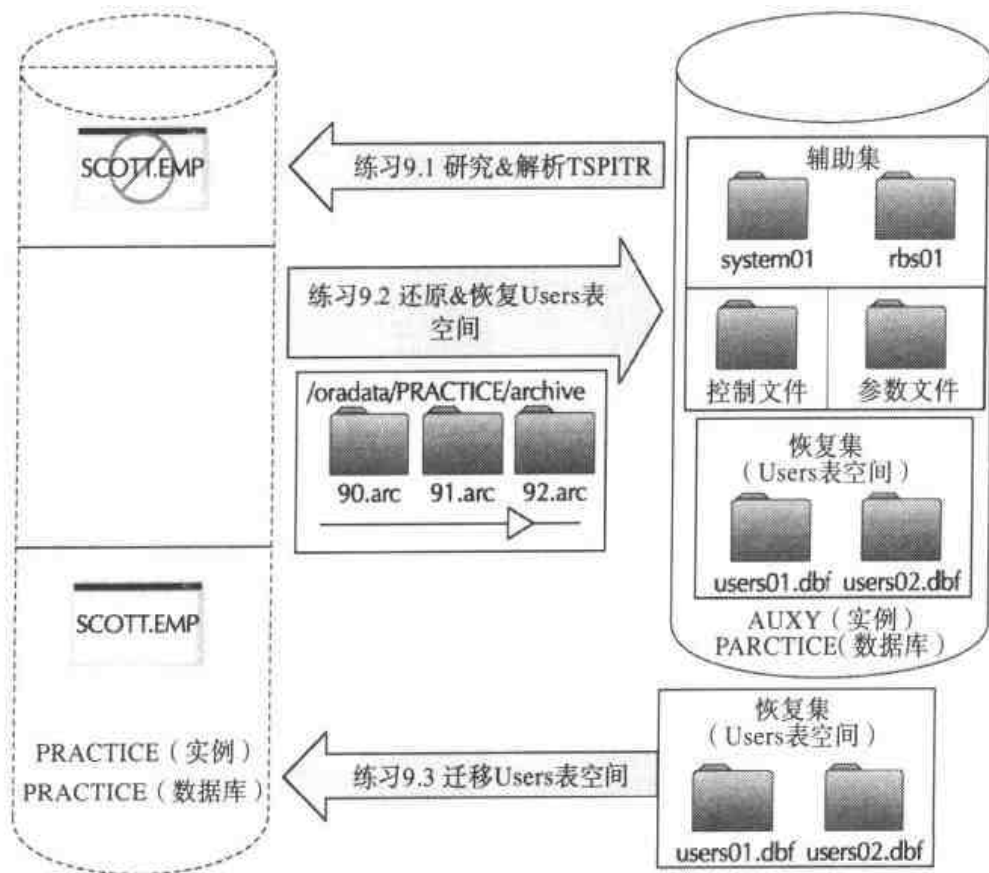


图9-1 PRACTICE数据库的表空间时间点恢复

在本章中，将遇到以下新名词：

- 辅助实例 (Auxiliary instance)：恢复特定表空间的后台进程和内存结构。这个实例将打开辅助数据库，在本章中，这个实例叫做AUXY。
- 辅助数据库 (Auxiliary database)：主数据库的一个副本或是它的一个子集，用于表空间的临时恢复。在本章中，AUXY实例将打开一个名为PRACTICE的数据库，这个PRACTICE数据库是来自PRACTICE主数据库的数据文件和控制文件的一个还原备份。
- 主数据库 (Primary database)：需要TSPITR的数据库。在本章中，即为PRACTICE数据库，该数据库中的USERS表空间将被恢复到与主数据库中所有其他表空间不同的某个时间点上。
- 恢复集 (Recovery set)：构成将被恢复到某一时间点上的表空间的数据文件。SYSTEM表空间数据文件不能作为恢复集的一部分。在本章中，恢复集是USERS表空间内的两个备份数据文件。
- 辅助集 (Auxiliary set)：表空间恢复所需的全部数据库文件。在本章中，恢复集是PRACTICE数据库的备份控制文件，SYSTEM、RBS和TEMP表空间的备份数据文件，辅助数据库参数文件以及PRACTICE主数据库的归档日志文件。

当考虑TSPITR使用Oracle的可迁移表空间特性时，请注意以下重要的事项：

- 在TSPITR完成后，不能使用恢复集数据文件的已有备份，必须使用新备份。这是因为旧的归档重做日志不能被应用于转移到产品数据库中的这些新文件。
- 修改用于TSPITR的辅助数据库实例中的数据的SQL将失败，因为辅助数据库只能被用于恢复。
- 某些数据对象在TSPITR恢复集中是不允许出现的，例如：拥有varray列的表、嵌套表、外部bfile等。
- 注意Oracle 8i中对于可迁移表空间的一般限制。源数据库和目标数据库运行在使用相同操作系统的相同硬件结构上，使用相同的字符集，数据块大小也相同。

在本章中，将把一个表空间恢复到数据库其他部分之前的某个时间点上。不必关闭PRACTICE数据库即可完成上述工作。

9.1 表空间恢复研究

TSPITR的方法很明确。在本章中，当在USERS表空间上实施TSPITR时，创建PRACTICE数据库的一个局部副本。这个辅助数据库包括辅助集的表空间（system、rollback和temporary表空间）数据文件和USERS表空间的恢复集表空间数据文件。所有这些表空间都被恢复到某个时间点上。当不完全恢复进行到预先设定的终止点时，把恢复了的USERS表空间数据文件迁移到PRACTICE数据库中。这些操作类似于第6章中介绍的。辅助数据库是利用主数据库（PRACTICE）的一个备份创建的。辅助实例打开辅助数据库，并利用主数据库归档日志文件将辅助数据库前滚，就像第4、5章中所介绍的那样。还原的数据文件可以是来自一个打开的或关闭的备份。

在实施TSPITR时，必须考虑一些意外情况。在所选终止恢复时间之后创建的恢复表空间中的一些对象和数据可能会丢失。也可能危及在恢复的表空间内外对象之间的关联关系，除非还

原并恢复了含有相关对象的全部表空间，否则必须亲自动手解决这类关联关系。研究这些风险和关系需要花一定的时间。在进行任何TSPITR之前，都应当权衡TSPITR操作的得失。在练习9.1中，将提供一个非常简单的想定，研究并解决TSPITR执行时出现的任何问题。在一个真实的数据库中，这个过程将会更长，耗费更多的精力。

练习9.1：研究并解决表空间的恢复

在本练习中，将“不慎”删除了一个表从而破坏了PRACTICE数据库。然后研究TSPITR实施的得失。

任务描述	时间（分钟）
1. 实施打开的数据库备份	10
2. 删除SCOTT.EMP表	10
3. 检查TSPITR	10
总计时间	30

在实施TSPITR之前，必须检查在恢复和未恢复表空间中表之间应用数据是否不一致。在行动之前希望了解那些需要解析的内在引用的相关性。任务3将告诉你在进行TSPITR之前如何发现和解决这些内在引用的相关性。

任务1：实施打开的数据库备份

为了实施TSPITR，将需要一个用于还原和恢复的备份。可以按照第4章介绍的步骤执行一个关闭的数据库备份，或按照第5章中介绍的方法进行打开的数据库备份。为了实施一个打开的数据库备份，运行第5章中使用的open_backup.sql脚本，将备份目录改为/oradata/PRACTICE/backup/ch09，就像SQL*Plus&dir用户变量定义的那样。

当使用这个打开备份脚本时，请注意在主数据库备份期间这个脚本完成以下三个重要内容：

- 1) 归档主数据库上的当前联机日志，这样所需的任何重做信息都将包含在一个归档日志中。
- 2) 确保组成辅助集和恢复集表空间的所有数据文件都在备份中。
- 3) 在数据文件备份之后创建在辅助集里使用的控制文件。

任务2：删除表SCOTT.EMP

在本章中，将创建一个典型的环境，在这个环境下TSPITR是比较合适的一种选择。这次还是选择SCOTT，删除他的雇员（EMP）表。将利用TSPITR恢复这个表。在删除这个表之前，首先在INDX表空间里创建雇员表的一个索引。在TSPITR过程中需要使用这个索引，因为这个索引和恢复集表空间内的某个对象相关，但索引又不存在于恢复的那个表空间中。其次，记下表被删除前的时间。最后，将USERS表空间恢复到这个时间点上，SCOTT的雇员表将被恢复。

```
SQL> CONNECT scott/tiger;
SQL> ALTER INDEX pk_emp REBUILD TABLESPACE indx;
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD:HH24:MI:SS';
SQL> SELECT sysdate FROM dual;
SQL> DROP TABLE SCOTT.EMP;
```

SCOTT.EMP表上的主键约束要求所有的雇员号都必须存在（非空），同时对于每个雇员行必须是惟一的。ALTER INDEX命令将索引移到INDX表空间中。当在USERS表空间上实施表空

间恢复时，需要删除并还原这个索引。ALTER SESSION命令通过会话改变所有日期列使用的日期格式。你可以看到时间符合基于时间恢复所希望的格式要求。练习9.2的任务3中，在AUXY实例上不完全恢复过程中，还要再次使用这个时间。

为了更好地验证TSPITR的问题与结果，在删除表命令使用之后，在USERS表空间中创建一个对象。同时，向USERS表空间内的某个表中插入一行。最后，在另一个表空间（非USERS）的某个表中插入一行数据：

```
SQL> CREATE TABLE dept_copy TABLESPACE users AS SELECT * FROM dept;
SQL> INSERT INTO dept (deptno, dname, loc)
      2  VALUES ('50', 'SUPPORT', 'DENVER');
SQL> COMMIT;
SQL> CONNECT sys/practice;
SQL> INSERT INTO tina.date_log VALUES (SYSDATE+365*9);
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

实施TSPITR之后，SCOTT.EMP表将不再在PRACTICE主数据库的USERS表空间内存在。由于在TSPITR之后对USERS表空间中对象所做的任何数据改变都不会被恢复，所以在部门（DEPT）表中将找不到support部门。最后，在TSPITR之后TINA.DATE_LOG中的数据仍将保留不变。这样在TSPITR之后将会看到一个未来9年之后的日期记录。TSPITR的好处在于所有没有被恢复的表空间都不会丢失任何数据或对象，弊端是如果与恢复表空间内的对象如索引、触发器和约束等直接相关，那么在未恢复的表空间内的对象仍然会受影响。

任务3：检查TSPITR

在使用TSPITR时，需要权衡利弊。可以回答以下三个问题来确定是否实施TSPITR：

- 1) 如果在一个表空间上把TSPITR实施到某一个特定的时间点上，哪些对象将会丢失？（删除对象）
- 2) 哪个数据库相关的对象可能会妨碍TSPITR的成功完成？（从属对象）
- 3) 在恢复集中是否有对象不能被迁移？（不可迁移对象）

在进行TSPITR之前，可以针对这些潜在问题进行检查。

1. 删除对象（Dropped object）：如果将一个表空间恢复到以前的某个时间点上，会丢失恢复时间之后在表空间内创建的对象。有一个数据字典视图有助于掌握对象的丢失情况，这个视图是TS_PITR_OBJECTS_TO_BE_DROPPED。通过查询这个视图并提供计划恢复的表空间以及计划终止恢复的时间，可以迅速了解如果实施TSPITR将会丢失哪些对象。

```
column owner    format a10
column name     format a10
column tname    format a10
column time     format a20
SELECT owner, name, tablespace_name tname, to_char(creation_time) time
      FROM sys.ts_pitr_objects_to_be_dropped
      WHERE tablespace_name in ('USERS')
            AND creation_time > to_date('2002-01-09:10:30:50',
                                         'YYYY-MM-DD: HH24: MI :SS')
      ORDER BY tablespace_name, creation_time;
```

查询的结果是DEPT_COPY。假设恢复终止在2002年1月9日的上午10:30:50，SCOTT的DEPT表的副本将不存在于恢复的USERS表空间中。

OWNER	NAME	TNAME	TIME
SCOTT	DEPT_COPY	USERS	2002-01-09:10:30:48

将丢失在恢复时间之后对表空间内对象所做的数据改动，如果想找出哪些数据改动没有被应用到USERS表空间中的数据对象上，必须查看从2002年1月9日上午10:30:50到当前日志文件时间之间的日志文件内容。可以利用第10章介绍的Log Miner应用程序检查日志文件的内容。

2. 从属对象 (Dependent object): 第2个检查内容是有什么对象会妨碍将表空间成功地恢复到USERS表。恢复集必须完全包括以下内容:

- 表及其索引。
- 主键/外键索引。
- 大型对象数据类型 (LOB) 的所有组成部分。
- 表/索引的所有部分和子部分。
- 嵌套表的所有部分。

为了找出这些对象及其相关性，可以使用一个名为DBMS_TTS.TRANSPORT_SET_CHECK的过程和一个名为TRANSPORT_SET_VIOLATIONS的视图。这个过程检查USERS表空间是否是自包含 (self-contained) 的。该过程需要2个参数。第一个参数给出一个或多个将被检查的表空间的名称。假如愿意在检查表空间集是否自包含时考虑参考完整性约束，那么第二个参数为真。在调用这个过程之后，从视图中提取出任何非法内容。如果视图不返回任何行，那么表空间集就是自包含的。

```
SQL> execute dbms_tts.transport_set_check('users', TRUE);
SQL> SELECT * FROM transport_set_violations;
VIOLATIONS
```

```
-----
Index SCOTT.ENAME_UK in tablespace INDX enforces primary constraints of table
SCOTT.EMP in tablespace USERS
```

在试图只导出恢复的USERS表空间时，可以看到原有在EMP表上创建的惟一性约束和索引会导致非法。如果只导出USERS表空间的元数据，会得到一个错误消息 “The transportable set is not self-contained.”。因此，为了导出USERS表空间，必须在恢复集中包括INDX表空间，同时还要删除INDX表空间内的索引。在Oracle 8i中，可以使用一个名为TS_PITR_CHECK的视图来查看那些可能妨碍TSPITR运行的相关性和限制。这个视图将确认在恢复集内外的对象的关系。在Oracle 9i中，这个视图已经不存在了，取而代之的是DBMS_TTS.TRANSPORT_SET_CHECK。TS_PITR_CHECK并不提供有关SYS拥有的对象的相关性和限制方面的信息。

```
SELECT *
FROM sys.ts_pitr_check
WHERE (ts1_name IN ('USERS') AND ts2_name NOT IN ('USERS'))
OR (ts1_name NOT IN ('USERS') AND ts2_name IN ('USERS'));
```

在导出之前，应当检查恢复集以外在主数据库和辅助数据库上都有的对象。如果在恢复集以外

有太多的对象在主数据库上，可以增加被包括在恢复集内的表空间。包含更多的表空间可能会减少消耗在删除和还原从属对象上的时间。典型情况是，在将备份文件还原到恢复集之前，检查恢复集以外的从属对象。在辅助数据库上完成恢复后，运行DBMS_TTS.TRANSPORT_SET_CHECK，以确保导出能够成功。同时还要考虑到存在于主数据库中的从属对象在恢复之后可能已经被添加了。因此，查看恢复集在辅助数据库和主数据库上是否都是自含的，比较任何例外情况。在这个简单的想定中，从查询中只会得到一个返回结果。在实际的数据库环境下，将会发现许多需要引起注意的相关性。

注意 这些视图和过程检查在Oracle数据库数据字典内部定义的结构相关性，而不是应用程序所定义和维护的数据相关性。

3. 不可迁移对象 (Non-transport object): 在TSPITR恢复集中不允许出现的对象类型有:

- 复制的主表 (master)。
- 具体化的视图和具体化的视图日志。
- 基于功能的索引。
- Scoped REF。
- 域索引 (用户定义的索引)。
- 属于SYS的对象，包括回退段。

了解了情况之后，就可以基于搜集的信息做出决策。实施TSPITR以找回SCOTT.EMP表和其他丢失的对象与重新创建其他对象所耗费的时间相比是否值得？因为必须找回SCOTT.EMP表，而且能够轻易地重新创建SCOTT.DEPT_COPY表，因此可以对USERS表空间实施TSPITR。

9.2 辅助数据库上的表空间恢复

恢复USERS表空间时，需要用到一个Oracle实例以及辅助集。该实例将需要一个备份控制文件、SYSTEM和ROLLBACK表空间文件。恢复过程将使用PRACTICE数据库的归档重做日志文件。

练习9.2：还原并恢复USERS表空间

在本练习中，将利用同一机器上的另一个实例还原和恢复USERS表空间。这个练习完成了与第6章中类似的任务，主要区别在于只恢复了恢复集和辅助集 (USERS、SYSTEM和RBS) 表空间内的表空间，而且将把这个数据库设置为一个克隆数据库。除此之外，还要完成很多类似于练习6.1中的任务。

任务描述	时间 (分钟)
1. 创建辅助实例	10
2. 将SYSTEM、ROLLBACK和USERS还原到辅助实例中	10
3. 将辅助数据库恢复到特定的时间	10
4. 打开辅助实例	10
总计时间	40

实施TSPITR的首选方法是在一台与主数据库不同的服务器上创建并恢复一个辅助数据库。

这样，可以避免由于覆盖控制文件或联机重做日志带来的损坏主数据库实例的风险。如果非常谨慎，而且有足够的机器资源保存第二个实例，也可以在同一台机器上成功地完成TSPITR。本练习中，在与主数据库相同的机器上创建PRACTICE主数据库的一个名为AUXY的辅助实例。然后，恢复恢复集数据文件，为PRACTICE数据库的导入/导出准备数据文件。

任务1：创建辅助实例

创建一个名为AUXY的实例，包括目录、Windows服务、参数文件以及口令文件，就像第6章中（练习6.1的任务1）介绍的那样。

在参数文件中，只需要有一个控制文件，名为/oradata/AUXY/auxiliary.ctl。同时在参数文件中设置DB_FILE_NAME_CONVERT、LOG_FILE_NAME_CONVERT和LOCK_NAME_SPACE等参数项。

```
DB_FILE_NAME_CONVERT = "PRACTICE","AUXY"
LOG_FILE_NAME_CONVERT = "PRACTICE","AUXY"
LOCK_NAME_SPACE = "AUXY"
```

也可以调整参数文件中的参数（DB_BLOCK_BUFFERS、SHARED_POOL_SIZE、LARGE_POOL_SIZE等），这样辅助实例可以使用较少的内存。DB_NAME参数不变，仍为PRACTICE。然后创建PRACTICE数据库的辅助实例。

警告 切记，辅助实例不要使用PRACTICE实例的控制文件，也不要使用PRACTICE实例的联机日志文件或打开PRACTICE实例的任何数据文件。这么做，会破坏主数据库。

任务2：将SYSTEM、ROLLBACK和USERS还原到辅助实例中

将练习9.1中由打开备份创建的备份控制文件拷贝到/oradata/AUXY/auxiliary.ctl中，这个备份控制文件将被用于加载数据库。

将恢复集中的数据文件从/oradata/PRACTICE/backup/ch09目录下拷贝到/oradata/AUXY目录下，在联机备份时这些数据被保存在这个目录下。这些数据文件来自SYSTEM、RBS和USERS表空间。当创建一个辅助实例用于TSPITR时，不仅要从准备恢复的表空间中还原数据文件，还要还原SYSTEM和ROLLBACK（Undo）数据文件。这是因为SYSTEM表空间中存储了数据字典。需要这个表空间打开数据库、登录数据库、查看对象，从数据库导出等。需要Undo表空间是因为在数据库上实施不完全恢复时，对于恢复取消时尚未提交的所有未完结的事务，数据块的改变都需要回退。

注意 如果希望在辅助实例上做某种排序操作时，也需要临时表空间数据文件。临时表空间很少需要，因为只要数据库一打开，就会导出这个表空间。另外，在磁盘上的排序段只有在排序不适应为实例定义的排序区域时才需要。

发布以下命令，为AUXY实例加载（mount）备份控制文件：

```
LINUX> export ORACLE_SID=AUXY
LINUX> sqlplus /nolog
SQL> CONNECT SYS/AUXY AS SYSDBA;
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT CLONE DATABASE;
```

当将PRACTICE数据库的辅助实例作为一个克隆数据库加载时，那个数据库即脱离出归档日志模式，而且所有的数据文件都脱机。被加载为克隆实例的数据库无法假定所有的数据文件都在控制文件定义的位置。这种特性为你基于可用的磁盘将文件还原到其他地方提供了灵活性。另外，在联机状态时克隆实例的多数数据文件将根本不需要，因为这些数据文件并非恢复集或辅助集的一部分。恢复少量选定的数据文件是这个数据库被设置为克隆数据库的惟一目的。

在加载了备份控制文件之后，可以查看控制文件中有关数据库中当前数据文件的记录。DB_FILE_NAME_CONVERT已经把所有数据文件的PRACTICE目录改为AUXY目录。

```
SQL> col name form a30;
SQL> col file# form 99;
SQL> SELECT file#, name, status FROM v$datafile;
FILE# NAME STATUS
-----
1 /oradata/AUXY/system01.dbf SYSOFF
2 /oradata/AUXY/rbs01.dbf OFFLINE
3 /oradata/AUXY/users01.dbf OFFLINE
4 /oradata/AUXY/temp01.dbf OFFLINE
5 /oradata/AUXY/tools01.dbf OFFLINE
6 /oradata/AUXY/indx01.dbf OFFLINE
7 /oradata/AUXY/users02.dbf OFFLINE
```

还要查询视图v\$logfile以确保联机日志文件保存在一个与PRACTICE主数据库不同的位置。

在恢复之前，将SYSTEM、RBS和USERS表空间数据文件置于联机。当将数据文件置于联机用于恢复时，这些数据文件必须保存在控制文件认为它们应该保存的地方。

```
SQL> ALTER DATABASE DATAFILE 1,2,3,7 ONLINE;
SQL> SELECT file#, name, status FROM v$datafile;
FILE# NAME STATUS
-----
1 /oradata/AUXY/system01.dbf SYSTEM
2 /oradata/AUXY/rbs01.dbf ONLINE
3 /oradata/AUXY/users01.dbf ONLINE
4 /oradata/AUXY/temp01.dbf OFFLINE
5 /oradata/AUXY/tools01.dbf OFFLINE
6 /oradata/AUXY/indx01.dbf OFFLINE
7 /oradata/AUXY/users02.dbf ONLINE
```

再查看视图v\$datafile，就会发现恢复集和辅助集里的数据文件处于联机状态，其他的数据文件是脱机状态。在本示例中，无需恢复TOOLS和INDX表空间。当恢复集文件联机时，可以开始将表空间恢复到从USERS表空间里删除SCOTT.EMP表之前的某个时间点。

任务3：将AUXY恢复到特定时间

将USERS表空间恢复到删除命令之前的某个时间。从辅助数据库的默认归档目的路径读取归档文件。由于这些数据库是在同一台机器上，可以应用PRACTICE数据库归档目的路径下保存的文件中包含的重做信息。设置LOGSOURCE SQL*Plus系统变量，将恢复命令指向/oradata/PRACTICE/archive目录。

```
SQL> SET LOGSOURCE /oradata/PRACTICE/archive;  
SQL> RECOVER DATABASE USING BACKUP CONTROLFILE  
2 UNTIL TIME '2002-01-09:10:30:50';
```

利用恢复命令，联机表空间所需的全部重做信息都被应用，直到2002年1月9日上午10:30:50，这个时间刚好就是SCOTT.EMP表被删除之前的时间。如果恢复结束，没有错误，将得到一个“media recovery complete”消息。还要注意的，不是执行了基于取消的恢复，就是执行了基于SCN的恢复。

任务4：打开辅助实例

打开辅助实例，重新设置日志。辅助实例已经打开了PRACTICE主数据库的一个备份作为辅助数据库，只有SYSTEM的回退段被置为联机状态。这将防止对数据库中的任何用户对象执行任何数据操纵语句（DML）。将一个非系统回退段置于联机状态的任何企图都将失败。

在辅助实例上，可以利用前面提到的DBMS_TTS包，比较结果，解决所有相关性，就像以前在PRACTICE主数据库上所做的那样。如果这个视图没有返回任何行，就可以确信TSPITR的导出阶段将会成功。此时，删除SCOTT的雇员表上的主键，这样期望进行的表空间迁移就会成功。

USERS表空间数据文件现在已经做好准备，可以从辅助实例迁移回PRACTICE实例了。

9.3 可迁移的表空间

还记得第8章中的第4种导出/导入模式吗？下面将利用这第4种模式把一个表空间从一个实例插入另一个实例中。可迁移表空间特性使你能够将一组表空间从一个Oracle数据库迁移或拷贝到另一个数据库中。为了迁移一个或多个表空间，首先要将源数据库上的表空间设置为只读。利用Oracle导出应用程序提取出保存在数据字典中的数据库信息（元数据）。数据文件和元数据导出文件都必须拷贝到目标数据库上。这些文件的迁移可以使用任何拷贝二进制文件的工具来完成，例如操作系统的拷贝工具或二进制模式的FTP。最后，使用Oracle导入应用程序把表空间元数据写入目标数据库的数据字典中。在元数据导入之后，可以将可迁移表空间集设置为读-写模式。

一个可迁移的表空间由两个部分组成：

- 1) 某个表空间的元数据（而非数据）以及其中包含的全部对象的一个导出。
- 2) 属于那个表空间的数据文件的一个副本。

迁移表空间是通过把表空间的两个部分拷贝到一个兼容的Oracle数据库中，再通过完成以下步骤把表空间“插入”到数据库实例中：

- 1) 将表空间数据文件还原到新的位置。
- 2) 导入表空间的元数据。

注意 浏览DBA_TABLESPACES和v\$datafile的plugged_in列，查看哪个表空间和数据文件已经被插入了。

练习9.3：迁移USERS表空间

目前在PRACTICE数据库的PRACTICE实例上有一个USERS表空间，但数据库中没有表

SCOTT.EMP。在辅助实例中也有USERS表空间，表SCOTT.EMP也存在。需要将这个恢复的表空间从AUXY实例迁移到PRACTICE实例中。下面将要使用Oracle的可迁移表空间特性来完成TSPITR操作的最后这一步。

任务描述	时间（分钟）
1. 改动AUXY上的USERS表空间	5
2. 从AUXY上导出USERS表空间	5
3. 从PRACTICE中删除USERS表空间	5
4. 拷贝数据文件	10
5. 在PRACTICE上导入USERS表空间	5
6. 验证PRACTICE上的TSPITR操作	10
7. 备份PRACTICE数据库	10
总计时间	50

在本练习中，将使用导入/导出应用程序，把USERS表空间内的元数据从AUXY实例迁移到PRACTICE实例中。使用操作系统的拷贝命令来拷贝数据文件。最后，在PRACTICE实例和PRACTICE数据库上验证结果。由于你在这些任务中同时在两个实例上工作，因此可能会打开两个操作系统会话：一个用于AUXY实例，另一个用于PRACTICE实例。

任务1：改动AUXY上的USERS表空间

更改不能出现在表空间迁移过程中，因此，在被导出之前，表空间必须被设置为只读。当将一个表空间设置为只读时，属于该表空间的数据文件成为检查点，在表空间内不允许再有任何更新。

```
LINUX> sqlplus /nolog
SQL> connect sys/auxiliary as sysdba;
SQL> alter tablespace users read only;
```

任务2：从AUXY上导出USERS表空间

使用导出应用程序将USERS表空间的元数据提取到一个二进制转储文件中。

```
LINUX> export ORACLE_SID=AUXY
LINUX> exp parfile=exp_transport.par
```

利用以下参数文件创建导出文件：

```
userid="sys/auxiliary as sysdba"
transport_tablespace=y
tablespaces=users
file=transport.dmp
log=transport.log
```

这个文件中的多个重要关键字指示导出应用程序将USERS表空间提取到一个二进制转储文件中。

- **userid** 表空间导出必须由某个具有SYSDBA身份的用户完成。为了将userid关键字定义为SYSDBA，必须使用引号。
- **transport_tablespace** 这个参数支持对可转移表空间元数据的导出。当设置为Y时，导出运行于表空间模式（tablespace_mode），并连同TABLESPACES参数一起使用。

- **tablespaces** 这个参数规定, 指定表空间内用于段的数据字典包含的全部元数据都将被导出。

利用这次导出的转储文件, 连同数据文件, 就可以把USERS表空间数据文件插入到另一个拥有同样的块尺寸、Oracle版本和操作系统的数据库中。

任务3: 从PRACTICE中删除USERS表空间

在USERS表空间可以被插入PRACTICE数据库之前, 不能有同名的表空间存在。当连带内容一起删除表后, 在该表内包含的对象将被从数据字典中删除, 同时也删除操作系统上的数据文件。

```
SQL> ALTER DATABASE DROP TABLESPACE users INCLUDING CONTENTS;
LINUX> rm /oradata/PRACTICE/users01.dbf
LINUX> rm /oradata/PRACTICE/users02.dbf
```

技巧 在Oracle 9i中, 当删除表空间时, 可以从操作系统中删除数据文件。为了删除USERS表空间及其数据文件, 使用这个命令: DROP TABLESPACE USERS INCLUDING CONTENTS AND DATAFILES。

任务4: 拷贝数据文件

AUXY实例恢复的文件可以从它们的当前位置转移到最终位置。

```
LINUX> cp /oradata/AUXY/users01.dbf /oradata/PRACTICE
LINUX> cp /oradata/AUXY/users02.dbf /oradata/PRACTICE
```

如果要将这些数据文件转移到另一台服务器主机上, 可以使用FTP的二进制模式来完成。

任务5: 向PRACTICE导入USERS表空间

现在USERS表空间数据文件已经准备就绪, 可以将该表空间插入到PRACTICE主数据库中。使用导入应用程序将元数据插入PRACTICE实例中:

```
LINUX> export ORACLE_SID=PRACTICE
LINUX> imp parfile=imp_transport.par
```

使用下面给出的这个参数文件:

```
userid='sys/practice as sysdba'
transport_tablespace=y
tablespaces=users
datafiles='/oradata/PRACTICE/users01.dbf','/oradata/PRACTICE/users02.dbf'
file=transport.dmp
log=imp_transport.log
```

请注意这个参数文件中的某些关键字:

- **userid** 表空间导入必须是以某个SYSDBA身份的用户来完成。为了将userid关键字定义为SYSDBA, 必须使用引号。
- **transport_tablespace** 这个参数支持对可迁移表空间元数据的导入。当将其设置为Y时, 导入运行于表空间模式 (tablespace_mode), 并连同TABLESPACES参数一起使用。
- **tablespaces** 当TRANSPORT_TABLESPACE被设置为Y时, 使用这个参数给出一系列将被迁移到数据库中的表空间。

- **datafiles** 当TRANSPORT_TABLESPACE被设置为Y时，使用这个参数列出将被迁移到数据库中恰当位置的数据文件。

任务6：验证PRACTICE上的TSPITR操作

已经被导入的表空间现在处于只读模式。将这个表空间设置到读-写模式：

```
LINUX> sqlplus /nolog
SQL> CONNECT sys/practice;
SQL> ALTER TABLESPACE users READ WRITE;
```

检查USERS表空间内的对象和数据是否像SCOTT.EMP表被删除之前那样存在着。查看SCOTT.EMP表是否存在，而SCOTT.DEPT_COPY表是否不存在。还要查看在SCOTT.DEPT中是否能够找到SUPPORT部门。现在应当能找到SCOTT.EMP表，但找不到DEPT_COPY表，或SUPPORT部门并不存在于SCOTT.DEPT表中。现在进行迁移后的后续工作。在本示例中，应记住在SCOTT.EMP表内添加迁移之前删除的主键。

查看其他表空间内的数据，其中包括了USERS表空间恢复之后更改过的数据。用以下命令查看TINA.DATE_LOG表中是否有一个未来9年后的时间插入值：

```
SQL> describe scott.emp;
SQL> describe scott.dept_copy;
SQL> SELECT * FROM scott.dept;
SQL> SELECT max(create_date) FROM tina.date_log;
```

这样就成功了！在不关闭PRACTICE数据库的同时，在该数据库上实施了表空间的时间点恢复。赶快告诉你的老板，你拥有TSPITR方面的经验，需要加薪\$10 000！（当然你必须要解释TSPITR是什么意思。）

额外的信任

在实施TSPITR时可以与练习9.3中介绍的稍有不同。不用将整个恢复表空间从AUXY实例迁移到PRACTICE实例中，可以从USERS表空间导出特定的对象。只要AUXY实例恢复了，而且打开了该实例，就可以使用Oracle导出应用程序创建某些对象的一个二进制转储文件。利用这个二进制转储文件，可以把那个对象导入到PRACTICE实例中。练习以下步骤：不要从AUXY实例中迁移整个USERS表空间，而只导出SCOTT.EMP表，并把这个表导入到PRACTICE实例中。利用这种方法，可以保留PRACTICE实例上完成的其他工作，包括DEPT_COPY表的创建。（在Oracle 8.0之前，这种对象恢复方法是实现TSPITR的唯一途径。）

任务7：备份PRACTICE数据库

一旦完成了TSPITR，必须再次备份数据库。不能用TSPITR之前的数据文件恢复表空间，因为来自那个备份的归档日志与新恢复的表空间不一致。可以像第4章中介绍的那样实施一个关闭数据库的备份，或像第5章中讲述的那样完成一个打开数据库的备份。

9.4 可迁移表空间的注意事项

除TSPITR之外，还可以在其他地方使用可迁移表空间，如：

- **数据仓库系统和大型的批操作** 你可以快速地将数据从一个数据库迁移或拷贝到另一个数

据库，就像把表空间从一个地方拷贝到另一个地方一样（在迁移表空间以及数据库实例存在于同一机器上时，甚至无需迁移文件）。对于执行大型的加载操作，可以在另一台服务器上的第二个数据库上完成。只要数据被加载了，而且正确地格式化了这些数据，数据就可以迁移到产品级环境中。这可以节省产品服务器的资源消耗。

- **在开发系统上的测试过程中重新设置数据** 在一个开发系统的一个或多个表空间内建立一个数据基线，利用可迁移表空间来导出元数据，并将数据文件拷贝到备份位置。在测试结束后，可以再次插入基线表空间。

9.5 疑难解答

如果在TSPITR操作过程中遇到问题，可以从下面的信息中找到可能的解决方法：

- 1) ORA-01547: warning: RECOVER succeeded but OPEN RESETLOGS would get error below:

ORA-01195: online backup of file 1 needs more recovery to be consistent

ORA-01110: data file 1: '/oradata/AUXY/system01.dbf'

查看为恢复所设置的时间是否正确。数据文件并没有被恢复到一个适于打开数据库的时间。或者辅助备份最初是来自一个还原的打开备份，而且尚未抵达终止热备份标记处。在数据库可以被打开之前需要应用更多的恢复。

- 2) Mount database ORA-01103: database name 'PRACTICE' in controlfile is not 'AUXY'

辅助实例必须拥有与源实例相同的数据库名。查看AUXY实例参数文件中的db_name设置。在本练习中它应该被设置为PRACTICE。

- 3) Mount database ORA-01102: cannot mount database in EXCLUSIVE mode

当试图加载（mount）一个已经被其他实例加载的数据库时，会显示这个错误。当打开同一机器上的两个同名数据库时，这两个数据库共享相同的内存空间。如果指定一个不同的LOCK_NAME_SPACE，内存空间指针将会不同。检查ORACLE_SID设置是否正确，以及AUXY实例是否与克隆数据库加载了同样的控制文件。

- 4) ORA-01698: a clone database may only have SYSTEM rollback segment online

当试图将一个非系统回退段设为联机时，将会出现此错误。为了防止DML在作为克隆数据库而打开的AUXY数据库上起作用，Oracle不允许把任何非系统回退段设为联机状态。

- 5) ORA-29303: user does not log in as SYS

为了导出一个表空间，必须以SYSDBA角色登录。

- 6) ORA-01696: controlfile is not a clone controlfile

如果不用ALTER DATABASE BACKUP CONTROL FILE创建一个备份控制文件，当用克隆选项设置一个数据库时，会出现这个错误。找到一个有效的备份控制文件，或者从主数据库创建一个新的控制文件。

9.6 小结

现在你能够在—个Oracle数据库上利用可迁移表空间实施表空间恢复了。当在一个实际的

数据库环境下完成这项工作时，应遵循同样的步骤，但需要花更多的时间来细心研究表空间恢复对数据库其他部分造成的影响。主要的考虑因素是恢复和未恢复的表空间在应用级上不一致的可能性。理解这些应用相关性有助于成功完成TSPITR。现在你应当对保护数据库充满信心了。

为了获得有关表空间导出/导入方面的更多信息，请查阅《Oracle8i Utilities manual》。第1、2章阅读《Oracle8i Administrator's Guid》的第9章和《Oracle8i Backup and Recovery guide》的第7章，了解更多有关在数据库间迁移表空间和使用TSPITR方面的信息。

习题

请回答以下问题，以加深对本章中的概念和练习的理解。

1. 在哪些情况下TSPITR可用于恢复？（选出所有正确答案）
 - A. 当某个表不慎被删除或截短时。
 - B. 当表数据被逻辑毁坏时。
 - C. 在数据库的某部分某个批处理任务运行不正确时。
 - D. 当一个非常大的数据库的完全恢复可能要持续很长的时间，而同时又仅需要几个关键对象时。
2. 哪两个数据字典视图能够帮助确定TSPITR可能对数据库造成的影响？
 - A. TS_PITR_CONFIRM
 - B. TS_PITR_OBJECTS_TO_BE_DROPPED
 - C. TRANSPORT_SET_VIOLATIONS
 - D. TS_PITR_OBJECTS_LOST
3. 只要在一个辅助实例上已经恢复了一个表空间集，就可以从辅助实例迁移整个表空间，或从辅助实例导出单个的对象。
 - A. True
 - B. False
4. 为了利用Oracle导出应用程序导出一个表空间的元数据，需要哪些关键字参数？（选出所有正确答案）
 - A. TABLESPACES
 - B. TRANSPORTABLE
 - C. TRANSPORT_TABLESPACE
 - D. DATAFILES
5. 为了利用Oracle导入应用程序导入一个表空间的元数据，需要哪些关键字参数？（选出所有正确答案）
 - A. TRANSPORTABLE
 - B. TABLESPACES
 - C. TRANSPORT_TABLESPACE
 - D. DATAFILES

答案

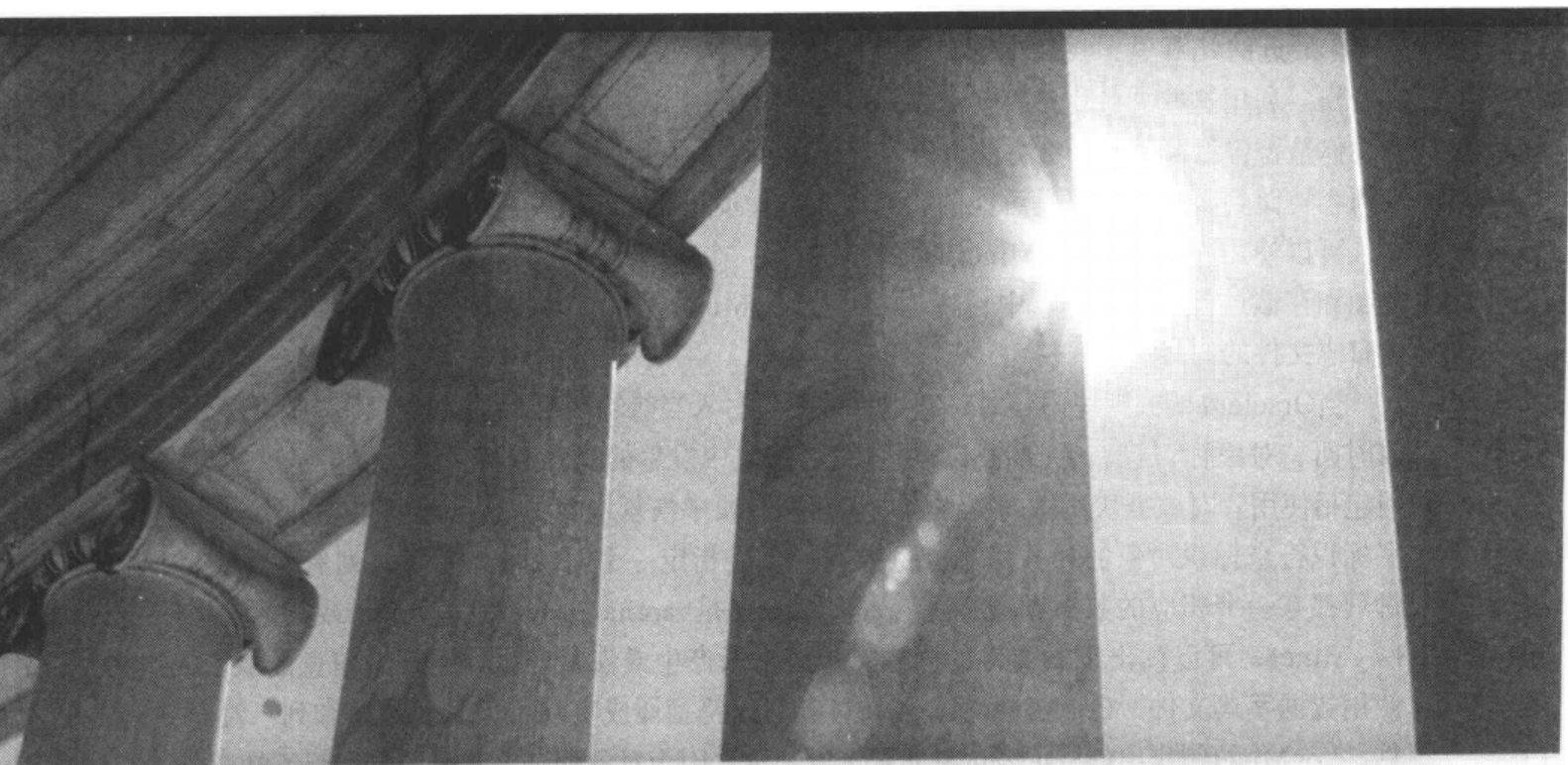
1. A, B, C, D。TSPITR可以用于多种故障情况下的恢复。必须就TSPITR的影响和恢复可能对数据库中其余表空间造成的影响做出判断。

2. B, C。可以使用这些视图回答两个问题：如果实施了TSPITR，会不会丢失什么对象；是否有什么对象类型会妨碍TSPITR的成功实现。

3. True。只要恢复了辅助实例上的一个表空间，就有了选择余地，可以迁移整个表空间，或是导出恢复表空间内的对象。

4. A, C。TRANSPORT_TABLESPACE参数指定导出应用程序运行于表空间模式。TABLESPACES参数定义出将提取哪个表空间中的元数据。

5. B, C, D。TRANSPORT_TABLESPACE参数指定导入应用程序运行于表空间模式。TABLESPACES参数定义出将要提取哪个表空间中的元数据。DATAFILES参数定义出那些包含表空间数据的文件。



第10章 LogMiner

在最近的几章里，我讨论了归档重做日志文件，对归档重做日志文件进行了备份并用于恢复操作。利用这些文件把改变传递到一个备用数据库中并将一个表空间及时恢复到一个特定的点。你是否曾思考过有什么方法可以直接观察一个Oracle重做日志文件的内部结构？将日志文件的内容转储到跟踪文件里是可能的，但这个过程很难解释清楚。从Oracle8i开始提供了另一种方法，可以使用一个名为LogMiner的工具来查看一个或多个日志文件的内容，包括一些数据字典视图和存储过程。在这一章中，将练习安装LogMiner，并使用LogMiner来观看和分析数据库重做日志文件的内容。

当Oracle改变数据块内容时，它把重做信息写入当前的联机重做日志文件。重做文件包括更改时间、对象标识符、更改的SCN号、在数据块中发生的操作和其他的重要信息。重做文件不但包括使用者对数据块所做的更改，也包括回退段中恢复块的更改。Oracle在数据字典中使用数字标识符来标识对象的相关信息，例如为一个表指定一个对象号，并且每列有一个列标识符。每列都有一个相应的数据类型标识符，表示该列是varchar2、date、number或其他变量。使用LogMiner，可以读出重做文件的内容，然后重新产生或恢复产生原始重做信息的SQL表达式。使用数据字典文件，LogMiner将Oracle对象标识符翻译成实际可以看懂的表和列。没有字典文件，LogMiner制作出的SQL表达将非常难以理解，因为对象的名字和列的名字都不知道。

在开始学习前，我将定义一些必要的术语。在进行本章练习中的任务时，将会对每一个术语进行更为详尽地解释。

- **当前数据库 (current database)**: 当LogMiner使用产生日志文件的服务器来分析重做日志文件时，重做日志文件来自当前数据库。如图10-1所示，LogMiner的过程和视图，可以通过使用创建日志文件的数据库来访问。
- **外部数据库 (foreign database)**: LogMiner可以分析异于当前数据库的其他数据库产生的重做日志文件。那个产生重做日志文件的数据库就是一个外部数据库。
- **字典文件 (dictionary file)**: LogMiner使用字典文件将Oracle内部对象标识与表名称、列、数据类型以及其他对象关联起来。字典文件必须由创建重做日志文件的数据库产生。

有很多情况下需要使用LogMiner。如果数据表中的数据被莫名其妙地改变了，使用LogMiner可以调查那个改变的运行细节，也可以使用LogMiner来撤销这些更改。可以使用LogMiner来检查在一个或多个表中SQL改变发生的次数，从而检查表上的工作量。通过进一步的检查，LogMiner可以指出一个错误的DROP TABLE或者一个DDL语句发生的准确时间和SCN（但是不记录这些原始的语句）。

在使用LogMiner前，应该理解以下要点：

- 可以在一个已经加载或者未加载的数据库上使用LogMiner。
- 分析一个来自外部数据库的重做日志文件时，当前数据库的块大小必须和外部数据库的一样大或者更大，因为在热备份模式，表空间中一个块的首次改变是作为一个单独的重做记录在重做流中创建一个完整的块镜像。如果LogMiner使用较小的块尺寸的数据库读取重做，

就会发生错误，因为单独的重做记录无法放入一个块中。

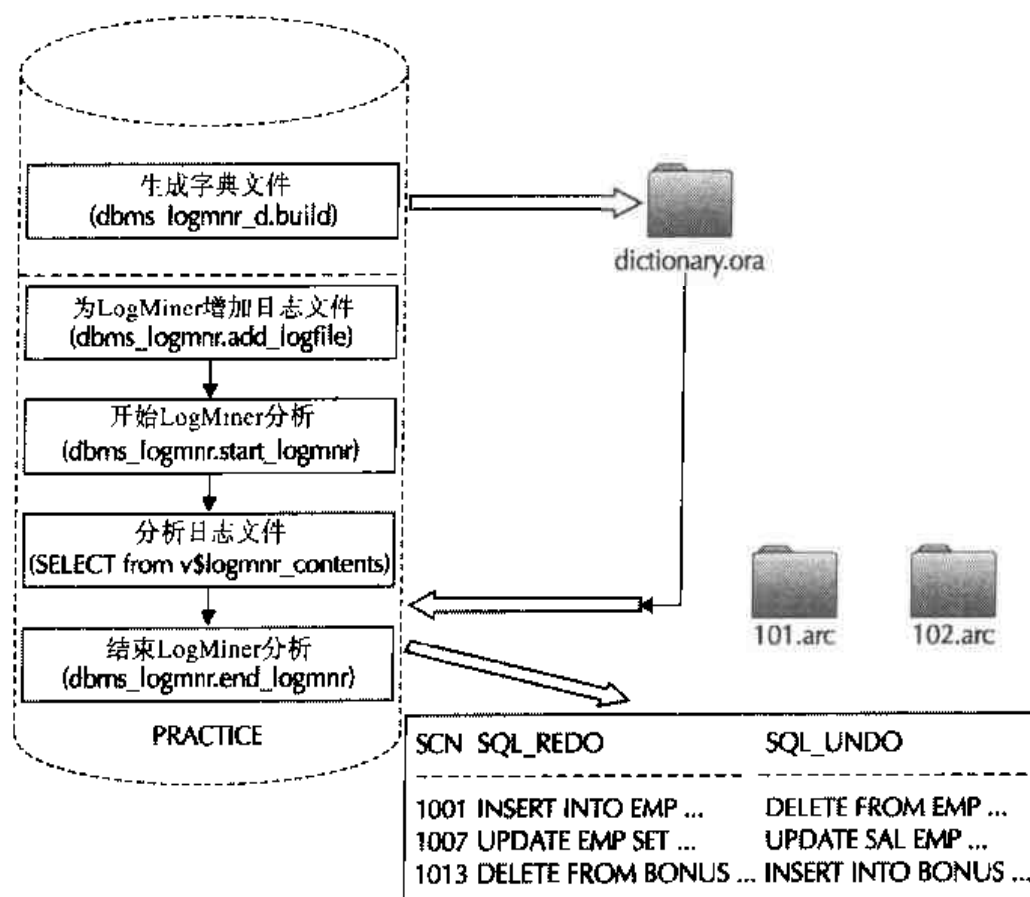


图10-1 PRACTICE数据库上的LogMiner操作

- 虽然LogMiner不解释重做日志的DDL语句，如DROP TABLE这样的DDL语句，但仍旧会在数据字典里创建DML。这些字典DML语句可以被用来检查对数据库发布的DDL命令。在Oracle9i中增强了对DDL的跟踪功能。
- LogMiner将不会重构原始的无日志的SQL操作，但是在数据字典上的DML操作结果将被记录到日志文件里。
- 虽然LogMiner可以偶然被用来进行日志文件分析，但不能当作常规意义上使用的一种功能，特别是在一个产生了庞大数量重做信息的数据库里挖掘重做信息。使用LogMiner分析庞大数量的重做信息，是一件非常耗费时间的东西。
- 如果LogMiner被用来从其他数据库的日志文件里挖掘信息，无论源数据库还是被分析的数据库都必须同时使用相同的硬件平台和操作系统，而且源数据库和被分析的数据库都必须使用相同的字符集。
- LogMiner无法分析某些数据库对象或数据类型产生的重做，例如按索引组织的表、成簇的表/索引、非标量数据类型和链接的行。
- 尽管可以创建数据字典DML，但LogMiner无法从直接的路径插入操作中生成原始的SQL。
- LogMiner将显示未提交的事务，因为重做日志包含提交的和未提交的数据。直到你查询受影响的对象才决定数据是否被提交。

在这一章中，将创建一个字典文件，产生一些数据库操作并使用LogMiner分析重做日志。

10.1 使用LogMiner分析重做

通过LogMiner读取重做日志事务涉及到使用Oracle提供的包过程、数据字典视图和一个数据字典外部文件。简单地说，下面是应用LogMiner分析重做日志文件的操作：使用LogMiner存储过程创建一个外部数据字典文件；然后，使用另一个存储过程创建一个被分析的重做文件的列表；最后，执行另一个存储过程来启动LogMiner。在完成这些步骤以后，就准备从一个显示了日志文件内容的数据字典视图里进行选择。当查询这个视图时，Oracle读取日志文件，然后以特定格式返回结果（日志文件的内容实际上不保存在一个永久的数据库对象里）。一旦分析了列出的日志文件，就调用另一个存储过程停止LogMiner。在表10-1中列出了重要的LogMiner存储过程和视图。

表10-1 重要的LogMiner过程和视图

过 程	用 途
dbms_logmnr_d.build	创建一个数据字典文件
dbms_logmnr.add_logfile	在列表中增加日志文件以供分析
dbms_logmnr.start_logmnr	使用一个可选的字典文件和前面确定要分析的日志文件来启动LogMiner
dbms_logmnr.end_logmnr	停止LogMiner分析
视 图	用 途
v\$logmnr_dictionary	显示用来决定对象ID名称的字典文件的信息
v\$logmnr_logs	在LogMiner启动时显示被分析的日志列表
v\$logmnr_contents	LogMiner启动后，可以使用这个视图，在SQL提示符下输入SQL语句来查询重做日志的内容

当执行本章中的练习时，你将会看到这些过程、文件和视图怎样一起工作，从而可以查看数据库重做日志文件。

练习10.1：分析重做文件

本章只有一个练习，在这个练习中，你将观察SCOTT.EMP表中一行记录的特定改变的详细情况。然后将看到怎样通过分析一个特定表上记录改变操作的重做日志文件，来检查在这个表上的数据库更改操作。最后，将学会怎样为一个特定的DROP TABLE命令确定准确的时间和SCN。这三个例子示范了在一个真实的数据库环境下使用LogMiner检查重做信息的几种方式。

任务描述	时间（分钟）
1. 创建数据字典文件	5
2. 产生数据库操作	5
3. 为分析指定日志文件	5
4. 启动LogMiner	5
5. 分析日志文件内容	15
6. 关闭LogMiner	5
总计时间	40

这个练习中的任务可以作为成功设置LogMiner和进行分析的向导。我将覆盖被LogMiner使用的文件、过程和视图来查看PRACTICE数据库中的重做信息。在任务2中，将执行一些DML和DDL（在任务5中也有类似操作）。如果能使用LogMiner视图定位这些特定的数据库变化，这个练习就成功完成了。

任务1：创建数据字典文件

使用LogMiner时，创建数据字典文件是一个可选步骤。字典文件极大地增加了日志文件内容的可读性。以下是一个没有数据字典文件的日志文件：

```
SQL> SELECT sql_redo FROM v$logmnr_contents;
SQL_REDO
-----
set transaction read write;
insert into UNKNOWN.Objn:4273(Col[1]) values (HEXTORAW('c1022'));
commit;
```

以下是一个有字典文件情况下的日志文件：

```
SQL> select sql_redo from v$logmnr_contents;
SQL_REDO
-----
set transaction read write;
insert into TINA.DATE_LOG(C1) values (1);
commit;
```

数据字典文件建立了Oracle内部对象ID数据到表、列及其他数据类型名称之间的对应关系。数据字典文件是一个文本文件，相当于数据字典的相关部分。文件包括SQL语句，看上去它们可以部分地重新创建数据字典。这些语句被LogMiner使用并且不能在任何数据库上运行，因为这可能破坏数据字典的内容。LogMiner过程使用这个文件将对象标识转化为有意义的名字。LogMiner为当前进程将字典文件的信息保存到PGA内存区，因为LogMiner也在PGA里进行数据处理，所以如果字典文件很大，就应确保系统必须具备足够的内存空间。如果一个字典文件没有被使用，则需要较少的PGA内存空间。然而，应明白在没有字典文件的情况下将这些数字转换成有意义的名称非常困难。因此在这个任务中将创建一个数据字典文件，然后指示LogMiner应用该文件把内部的数字翻译成任务4中分析日志文件时能够识别的名称。

- 字典文件应该与日志文件的日期一致。如果被LogMiner分析的一个对象无法在字典文件中找到，LogMiner将无法显示数据库对象的表和列的名称。
- 建立字典文件的数据库必须和建立日志文件的数据库相同。例如，不能使用克隆数据库中的字典文件来分析PRACTICE数据库中的日志文件。

在建立数据字典文件之前，PRACTICE数据库必须被写到服务器硬盘的一个目录下。在数据库参数文件内设置一个参数UTL_FILE_DIR,这个参数允许你为Oracle可使用的PL/SQL文件I/O定义一个或多个目录。可以在这个参数文件里连续地分行输入UTL_FILE_DIR，以定义更多的目录。把下面这个练习目录加到参数文件中：

```
utl_file_dir=/oradata/PRACTICE/backup/ch10
```

因为这个参数不是动态数据库参数，所以必须重新启动数据库才能使设置生效。一旦重启

了数据库，登录并查看设置是否已经生效，命令如下：

```
LINUX> sqlplus /nolog
SQL> CONNECT sys/practice;
SQL> SHOW PARAMETER utl_file_dir;
```

为了创建一个数据字典文件，可以执行DBMS_LOGMNR_D包中的BUILD过程。这个过程查询当前的数据库字典表，然后创建一个包含其内容基于文本的文件。这个文件在调用的过程的目录中生成。执行过程如下：

```
SQL> EXECUTE dbms_logmnr_d.build( -
dictionary_filename => 'dictionary.ora', -
dictionary_location => '/oradata/PRACTICE/backup/ch10');
```

注意每行末尾的连字符“-”表示一个SQL*Plus命令的多行形式。这个过程执行之后，在/oradata/PRACTICE/backup/ch10目录下打开文件dictionary.ora，浏览它的内容。

任务2：产生数据库操作

在第二个任务中，产生以后将在日志文件中找到的SQL操作。更改EMP表中的一行记录，确认行记录被插入到TINA.DATE_LOG表中，删除SCOTT.BONUS表。

首先，确认在TINA.DATE_LOG表中生成了行记录。如果需要，可以参照第3章的相关内容。

其次，改变SCOTT.EMP表中的一行记录。使用下面的SQL语句更改其中一个雇员的薪水和佣金：

```
LINUX> sqlplus /nolog
SQL> CONNECT scott/tiger;
SQL> UPDATE emp SET sal = 3000, comm = 5000
2 WHERE empno = 7844;
SQL> COMMIT;
```

后面将可以挖掘这个SQL语句，来了解如何使用LogMiner。

再次，删除SCOTT.BONUS表。现在BONUS表中没有任何记录了，在本书的其余部分将不再使用该表。

```
SQL> DROP TABLE bonus;
```

使用钟表或者使用SQL*Plus命令“set time on”，记录这些SQL命令运行的时间。set time on命令生效后给SQL提示增加一个时间值，如下所示：

```
SQL> set time on;
17:21:20 SQL> DROP TABLE bonus;
Table dropped.
17:21:26 SQL>
```

UPDATE和DROP (字典中对应DML)语句的重做信息将被保存在当前联机重做日志文件中。从v\$log视图里找出日志文件的序号。最后，强制一个日志切换，使当前的联机日志归档。运行LogMiner来分析这个新归档的日志文件，试验一下是否能从这个日志中找到关于SQL更新和删除表的信息。

```
17:22:01 SQL> SELECT sequence# FROM v$log WHERE status = 'CURRENT';
17:22:01 SQL> ALTER SYSTEM SWITCH LOGFILE;
```

注意来自v\$log视图的选择。数据库改变期间，当前日志的序号很可能包含那些改变的重做信息。在这一章的讨论中，日志切换前的当前日志序号将是102，因此日志切换期间产生的归档日志文件的名称是102.arc。

任务3：为分析指定日志文件

LogMiner将分析指定的日志文件，因此，必须创建一个可供分析的日志文件列表。使用Oracle提供的DBMS_LOGMNR包中的ADD_LOGFILE过程来添加日志文件。增加的日志文件将显示在v\$logmnr_logs视图里。

```
LINUX> sqlplus sys/practice
SQL> SELECT db_name, thread_sqn, filename
2 FROM v$logmnr_logs;
```

在向这个列表中添加可供分析的日志文件之前，不会从选定的视图返回任何行记录。前一个任务中SQL所做的改变保存在序号为102的日志文件里，那么这里在LogMiner分析前添加这个日志文件和前面的两个文件。

```
BEGIN
  dbms_logmnr.add_logfile(
    logfile => '/oradata/PRACTICE/backup/ch10/100.arc',
    options => dbms_logmnr.NEW);

  dbms_logmnr.add_logfile(
    logfile => '/oradata/PRACTICE/backup/ch10/101.arc',
    options => dbms_logmnr.ADDFILE);

  dbms_logmnr.add_logfile(
    logfile => '/oradata/PRACTICE/backup/ch10/102.arc',
    options => dbms_logmnr.ADDFILE);
END;
/
```

当把第一个日志文件添加到列表中，标志着一个新的列表就产生了。add_logfile过程第一次调用时，选项被设为dbms_logmnr.NEW，这就创建了一个供LogMiner分析的新的日志文件列表。之后调用add_logfile过程时，使用ADDFILE选项为这个新建的列表添加文件。每一次使用NEW选项时，日志文件列表将被重置为一个名称来自调用add_logfile的日志文件。如果再次指定新的参数，任何在此之前添加的文件都将从这个列表中删除。你是否注意到，当运行这些过程的时候，返回的速度有多快？这个过程调用没有分析文件，只不过把文件加入列表供以后分析。

查询v\$logmnr_logs视图，将看到已经添加到列表中的文件：

```
SQL> SELECT db_name, thread_sqn, filename FROM v$logmnr_logs;
DB_NAME  THREAD_SQN  FILENAME
-----
PRACTICE      100 /oradata/PRACTICE/archive/100.arc
PRACTICE      101 /oradata/PRACTICE/archive/101.arc
PRACTICE      102 /oradata/PRACTICE/archive/102.arc
```

可以在DBMS_LOGMNR.ADD_LOGFILE过程中通过指定REMOVEFILE选项来删除重做日志文件。在本示例中，可以确定序号100的日志文件没有包含寻找的重做信息，因此，可以将序

号为100的日志文件从可供分析的列表中删除：

```
SQL> EXECUTE dbms_logmnr.add_logfile( ~
2     logfile => 'd:\oradata\PRACTICE\archive\100.arc', ~
3     options => dbms_logmnr.REMOVEFILE);
```

任务4：启动 LogMiner

准备好字典文件和日志文件列表后，就可以准备启动LogMiner了。

```
EXECUTE dbms_logmnr.start_logmnr(
    dictfilename => '/oradata/PRACTICE/backup/ch10/dictionary.ora');
```

启动LogMiner之后，指定先前创建的字典文件的有效全路径名。

可以为这个过程有选择地指定其他几个参数。可以指定分析开始的SCN和/或结束的SCN。可以指定分析开始时间和/或结束时间。如果必须分析大量的归档日志或者只想查看很少一段时间之内的重做，这些参数将会很有帮助。在这个小型的PRACTICE数据库中，将分析日志文件列表中的全部重做。完成这个命令，就准备好分析列表日志文件的内容了。

任务5：分析重做文件内容

在v\$LOGMNR_CONTENTS视图中，包含日志文件101.arc 和 102.arc的内容。使用这个视图，可以调查重做日志来回答几个可能产生的示例问题。当查询v\$LOGMNR_CONTENTS 视图时，包括在v\$LOGMNR_LOGS中的日志文件将被顺序读出，并以v\$LOGMNR_CONTENTS视图定义的行结构返回数据。该视图有53列（其他有用的列请参见表10-2）。所有日志文件内容不是永久保存在一个表或者内存中的，因此每次选择有关V\$LOGMNR_CONTENTS的内容，日志文件就会被扫描。如果必须多次检查大量的日志文件来寻找需要的内容，那么最好使用Create Table As Select (CTAS)命令制作一个结果的表段。

表10-2 V\$LOGMNR_CONTENTS 视图中有用的列

列名称	描 述
OPERATION	重做记录中记录的操作 (INSERT,UPDATE, DELETE等)
TIMESTAMP	数据改变发生的日期和时间
SCN	特定数据变化的系统更改号
SEG_OWNER	段的所有者名称 (如果段发生了更改)
SEG_NAME	数据发生了改变的段名称
SEG_TYPE	数据发生了改变的段类型
TABLE_SPACE_NAME	变化段的表空间名称
ROW_ID	特定数据变化行的ID
USER_NAME	执行数据改变的用户名
SESSION_INFO	数据发生变化时用户进程信息
SQL_REDO	可以为重做记录重做指定行变化的SQL语句
SQL_UNDO	可以为重做记录回退或恢复指定行变化的SQL语句

在这个任务里，将使用PRACTICE数据库中的重做信息完成三件事：

- 1) 确定数据改变的细节：确认一个雇员薪水更改的细节。
- 2) 完成容量分析：检查TINA.DATE_LOG表的工作量。
- 3) 确定DDL命令的详细情况：定位SCN和删除表的时间。

v\$logmnr_contents 数据字典视图显示了重做日志文件的内容。

1. 检查数据更改的细节

数据库里的数据可能因为意想不到的原因或者因为错误而发生改变。在重做日志文件中可以找到这些更改的细节。在示例中，可能有报告显示一个雇员的薪水和佣金发生巨大改变。经理告诉你，有一个名叫Turner的销售员的薪水和佣金改变了，她希望你能查出发生了什么事，以及改变前的薪水和佣金数。她还想知道更多的细节，比如是谁做了这些改变，以及更改是何时发生的。

要回答她的问题，需要查找涉及SCOTT.EMP表的所有SQL语句。这个表中有一个表示当前薪水的SAL列和表示佣金的COMM列。要找到所有涉及这个表的SQL，可以运行如下SELECT语句：

```
SQL> SELECT operation, timestamp, scn
2    FROM v$logmnr_contents
3   WHERE seg_name = 'EMP'
4         AND seg_owner = 'SCOTT';
5         AND seg_type_name = 'TABLE';
OPERATION  TIMESTAMP SCN
-----
UPDATE 10-JAN-2002 10:42:42 340982
```

可以看到结果，在1月10日上午10:42 对SCOTT.EMP表执行了更新操作。要找出对这个表所做的具体改变，最好再查看SQL_REDO和SQL_UNDO列：

```
SQL> SELECT sql_redo, sql_undo
2    FROM v$logmnr_contents
3   WHERE seg_name = 'EMP'
4         AND seg_owner = 'SCOTT'
5         AND seg_type_name = 'TABLE';
SQL_REDO
-----
update "SCOTT"."EMP" set "SAL" = 3000, "COMM" = 5000 where ROWID =
'AAAAyQAADAAAAASAAJ';
SQL_UNDO
-----
update "SCOTT"."EMP" set "SAL" = 1500, "COMM" = 0 where ROWID =
'AAAAyQAADAAAAASAAJ';
```

SQL_REDO列定义了特定的SQL语句，可以使你得到与任务2中SQL相同的结果。注意，那个重做日志的SQL不同于前面运行的SQL。重做日志包含数据库特定行的实际数据更改，但不包含实际的SQL。SQL_UNDO列包括的SQL，允许撤销SQL_REDO的改变。实际上，它代表了更改发生前的数据值。撤销数值来自保存在重做日志的撤销改变的回退段。写入回退段的撤销数据的改变，同时也写入到当前的联机重做日志文件中。LogMiner可以显示保存在重做日志文件的所有信息，从而可以看到已经改变的列在改变前的内容。从当前的表中挑选同样的行来查看所有其他列的数值是一个好方法。当你向经理报告时，这会很有用，你可以告诉她什么数据改变了，以及确认那个数据还没有被再次改变。

你找到了数据的更改和更改发生的时间。还想弄清楚是谁做了这个更改吗？查看V\$LOGMNR_CONTENTS的USERNAME列和SESSION_INFO列。在这些列中，将找到有用的会话信息，可以帮助你进一步解释Turner的薪水和佣金是怎样被改变的。

```
SQL> SELECT username, session_info
2     FROM logmnr_contents
3     WHERE seg_name = 'EMP'
4           AND seg_owner = 'SCOTT'
5           AND seg_type_name = 'TABLE';
```

USERNAME

SCOTT

SESSION_INFO

```
-----
LoginUserName = SCOTT, ClientInfo = , OsUserName = SJONES, MachineName = SJCOMP
```

注意 许多应用软件使用独立或共享的登录来执行数据库操作。当通过分析重做来调查数据改变时，这些中间层的连接使我们难以找到有价值的会话信息。

可以看到某人使用用户名SCOTT登录并执行了更新操作，你也知道了这个嫌疑人使用名叫SJONES的操作系统登录到名叫SJCOMP的机器上。看上去所有的矛头都指向Scott！你可以把这个信息报告给你的经理，同时使用SQL_UNDO列的内容把数据恢复到改变以前的数值。不要忘记，检查当前存在于表内的该行的内容，以确保它没有改变后续的时间。

```
SQL> update "SCOTT"."EMP" set "SAL" = 1500, "COMM" = 0
2 where ROWID = 'AAAAyQAADAAAAASAAJ';
```

警告 这里数据改变的情况只是简单地用于演示的目的。在一个产品级数据库环境中，不可以在没有经过仔细思考和适当批准的情况下就改变产品数据，还必须考虑被更改表上的所有外键关联或者被其他应用维护的所有一致性约束。

2. 执行容量分析

作为一个DBA，你需要做一些性能调整或容量计划，要完成这个任务可能涉及到使用许多不同的工具和程序。作为重点之一，可以使用LogMiner来检查数据库表中发生的活动。例如，LogMiner将帮助你调查表中使用DML的次数和频率。也可以使用表审计来实现同样的功能。当更新表的时候，使用LogMiner不会降低运行时间性能，而使用表审计则会导致运行时间性能开销过大。

让我们查询一下TINA.DATE_LOG表中的SQL操作。启动LogMiner，可以检查这个表中执行插入、更新和删除的次数（SELECT语句不产生重做，因此，通过LogMiner看不到这些）。下面这个非常简单的例子为分析数据库表上DML操作提供了一些思路与方法：

```
SELECT operation, to_char(timestamp, 'HH') hour, count(*) total
FROM v$logmnr_contents
WHERE seg_name = 'DATE_LOG'
      AND seg_owner = 'TINA'
      AND seg_type_name = 'TABLE'
```

```

GROUP BY operation, to_char(timestamp, 'HH');
OPERATION HOUR TOTAL
-----
INSERT      10      6
INSERT      11      6
INSERT      12      6
INSERT      13      6

```

查询的结果显示, DATE_LOG 表每小时增加6行记录。

当必须分析大量的重做信息时,要在全面分析和分析速度之间进行权衡。添加到LogMiner列表中待分析的日志文件越多,可以分析的重做信息也就越多。然而,在LogMiner列表中太多或太大的日志文件,将会导致需要更长的时间来完成对V\$LOGMNR_CONTENTS视图访问的SQL。

3. 寻找DDL命令的细节

就像本章所作的一样,在第5章和第9章中曾模拟了一个意外删除表的操作。在那几章中,我曾提到过要知道错误删除表的时间比较困难。使用LogMiner,就可以推算出删除表语句的确切时间和SCN。可以将这个时间或者SCN用于不完全数据库恢复或者表空间时间点恢复(TSPITR)。

如前所述,LogMiner只是将重做日志信息翻译成数据处理语言(插入、更新、删除),在V\$LOGMNR_CONTENTS视图中将不会找到DROP TABLE语句。既然DROP TABLE语句在数据字典表上运行了DML,那么可以从SYS.OBJ\$或者SYS.TAB\$数据字典表里搜索发生删除操作的视图。

```

SQL> SELECT seg_name, operation, scn, timestamp, count(*)
2   FROM v$logmnr_contents
3   WHERE operation = 'DELETE'
4   GROUP by seg_name, operation, scn, timestamp
5   ORDER by scn;

```

SEG_NAME	OPERATION	SCN	TIMESTAMP	COUNT(*)
FET\$	DELETE	362948	2002-01-10:07:59:11	1
COL\$	DELETE	362952	2002-01-10:07:59:16	4
OBJ\$	DELETE	362952	2002-01-10:07:59:16	1
TAB\$	DELETE	362952	2002-01-10:07:59:16	1
SEC\$	DELETE	362954	2002-01-10:07:59:16	1
UET\$	DELETE	362954	2002-01-10:07:59:16	1

从以上SQL结果可以看出:在1月10日上午7:59,一个对象,一个表,和四个列(SCOTT.BONUS有四列)被从数据字典中删除了。这个数据字典对象删除操作可能就是删除bonus表语句。接下来,查看DROP TABLE命令的SQL_REDO:

```

SQL> SELECT sql_redo FROM v$logmnr_contents
2   WHERE scn = 362952 and seg_name = 'OBJ$';
SQL_REDO
-----
delete from SYS.OBJ$ where OBJ# = 13079 and DATAOBJ# = 13079 and OWNER# =
38 and NAME = 'BONUS' and NAMESPACE = 1 and SUBNAME IS NULL and TYPE# = 2
and CTIME = TO_DATE('2002-01-10:07:59:16', 'DD-MON-YYYY HH24:MI:SS') and
MTIME = TO_DATE('2002-01-10:07:59:16', 'DD-MON-YYYY HH24:MI:SS') and STIME

```



```
= TO_DATE('2002-01-10:07:59:16', 'DD-MON-YYYY HH24:MI:SS') and STATUS = 1
and REMOTEOWNER IS NULL and LINKNAME IS NULL and FLAGS = 0 and ROWID =
'AAAAASAAABAAAGXKAAA';
```

在SQL_REDO列的输出中有OBJ#。这个和OBJ#相关联的数值与存放在数据字典表TAB\$.OBJ#中的Oracle ID一致（在DBA_OBJECTS视图中也可以看到OBJECT_ID）。要将被删除表和删除表的SQL语句发生的实际时间和SCN联系到一起，需要知道表在何时被删除以及表中包含了多少列。使用这个信息，就可能找到准确的删除表的时间以及用户信息——如果有必要知道的话。那个对象ID可以在字典文件里找到（在任务2里创建这个文件）。在一个编辑器中打开字典文件，同时查找单词BONUS，将会看到一个如下所示的插入语句：

```
INSERT INTO OBJ$_TABLE VALUES (13079,13079,38,'BONUS',1,'',2,
to_date('01/10/2001 06:25:28', 'MM/DD/YYYY HH24:MI:SS'),
to_date('01/10/2001 06:25:28', 'MM/DD/YYYY HH24:MI:SS'),
to_date('01/10/2001 06:25:28', 'MM/DD/YYYY H24:MI:SS'),
1,'','','0,,,,','',);
```

如果没有在字典文件中找到有关BONUS的记录，就不可能在数据库中已有这个表的情况下创建这个字典文件。如果是这种情况，就必须根据自己的判断拼凑出被删除的表的数据。在SQL_REDO列内的OBJ#数值将和第一次为删除表而插入到数据字典文件的数值完全相同（在上面的输出中，BONUS表的对象ID是13079）。如果完全把这些信息关联起来，就可以确信删除表的时间为2002-01-10:07:59:16，SCN为362952。如果必要，使用这些数值就可以执行TSPITR。

简而言之，你可以通过将数据字典文件的对象编码与删除操作发生的SYS.OBJ\$数据字典表的SQL_REDO列的对象标识符相匹配，以找到删除表命令发生的时间和SCN。

任务6：关闭 LogMiner

使用LogMiner后，要释放这个程序使用的资源。LogMiner为重做文件的分析分配了内存和数据结构。当完成LogMiner工作之后，V\$LOGMNR_CONTENTS的内容就不再可用。如果需要更进一步分析重做日志文件的内容，使用V\$LOGMNR_CONTENTS视图的内容创建一个永久的数据库表将非常有帮助。这样，在终止LogMiner后还可以检查重做日志文件的内容。创建命令如下：

```
SQL> CREATE TABLE logmnr_contents AS SELECT * FROM v$logmnr_contents;
```

一旦完成了重做日志的检查，运行DBMS_LOGMNR中的END_LOGMNR。

```
EXECUTE dbms_logmnr.end_logmnr;
```

结束LogMiner时，释放为重做日志分析分配的会话内存。

Oracle 9i的增强特性

Oracle 9i 已经改进了LogMiner的一些性能。大多数的改进允许你提取以前在Oracle 8i中不可用的重做信息。现在，在Oracle 9i中LogMiner可以做以下事情：

- 分析包含链接行和成簇表/索引的重做。
- 提供对数据字典信息的三种访问方式：

- 与Oracle 8i一样，字典对象信息可以保存为ASCII文本文件。
- 字典对象信息可以保存在数据库日志文件的重做流中。
- Logminer可以联机访问数据库的字典文件。
- 跟踪DDL语句。
- 如果数据库运行在归档模式下，分析直接加载操作中产生的重做信息。
- 只显示提交的事务，不显示未提交的事务。

最后，Oracle企业管理器提供了使用图形用户界面访问LogMiner的工具，叫做LogMiner Viewer。

10.2 疑难解答

当使用LogMiner遇到疑难问题时，可能是以下错误中的一个。这些注释可以给你在解决日志分析问题上指出正确的方向。

1) ORA-06532: Subscript outside of limit

ORA-06512: at "SYS.DBMS_LOGMNR_D", line 793

当创建了一个字典文件时，可能遇到这个错误。要修正这个错误，需要改变\$ORACLE_HOME/rdbms/Admin目录下的Oracle脚本。打开这个名为dbmslmd.sql的文件，改变下列行：

```
TYPE col_desc_array IS VARRAY(513) OF col_description;
```

为

```
TYPE col_desc_array IS VARRAY(700) OF col_description;
```

改变这个脚本文件后，以用户SYS再次运行dbmslmd.sql。(我上Metalink并在搜索屏幕输入“ORA-06532:Subscript outside of limit”时，发现了这个解决方案。其他人和我一样也遇到过同样的错误，所以我可以迅速地解决我的问题。尽管如此，任何时候修改Oracle提供的脚本时都要非常小心。)

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

运行LogMiner的顺序非常重要。必须为LogMiner指定日志文件，并且在能查询LogMiner视图内容之前，启动LogMiner。

3) ORA-01292: no log file has been specified for the current LogMiner session

在能查询LogMiner视图前，必须先添加一个日志文件到分析列表中，然后再启动LogMiner。接下来可以从V\$LOGMNR_CONTENTS中进行查询。

10.3 小结

偶尔地联机重做日志文件的分析可以帮助你找出表数据修改的时间或者进行Oracle数据库容量分析。LogMiner提供了一个便捷的工具来查看Oracle 8 或 8i 数据库重做日志的内容。本章介绍了LogMiner和它的组成，示范了如何提取归档日志文件来找出什么时候以及谁改变了表数据，当一个表被删除时，你可以更准确地进行一些形式的恢复以及说明一段时间内的DML活动的次

数。已经清楚地说明了使用数据字典文件的好处，LogMiner的限制也被重点提示了。最后，列出了一个Oracle 9i中的新特色的列表，这些特点使 LogMiner 更通用、更易用。

要获得更多关于LogMiner的信息，可以参看《Oracle 8i Administrators Guide》中的第7章(“Managing Archived Redo Logs”),也可以查看《Oracle8i Supplied PL/SQL Packages Reference》中关于LogMiner过程的细节，以及《Oracle8i Supplied PL/SQL Packages Reference》中关于LogMiner视图的细节。

习题

回答下列问题来加强对本章概念的理解。

1. Oracle的LogMiner中字典文件的用途是什么?
 - A. 在LogMiner分析时，将对象ID号翻译为可读的对象名称
 - B. 查找拼写正确的单词
 - C. 备份SYSTEM表空间
 - D. 在LogMiner分析时保存错误
2. 哪个视图包含当前分析过的重做日志文件的内容?
 - A. v\$LOGMNR_DICTIONARY
 - B. v\$LOGMNR_LOGS
 - C. v\$LOGMNR_CONTENTS
 - D. v\$LOG
3. LogMiner 可以在分析时从重做日志文件里找到DML 和 DDL。
 - A. True
 - B. False
4. 什么过程将日志文件添加到LogMiner分析列表中?
 - A. DBMS_LOGMNR_D.BUILD
 - B. DBMS_LOGMNR.ADD_LOGFILE
 - C. DBMS_LOGMNR.START_LOGMNR
 - D. DBMS_LOGMNR.END_LOGMNR
5. LogMiner v\$LOGMNR_CONTENTS视图的内容只在运行LogMiner时可用，当LogMiner结束后不可用。
 - A. True
 - B. False

答案

1. A。LogMiner 使用字典文件将Oracle 内部对象ID翻译为SQL表达中的对象名称和列名称。
2. C。选择了v\$LOGMNR_CONTENTS后，可以顺序读取在LogMiner 分析列表中的文

件，以及有选择地把Oracle 对象ID 翻译为可读的名称。

3. False。LogMiner 不会找到改变对象定义的DDL命令。它将找到改变表数值的SQL，根据SQL对数据字典表的改变，可以推理出一些DDL语句。

4. B。DBMS_LOGMNR.ADD_LOGFILE过程指定了日志文件或者要读取的日志文件。可以使用视图V\$LOGMNR_LOGS 来查看指定的日志文件的信息。

5. True。虽然在数据库上运行LogMiner时可以有多种方式，但V\$LOGMNR_CONTENTS 的内容只在LogMiner 启动后有效。如果分析不同的日志文件，每个会话将看到不同的数据值。



第三部分 服务器管理恢复

第11章 RMAN 配置

恢复管理器 (Recovery Manager, RMAN) 是 Oracle 的一个工具, 该工具可以用来备份、还原以及恢复 Oracle 数据库。从 Oracle 8.0 版本开始引入 RMAN, 从此改变了 DBA 手工备份、恢复数据库的方法。RMAN 是一个将用户命令解释为 PL/SQL 的命令行解释器, 解释后, 这些 PL/SQL 命令在需要操作的数据库上进行备份、还原以及恢复操作。用 RMAN 创建的备份可以存储在磁盘或者磁带介质上, 备份的相关信息被记录在备份数据库的控制文件中和一个被称为恢复目录 (Recovery Catalog) 的可选存储区中。使用不同的数据库视图或者 RMAN 命令, 可以查询控制文件或者恢复目录以查验备份的状态。RMAN 被称为服务器管理的恢复是因为它负责处理绝大多数备份、还原以及恢复的工作。

本章将介绍 RMAN 的体系结构、使用的术语, 以及如何配置 RMAN。本书后续的章节将描述如何使用基于本章配置的 RMAN 来完成备份与恢复任务。

11.1 RMAN 的体系结构

在开始介绍 RMAN 之前, 应首先了解它的体系结构以及各个组件之间是如何交互的, 如图 11-1。下面是对 RMAN 各个组件的描述:

- **RMAN 执行体**: 用于备份、还原和恢复数据库的命令行工具。该执行体将翻译用户提交的一系列命令。这些命令步骤将在目标数据库上执行 PL/SQL。RMAN 执行体也负责用新的信息更新恢复目录。该程序在安装 Oracle 服务器时自动安装, 并且可以从数据库备份的服务器上进行调用, 或者被其他的主机调用。
- **Recover.bsq**: 组成 RMAN 执行体的一个独立文件。该文件驻留在 \$ORACLE_HOME/rdbms/admin 目录下, 而且必须对 RMAN 可见。该文件包含了结构信息及其他 RMAN 使用的数据 (例如函数或者 SQL 游标)。
- **目标数据库**: 被备份、还原或恢复的数据库。
- **目标数据库控制文件**: 被备份、还原或恢复的数据库的控制文件。只要在数据库上进行 RMAN 备份, 备份的信息都要保存在目标数据库控制文件中。为了防止控制文件变得太大, 在一段时期后就会覆盖过时的备份记录。
- **通道**: 允许 RMAN 执行体从目标数据库向备份介质传送数据的通信信道。当通道打开时, 就创建一个到服务器的服务会话进程 (只有当用 SQL*Plus 连接数据库时才会创建会话进程)。然后, 在执行 RMAN 命令时, 利用该通道处理数据库向内、向外的数据传输。
- **恢复目录**: 可选用户模式, 包含了 RMAN 信息但存储在其他数据库 (而不是目标数据库) 中。该目录是表、视图和 PL/SQL 包的集合, 而这些都包含在与备份的数据库分离的服务器和数据的用户模式中。该目录存储了与控制文件 (有辅助的存储脚本) 中相同的备份信息。在该目录中的备份信息比较笼统 (尽管可以手工清理), 而在控制文件中的备份信息则更清晰。恢复目录可以保留一个 RMAN 的所有备份的历史资料库, 可以存储 RMAN 脚本,

提高对自动还原和RMAN恢复功能的利用率。

- **介质管理层 (MML)**: 允许RMAN对供应商提供的磁带设备进行读写的软件层。为了向磁带备份, RMAN使用介质管理层 (MML) 与物理的磁带驱动设备通信。MML将数据从备份的目标数据库传送到介质管理服务器 (例如Legato Networker, Veritas Netbackup等), 再由介质管理服务器将数据存储到磁带上。如果是向磁盘备份, 就不需要MML。同样, RMAN也是通过MML从磁带还原文件。

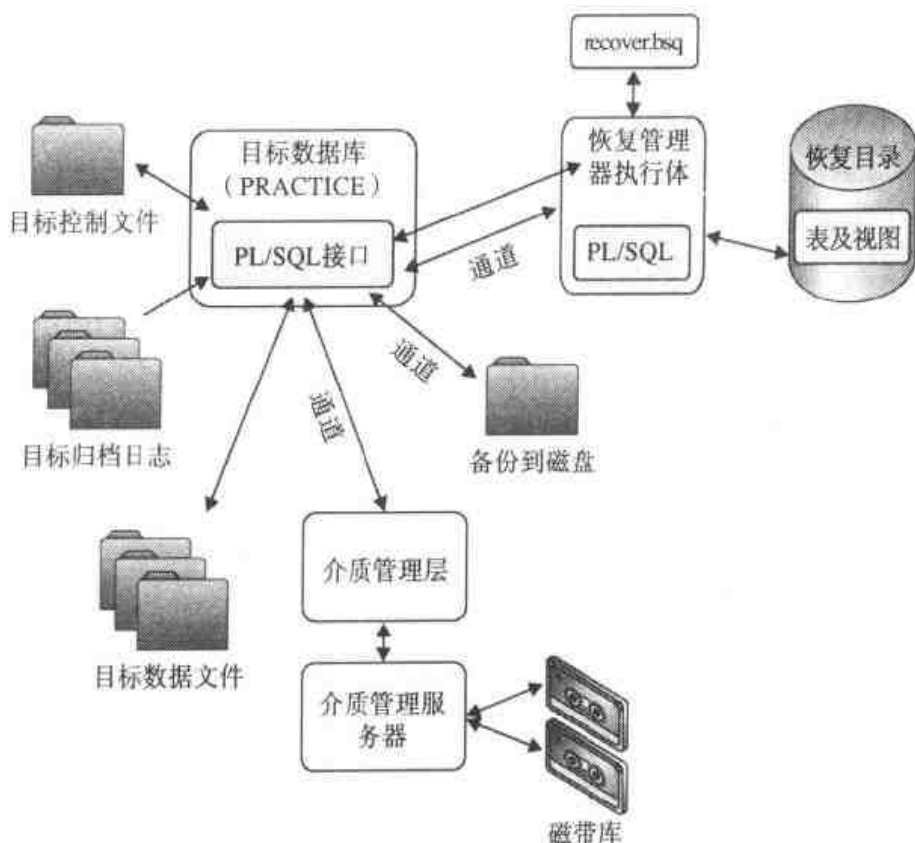


图11-1 RMAN的体系结构

11.2 RMAN的重要特性

前面介绍了主要的组件, 下面研究一下RMAN如何使备份与恢复诸多Oracle数据库变得容易。RMAN的许多特性使它在处理备份与恢复方面比前面几章所讲述的用户管理恢复技术更合适。

- **自动备份、还原与恢复** 如果由增加数据文件、删除表空间或创建更多的归档日志文件等引起了物理数据结构的改变, RMAN将自动备份最新生成的结构。正如前面所讲的, 数据库的控制文件包含了数据库的物理结构。每次要备份数据库时, RMAN都读取控制文件以确保备份了正确的结构内容 (数据文件、控制文件和归档日志文件)。
- **自动备份记录** RMAN自动将每次进行的备份都记录在目标数据库控制文件 (记录次数受限制) 和可选的恢复目录中。这就意味着, RMAN知道哪个备份需要还原以及在恢复时需要哪个日志文件。这样减轻了DBA保留准确的备份记录的负担。
- **增量备份** 用用户管理备份来备份数据文件, 整个文件都必须被拷贝到其他位置上 (磁盘

或磁带)。这就是说,如果数据库占用了200GB的磁盘空间则数据库备份也将需要占用200GB的空间。由RMAN提供的增量备份可以只备份使用了的数据块(与之相对的是空的、未被使用的数据块),或者只备份上次增量备份以来被改变的数据块,这样将减少备份的数据量并提高恢复的性能。

- **由NOLOGGING操作恢复** 当对某数据文件中的对象进行了NOLOGGING操作时,Oracle建议刷新该数据文件的备份。如果必须根据在NOLOGGING操作期间生成的日志文件向还原文件应用重做操作,则这些被改变了的数据块中的数据就不可访问了,原因是,从NOLOGGING操作应用重做记录将使这些数据块受到损坏。使用RMAN执行增量备份,这些在NOLOGGING操作中被改变的数据块将被备份。当使用增量备份恢复时,被NOLOGGING操作改变的数据块将被整个还原。因此,在NOLOGGING操作以前的重做是不需要的,而且仍然可以访问改变的数据块。
- **损坏检查** RMAN可以在备份期间检查每个数据块的损坏类型(介质损坏、校验和非法以及逻辑结构损坏)。这些损坏检查也可以在恢复过程中执行。任何在备份期间发现的损坏的块都将在目标数据块的alert.log中给出报告,而且也可以在目标数据库的V\$BACKUP_CORRUPTION和V\$COPY_CORRUPTION视图找到。
- **智能的打开数据库备份** 在打开的数据库备份期间,RMAN不需要每个表空间都被设置为热备份模式(即不需要太多的表空间BEGIN/END备份语句)。因此,打开数据库备份不需要类似用户管理打开数据库备份那样增加重做生成的数量,这样RMAN打开数据库备份速度比较快,而且不需要额外的重做动作。
- **性能调整提高** RMAN允许调整备份、还原和恢复操作的性能。可以并行方式操作数据库、复用读写磁带或磁盘设备、限制备份文件的读取速度等方式减少I/O系统的工作。
- **简易的归档日志备份** 与数据文件类似,归档重做日志文件可以通过RMAN轻松备份。没有必要手工生成一系列备份所需要的归档日志文件,而后又都删除——RMAN能完成这一切工作。在恢复期间,RMAN检查需要哪个日志文件,然后还原并应用。

正是具有了这些特性,与传统的用户管理技术相比RMAN是恢复与备份的更好方法,在本章以及后续三章中,将详细解释这些特性。

11.3 恢复管理器的目录

对于数据库管理员而言,保护Oracle数据库是首要的工作,经常进行备份操作已经成为习惯。使用用户管理恢复,必须了解备份存储在什么位置以及备份包含的内容。当需要恢复数据时,DBA必须迅速找到并还原正确的备份、找到恢复所需要的归档日志文件。而使用服务器管理恢复的最大优点就是,备份的记录被存放在控制文件或可选的恢复目录中。因此,在还原与恢复过程中,只要掌握了正确的RMAN命令,RMAN就能将这些命令解释执行为一系列数据库恢复所需要的数据文件和归档日志文件。

既然所有的RMAN信息都存储在目标数据库的控制文件中,那么为什么还要额外努力去维护RMAN的目录呢?请参见表11-1,其中罗列了应用RMAN目录的一些优点。

表 11-1 使用RMAN目录的优点

只有目标控制文件	目标控制文件与目录
如果所有的目标数据库控制文件都丢失了，可以恢复但非常困难	如果所有的目标数据库控制文件都丢失了，则只需要几个简单命令就可以恢复。最新的目录包含了控制文件备份的位置
控制文件包含了备份信息的时间限制量	目录包含了所有RMAN备份操作的无限制历史记录
控制文件只保留了当前数据库结构模式信息。例如，控制文件只知道数据文件的名称以及所在的表空间	目录存储了所有模式信息并随着目标日志文件信息的更新而更新。因此，可以报告任何以前存在的结构模式
RMAN命令可在RMAN提示符下或通过脚本文件运行。	RMAN命令可以在RMAN命令行，通过脚本文件和存储脚本运行
没有目录的情况，目标控制文件备份必须发生在每次RMAN备份之后（该控制文件目前包含了新的备份信息）。同样，DBA必须保留详细的控制文件备份的路径和细节	目录包含了所有类似于包含在目标控制文件中的关于备份需要的信息。因此，目录提供了一个备份记录保存功能并且可以被用于从RMAN备份还原控制文件

目标数据库控制文件在某个极大日期范围内存储值得保留的备份与归档日志信息，而这一范围由init.ora文件中的CONTROL_FILE_RECORD_KEEP_TIME参数确定。默认情况下，该参数设定为7天。即在第8天时，最陈旧的信息（从第1天开始）可以被覆盖。

恢复目录比目标数据库控制文件保留了更多的RMAN历史动作记录。如果没有使用恢复目录而只以控制文件来记录备份，那么所有当前的日志文件都丢失时该怎么办？除非在其他地方还有备份的历史记录，否则就会丢失最近的RMAN备份信息。即使其他地方有记录的信息（没有使用目录），则还原和恢复还需要自己更多的手工操作以及Oracle支持的帮助。使用目录，完全丢失的数据库（包括所有控制文件）都可以被恢复。（假定已经保留了最新的目录。）

恢复管理目录包含了关于目标数据库的信息（表空间、数据文件以及日志文件等），这些信息都是从目标的控制文件中拷贝的。RMAN使用目录信息来执行备份、还原、恢复以及维护操作。例如，目录存储了所有目标数据文件和归档日志文件的文件名以及位置。RMAN提供报告机制，这样就可以查询目录浏览关于RMAN在目标数据库上执行备份的信息。目录可以有选择性地包含成组的RMAN命令，即存储脚本（第13章中将详细解释）。表11-1列出了只使用目标控制文件与恢复目录之间的区别。

当用RMAN部署一个恢复目录时，应注意以下事项：

- 将目录保存在与注册在其中的任何一个目标数据库都不同的数据库和机器上。也可以使用一个产品数据库作为其他产品数据库的目录数据库，反之亦然。例如，可以在数据库PRD1上生成一个目录以管理PRD2上的RMAN动作，而在PRD2上生成一个目录来管理PRD1的RMAN的动作。
- 为了避免在不同目标数据库版本上使用不同RMAN版本而引起的升级和兼容性问题，应该为每个准备使用RMAN的ORACLE数据库版本应用一个独立的目录所有者，为每个需要备份的不同版本的数据库创建不同的用户模式。对于一个8.0.6数据库，创建一个称为rman806的模式并使用8.0.6指导创建一个目录（使用catrman.sql脚本）。当备份8.0.6数据库时，应该使用8.0.6RMAN执行体。对于一个8.1.7数据库，创建一个称为rman817的模式并

使用创建目录命令创建一个8.1.7恢复目录。对这个备份，应该使用8.1.7的RMAN执行体。一个RMAN目录可以管理多个目标数据库。注意，Oracle确实支持一种版本的目录保存多个版本的目标数据库。例如，可以创建一个8.1.7目录并保存来自8.0.6、8.1.6和8.1.7数据库的备份。通过为每个版本建立一个目录用户，可以避免必须进行的烦琐的配置步骤以便RMAN可以在旧版本的数据库上工作（在Oracle 8i恢复管理器用户手册与指南中有详细的描述）。

- 应该确保备份目录数据库（有或没有RMAN）或至少目录数据库（使用导出功能的）中的任何目录用户。在恢复目录中包含的信息对于管理大量的数据库备份是至关重要的。

练习11.1：创建恢复管理器目录

本章练习如何创建一个恢复管理器目录。并将在新创建的目录中注册PRACTICE数据库。一旦完成了这些任务，就可以用RMAN命令和SQL*Plus查询语句来分析该目录。

任务描述	时间（分钟）
1. 准备目录数据库	10
2. 创建恢复管理器目录	5
3. 注册PRACTICE数据库	5
4. 为目录再同步控制文件	5
5. 备份目录用户	5
总计时间	30

本练习任务操作比较简单。当完成任务中的每一步时，都将讲述操作的原因。

任务1：准备目录数据库

因为目录是一个驻留在数据库中的模式，所以必须选择该目录驻留的数据库。尽管目录所有者（一个数据库用户）能被任何Oracle数据库创建，但可以只为了保留RMAN记录而创建一个数据库。回顾第3章，为RMAN目录创建了一个名为RCAT的数据库。在实际情况中，保存恢复目录的数据库应该保存在与目录中注册的任何一个数据库都不同的服务器上。

当创建一个目录用户时，需要表、索引、视图以及PL/SQL对象等的存储空间。尽管目录模式可以在任何表空间上创建，但最好在RCAT数据库中创建一个独立的表空间：

```
SQL> CREATE TABLESPACE cattbs DATAFILE '/oradata/RCAT/cattbs01.dbf'
2 SIZE 20M EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

将目录存放在独立的表空间中，可以迁移目录表空间，导出它（用Oracle 9i表空间导出）并从其他表空间分别备份。下一步是用SQL语句创建一个包含目录模式的用户：

```
LINUX> sqlplus sys/rcat@rcat
SQL> CREATE USER rman817 IDENTIFIED BY rman
2 TEMPORARY TABLESPACE temp
3 DEFAULT TABLESPACE cattbs
4 QUOTA UNLIMITED ON cattbs;
SQL> GRANT connect, recovery_catalog_owner TO rman817;
```

RECOVERY_CATALOG_OWNER是一个特殊的角色，应该只被看作恢复目录的所有者。该

角色间接地向被选择的用户赋予了许多系统特权。该用户必须在恢复管理器创建目录之前就存在了。在下一个任务中，将要使用RMAN在RMAN817模式中创建该目录。

任务2：创建恢复管理器目录

在创建该目录之前，首先介绍几个从操作系统的提示符下运行RMAN时的命令行选项。输入rman就可以启动RMAN。当执行RMAN时可以增加命令行参数。表11-2列举了一些有用的命令行参数。

表11-2 RMAN命令行参数

命令行参数	描 述
target	为目标数据库定义的一个连接字符串。当连接到一个目标数据库时，该连接是SYSDBA连接。该用户拥有启动和关闭数据库的权利，必须属于OS DBA组，必须建立一个口令文件允许SYSDBA连接
catalog或revcat	为一个恢复目录用户定义的一个连接字符串。当连接到一个目录数据库时，该连接不是一个SYSDBA连接。这与8.0版本开始引入的revcat一样
nocatalog	特殊用法，RMAN将不应用恢复目录。该参数与catalog参数互斥
cmdfile	定义了输出命令文件名称的字符串。当运行RMAN时，可以运行命令文件或者交互式运行
log或msglog	定义了包含RMAN输出信息的文件的字符串。log参数只能特别运用在命令行中。不能在RMAN中启动spooling，而在SQL*Plus的spool命令中可以执行。当应用日志文件时，输出的信息并不在屏幕上显示
trace	类似于log/msglog参数，将产生一个显示RMAN输出信息的文件。使用trace能在屏幕上显示信息
append	特殊用法，如果消息日志文件存在则将消息追加到该文件中。该参数经常与log参数联合使用

当在命令行中向RMAN传送参数时，参数值可以是有引号的字符串或者是没有引号的字符串。例如，当通过命令行定义目录数据库时，下面每个命令都是一样的：

```
LINUX> rman target sys/practice@practice # no equal sign
LINUX> rman target 'sys/practice@practice' # enclosed in '
LINUX> rman target "sys/practice@practice" # enclosed in quotes
LINUX> rman target / # OS authentication
LINUX> rman
RMAN> connect target sys/practice@practice
```

在本书中，将启动RMAN并在RMAN提示符下使用命令连接。这样连接信息就不会轻易地被操作系统看到。

这些同样的连接选择可用于关键字目标到关键字目录的连接，在RMAN提示符下创建一个目录：

```
LINUX> rman
RMAN> connect catalog rman817/rman@rcat
RMAN> CREATE CATALOG TABLESPACE cattbs;
```

在创建过程中，将创建表、索引、视图和PL/SQL包来存储目标数据库的元数据。以一个目录所有者身份通过SQL*Plus连接就可以查询这些对象的列表。

```
LINUX> sqlplus /nolog
```

```
SQL> CONNECT rman817/rman@rcat
SQL> SELECT object_name, object_type FROM user_objects;
```

任务3：注册PRACTICE数据库

在RMAN可以对一个有目录的数据库操作之前，必须在该目录中注册该目标数据库。在注册期间，RMAN从目标数据库的控制文件得到目标数据库结构信息并组装目录的表。诸如数据库ID、数据库名称、表空间、数据文件、重做日志文件以及归档日志文件等都可以被保存到目录中。

当注册PRACTICE数据库时，将要在同样的RMAN会话中连接到目录数据库和目标数据库。可以在RMAN提示符下应用连接命令建立这些连接。然后，在目录中注册连接的目标数据库：

```
LINUX> rman
RMAN> connect catalog rman817/rman@rcat
RMAN> connect target sys/practice@practice
RMAN> register database;
```

完成注册之后，rman817目录用户将包含备份PRACTICE数据库所需的所有PRACTICE数据库的信息。

在RMAN提示符下，执行两个命令可以提供有关当前目标数据库状态的信息。可以用LIST INCARNATION命令查看已经注册的数据库。如果有多个数据库在该目录中注册，则下面的命令将显示所有这些数据库：

```
RMAN> list incarnation;
RMAN-03022: compiling command: list
List of Database Incarnations
DB Key  Inc Key DB Name  DB ID                CUR Reset SCN  Reset Time
-----
1        2      PRACTICE 2629144163         YES 64301         10-JAN-02
```

使用report命令可以查找关于目前存储在目录中的PRACTICE数据库的注册信息。

```
RMAN> report schema;
RMAN-03022: compiling command: report
Report of database schema
File K-bytes  Tablespace          RB segs Name
-----
1          102400 SYSTEM              YES  /oradata/PRACTICE/system01.dbf
2           20480 RBS                YES  /oradata/PRACTICE/rbs01.dbf
3           10240 USERS              NO   /oradata/PRACTICE/users02.dbf
4           20480 TEMP                NO   /oradata/PRACTICE/temp01.dbf
5           20480 TOOLS              NO   /oradata/PRACTICE/tools01.dbf
6           20480 INDX                NO   /oradata/PRACTICE/indx01.dbf
7           10240 USERS              NO   /oradata/PRACTICE/users01.dbf
```

如果得到了类似上面的PRACTICE数据库，则表明已经成功地将PRACTICE数据库注册到RCAT数据库的目录中了。

任务4：为目录再同步控制文件

当将RMAN应用到恢复目录时，必须保持目录信息与目标数据库控制文件的同步。RMAN使用两种类型的同步方式：全部和部分。全部再同步将以控制文件中所有已经改变的非重用的记录来更

新目录，非重用记录包括数据文件、表空间以及联机重做日志文件。部分再同步将用控制文件中的重用记录来更新目录，重用记录包括日志历史、归档日志信息以及任何RMAN备份记录。可以用RESYNC CATALOG命令重新同步控制文件。为了进行全部再同步，RMAN将提取当前目标控制文件的一个快照拷贝。该快照控制文件确保控制文件的信息在目录更新时没有改变。在Linux上，默认的快照控制文件名是snapcf_\$ORACLE_SID.f (snapcf_PRACTICE.f 是PRACTICE数据库的)，可以在\$ORACLE_HOME/dbs目录下找到该文件。在Windows NT中，快照控制文件被命名为SNCF%ORACLE_HOME%.ORA (SNCFPRACTICE.ORA 是 PRACTICE的)，位于目录%ORACLE_HOME%\database下。可以使用以下命令在下次再同步时将该文件转移到其他目录下：

```
LINUX> rman
RMAN> connect catalog rman817/rman@rcat
RMAN> connect target sys/practice@practice
RMAN> set snapshot controlfile name to
2>      '/oradata/PRACTICE/snap_PRACTICE.ctl';
RMAN> resync catalog;
```

该resync命令是一个再同步的示例。一些RMAN操作会自动为目录同步数据库控制文件。RMAN将检测是需要全部再同步还是部分再同步。如果只是需要部分再同步，就不应用快照控制文件。也可以应用RESYNC CATALOG命令。在这个命令中，目录表将使用全部再同步的机制更新最新的控制文件信息。能触发目录部分或全部再同步的命令包括Backup、Copy、Crosscheck、List、Report、Delete Expired Backup Set、Duplicate、Restore、Recover和Switch。

为了解释控制文件与目录的同步，再打开一个命令提示并执行以下这些命令，重新命名一个数据文件（当完成这个测试后，可以再将该数据文件的名称复原）：

```
LINUX> sqlplus sys/practice@practice
SQL> ALTER DATABASE DATAFILE 6 OFFLINE;
SQL> host mv /oradata/PRACTICE/indx01.dbf
2      /oradata/PRACTICE/indx1.dbf
SQL> ALTER DATABASE RENAME FILE
2      '/oradata/PRACTICE/indx01.dbf' TO
3      '/oradata/PRACTICE/indx1.dbf';
SQL> ALTER DATABASE DATAFILE 6 ONLINE;
```

PRACTICE数据库的控制文件知道INDX表空间数据文件的真实位置，但目录并不知道准确的位置。当在RMAN提示符下运行report模式时，将会执行全部再同步。在report命令运行之前，RMAN自动执行目录的再同步：

```
RMAN> report schema;
RMAN-03022: compiling command: report
RMAN-03024: performing implicit full resync of recovery catalog
RMAN-03023: executing command: full resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete
Report of database schema
File K-bytes Tablespace RB segs Name
-----
1          102400 SYSTEM          YES      /oradata/PRACTICE/system01.dbf
```

2	20480 RBS	YES	/oradata/PRACTICE/rbs01.dbf
3	10240 USERS	NO	/oradata/PRACTICE/users02.dbf
4	20480 TEMP	NO	/oradata/PRACTICE/temp01.dbf
5	20480 TOOLS	NO	/oradata/PRACTICE/tools01.dbf
6	20480 INDX	NO	/oradata/PRACTICE/indx1.dbf
7	10240 USERS	NO	/oradata/PRACTICE/users01.dbf

注意REPORT SCHEMA命令输出的信息。在REPORT SCHEMA命令执行过程中，RMAN自动用目标控制文件检查当前的目录。因此，一个全部再同步就会被执行。同时要注意的是，最新的信息包含在目标控制文件中。同步将从目标控制文件向目录传送信息。

技巧 每天针对目录同步安排几次再同步，以确保目录随着目标控制文件而更新。如果目标数据库每天生成了许多归档日志文件，则为了包括这些日志文件需要更新目录，这将确保该目录能成功地在失效的情况下执行数据库自动还原和恢复。

任务5：备份目录用户

要保护Oracle数据库，还应该保护目录数据库的内容。可以通过用户管理或者服务器管理方法备份整个目录数据库。还可以经常导出目录用户。当导出目录用户时，可以使用如下的导出参数文件：

```
USERID = rman817/rcat@rcat
FILE    = /oradata/PRACTICE/backup/ch12/export_user_rman817
LOG     = /oradata/PRACTICE/backup/ch12/export_user_rman817
OWNER   = rman817
```

当应用该参数文件执行导出时，可以创建一个二进制导出文件以便以后用于还原恢复目录。

注意 在本书所有的RMAN练习中我将使用同一个目录。如果只计划使用目标控制文件，则应在该练习的RMAN目录中使用nocatalog选项并且不要连接该目录。然而，本章其他一些练习可能需要目录。

创建一个冗余的目录

可以将数据库注册到另一个恢复目录中，以便更好地保护目标数据库RMAN的部署。一旦目标数据库失效且第一个恢复目录也失效时，第二个目录就可以用于快速、方便的恢复。可以试用这个窍门并管理一个冗余的目录：与任务1类似，在独立于目标数据库和第一个目录数据库的其他数据库上创建一个享有权利的用户。使用任务5中导出的内容，将该用户导入到其他的数据库中。每次原始的目录与目标数据库同步后，直接使用resync命令，连接到另一个目录并应用同样的resync命令。每次备份之后，第二个目录也需要再同步，以确保包含了最新创建的备份。像这样管理两个恢复目录模式比较容易，而且一旦目标数据库和原始恢复目录失效时可以提供额外的保护。

11.4 介质管理层

Oracle恢复管理器自动配置为向磁盘备份数据库文件。RMAN也能对一个或多个磁带设备执

行备份与恢复操作。为了完成这些功能，需要配置Oracle服务器软件与磁带供应商的软件进行通信。位于Oracle RMAN组件和磁带供应商的软件之间的通信层叫做介质管理层（MML）。Oracle在Oracle的web站点(<http://www.oracle.com/database/recovery>)上列出了一系列与恢复管理器协同工作的磁带管理软件供应商。

Oracle安装了Oracle备份解决方案程序，提供对磁带管理软件产品范围的认证，这些产品符合Oracle的介质管理层（MML）API标准。Oracle发布该标准，而且软件供应商编写符合该MML接口的软件。因此，Oracle服务器会话可以不知道（或不关心）供应商提供的磁带访问方法就能向介质管理器发布命令来备份或还原文件。

介质管理器是为了备份以及恢复数据而加载、标记、卸载顺序介质（比如磁带驱动等）的一种应用。当为Oracle备份配置MML时，可以从磁带备份供应商那里获得一个库文件并将该库文件连接到Oracle执行体上（某些版本的Oracle需要重新连接）。该共享的库类似于在计算机上一个安装了硬件设备的驱动。当RMAN与SBT（系统备份到磁带）接口交互时，该共享库与磁带管理软件通信。一旦MML与Oracle服务器软件配置结束，RMAN就可以通过磁带管理软件来读写磁带了。因此，介质管理层将RMAN命令翻译为介质管理器可以理解的指令。例如，RMAN命令将文件拷贝到SBT接口，MML将该命令翻译为磁带管理软件可以理解的本地指令。一个简单的拷贝命令意味着磁带必须加载并且在拷贝开始之前定位。这样，RMAN发送简单指令而磁带软件处理所有磁带设备的细节。

11.4.1 配置介质管理层

当Oracle销售时，它提供了一个在UNIX上被称为libobk.so的虚拟共享库。可以这样在UNIX上配置MML：为了安全重新命名该文件，比如libobk.so.backup等类似的名称。然后创建一个名为libobk.so的符号链接到磁带软件供应商的RMAN兼容库上。这样，当Oracle从RMAN调用该libobk.so时，实际上是向磁带管理软件库发送命令。

作为一个通用的示例，假设UNIX服务器已经连接到了一个磁带设备，该设备由一个软件公司命名为XYZ的软件管理。如果需要配置RMAN而向XYZ软件介质管理器写入，则首先查找XYZ软件是否有一个与备份解决方案程序中的Oracle标准兼容的库。可以在Oracle的web站点上查看是否收录了XYZ软件或直接联系XYZ。一旦XYZ软件被安装并且可以成功地对磁带进行读写，则查找Oracle服务器软件必须链接的库。磁带软件公司可以提供支持的库文件名以及特殊配置的指导。当找到该库的名称后，通过以下步骤与Oracle软件链接（本示例中的库文件名为libxyz.so）：

- 1) 关闭所有共享当前ORACLE_HOME目录的Oracle数据库；
- 2) 重新命名以前连接到libobk.so的符号链接：

```
mv $ORACLE_HOME/lib/libobk.so $ORACLE_HOME/lib/libobk.so.backup
```

- 3) 在希望使用的共享库与libobk.so之间建立一个符号链接：

```
% ln -s $ORACLE_HOME/lib/libobk.so $TAPE_SOFTWARE/lib/libxyz.so
```

- 4) 应用sbttest测试工具备份文件以验证符号链接是否成功。例如，输入

```
% sbttest testfile.txt -trace sbtio.log
```


在UNIX平台（不是Windows）上，可以使用SBTTEST的工具来辅助检查介质管理。应用该工具，不运行RMAN就可以测试介质管理软件。该工具向磁带写入一个示例文件（比如testfile.txt），然后删除。参考《the Oracle8i Recovery Manager User's Guide and Reference》的第9章，其中详细介绍了有关介质管理层的问题。

如果使用Oracle 8.0.x，则需要在上述建立符号链接之后手工将RDBMS与MML软件链接。在重新连接Oracle软件时应确认将所有8.0.x的Oracle Home下的Oracle软件都关闭了。通常情况下，每个MML连接到Oracle的方法都依赖于选择的MML。参考适合于Oracle应用的MML文档以获得更多的指导。

下面是一个简短的示例，演示如何在Windows NT/2000上将Legato存储管理器与Oracle 8i集成在一起：

- 1) 确保已经从Oracle 服务器的CD-ROM中安装了Legato存储管理器。在Oracle安装CD-ROM中，在\disk1\lsm目录下有一个名为lsm57_nt.zip的文件。解压缩该文件并执行setup.exe，安装Legato存储管理器。该磁带驱动一定在Settings | Control Panel | Tape Devices菜单下可以找到。
- 2) 通过控制面板在path环境变量中设置一个bin目录，即已经安装了Legato存储管理器的目录。例如，如果在c:\Program Files\nsr目录下安装了Legato，则将c:\Program Files\nsr\bin增加到path变量中。

有许多安装MML的细节都是关于操作系统和磁带软件供应商的标准的，所以我在本节中并没有包括更多配置MML的细节，尽管我希望本节可以为读者在自己的环境下成功地配置MML指出一个正确的方向。当然也可以向Metalink以及磁带软件供应商寻求帮助。

11.4.2 咨询磁带管理供应商

当共享库连接并测试后，就可以对磁带执行RMAN操作了。在RMAN提示符下对磁带进行读写时，磁带管理供应商可能针对他们自己的软件有一些特殊的参数设置。向磁带供应商咨询关于磁带供应商要求的特殊事项。

不仅能在RMAN提示符下对磁带执行RMAN的操作，许多磁带管理供应商还设计了软件可以调用RMAN脚本来执行备份。磁带软件工作执行的操作与RMAN提示符下执行的操作类似。因此，RMAN进程可以运行或由磁带管理供应商的软件执行。可以购买这里描述的第三方软件来备份OS文件系统、管理并自动化RMAN备份的过程。Legato Networker、Veritas Netbackup、HP OmniBack、Solstice Backup和ARCserve等产品都有与Oracle恢复管理器的定义良好的接口。

注意 本书中所有的RMAN操作都将被写到磁盘上。通过书中的示例，可以在安装了MML的环境下用磁盘代替磁带（SBT_TAPE）。

11.5 Oracle 9i 的新特性

Oracle 9i对RMAN进行了大幅度的改进，使得RMAN比以前更实用了。下面是一些得到增强的特性：

- **自动通道配置** 在用RMAN 8.x版本备份、拷贝或者还原数据库文件以前，需要为目标数据库的输入和输出通信分配一个通道。现在用CONFIGURE命令就能够以不同的参数来创建一套固定的可重用的通道。这些通道在需要发送类似备份、拷贝、还原或恢复等命令时由Oracle自动分配。
- **保留策略** 类似于在介质管理软件使用的那些保留策略，RMAN可以被配置为期限到达管理备份。当数据到达一定的时期或被备份超过一定的次数时，可以将备份标记为过时的或将其删除。
- **数据块介质恢复** 在Oracle 9i以前，如果在一个数据文件中发现有一个或多个数据块损坏，则这个文件需要脱机，从一个备份还原并应用归档日志向前回退。使用块恢复，可以使受损坏的文件继续联机，允许访问没有受损坏的数据块，并只对那些需要恢复的数据块进行归档重做。
- **可持续备份** 如果数据库的备份部分有遗漏，则可以重新备份，而且只备份第一次没有备份的文件。
- **备份优化** RMAN可以检测到同一文件是否已经被备份了一定的次数。如果已经到达了该次数而且该文件相同，则将不再进行备份。这对于归档日志文件比较合适，因为这减少了归档日志被备份到磁带上的次数。
- **还原优化** 这个新的特性强迫RMAN去检查数据文件是否已经被还原到磁盘上了。如果当前的数据文件被找到而且比以前的备份更适合恢复，则RMAN将不还原该数据文件，而当前的数据文件将被用于恢复。

11.6 疑难解答

我希望本章这个简单的安装练习没有给读者带来多少问题，但是如果遇到一些错误，以下这些提示可能有助于解决问题。

1) RMAN-20002: target database already registered in recovery catalog

如果在目录中已经注册了数据库，那么在注册数据库命令后会出现这个错误提示。如果需要从目录中注销一个数据库，以目录所有者的身份连接该目录数据库。在注销目录之前，应将该数据库的所有备份集都删除。对需要注销的数据库用db_key和db_id执行存储过程。查询属于目录用户的db表，得到db_key和db_id。例如，如果找到的PRACTICE数据库的db_key是2而db_id是13231834，则可以这样注销该数据库：EXECUTE dbms_rcvcat.unregisterdatabase(2, 13231834)。

2) RMAN-06008: connected to recovery catalog database

RMAN-06428: recovery catalog is not installed

如果以一个没有拥有目录模式的用户连接目录数据库，或者该目录还没有创建时，将得到这个错误信息。检查为目录数据库提供的用户和口令是否正确。如果提供了正确的用户和口令，则使用创建目录的命令来创建该目录。

3) RMAN-06004: ORACLE error from recovery catalog database:

ORA-01658: unable to create INITIAL extent for segment in tablespace

CATTBS

这个错误信息说明在创建目录模式对象过程中，目录创建进程超出了范围。当创建目录时，表和索引必须被创建在目录表空间或者在默认的目录所有者表空间内。如果超出了这个空间范围，将不能成功完成目录的创建。删除、重新创建目录用户，为目录对象的表空间分配更多的空间，然后重建该目录。如果目录所有者除了目录对象外还有自己的对象，则必须在创建目录之前将这些目录对象逐一删除。因为诸多原因，目录所有者应该只拥有目录对象。

除了这些疑难解答的技巧外，我认为对于RMAN信息的简要解释和MML的疑难解答对读者也有帮助。

11.6.1 RMAN的信息

当运行RMAN命令时，会生成大量的输出结果。大多数情况下，输出的都是相关信息，但有时则包含了错误。通常情况，RMAN的错误信息包含的文字类似如下：

```
RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
```

一旦出现这样的信息，则随后的将是引发错误的问题信息。应该阅读后续的信息。有时会遇到一个ORA-错误信息。常见的Oracle和RMAN错误信息都可以在Oracle8i 错误信息手册上找到。经常有一些RMAN信息包含了引起错误原因的所有信息。当阅读这些错误信息时，应该从下向上看。从最下面的消息开始然后逐步向上浏览，因为最下面的消息通常是错误的原因。

11.6.2 介质管理层

当用磁带供应商的介质管理层（MML）配置RMAN时，可能会遇到许多问题。可以检查上述的输出信息，并执行下面的一些任务以检测RMAN/MML的问题：

- 操作平台可能带有一个MML检测工具，sbttest。可以使用该工具模拟访问磁带并检查MML。在命令行中输入sbttest，观察命令行选项。
- 介质管理层可能会将重要的信息输入到名为sbtio.log的日志文件中，该文件在\$ORACLE_HOME/rdbms/log/目录或者在用户dump目录下。
- 告警日志文件、目标和目录数据库可能包含解决MML问题的有用信息。
- 检查磁带管理供应商的跟踪日志工具和文件。可以将RMAN的输出和这些文件进行相互联系，就可以发现是哪些RMAN动作引起了磁带管理的问题。
- 查看RMAN和磁带供应商的论坛。用磁带供应商和MML进行搜索，查询是否有人解决了相同的问题。

尽管我不能在本书中涵盖MML更多复杂的方面，但我认为这个简要的疑难解答指导能给出在Oracle数据库环境部署RMAN的正确方向。

11.7 小结

在本章中，首先介绍了RMAN的体系结构、服务器管理恢复，描述了基本组件并详细讨论

了使用恢复目录的优点。用创建目录的命令创建了一个RMAN目录，并讲解了可以创建包含不同版本数据库信息的多个目录。讲述了注册数据库命令以及如何用目标数据库控制文件同步恢复目录。还讨论了介质管理层（MML），并大致讲解了如何配置Oracle应用所选择的介质管理软件。关于RMAN概念和配置的更多信息可以参考《Oracle8i Recovery Manager User's Guide and Reference》的第1、2、3章。为了更好地在RMAN上安装MML，请联系磁带管理软件供应商。

在下一章，我们将讲解如何执行关闭和打开的数据库备份，还将讲述RMAN如何备份控制文件和归档日志文件。

习题

请回答以下问题，以巩固本章的概念和练习。

1. 在RMAN提示符下使用什么命令可以创建恢复目录？

- A. register database
- B. create catalog
- C. resync database
- D. list database

2. 使用RMAN目录而不仅仅使用目标控制文件有什么优点？（多项选择）

- A. 如果目标控制文件丢失，使用RMAN备份时恢复会更简单。
- B. 目录信息可以用于更新目标控制文件。
- C. 目录保存了目标控制文件中没有的历史备份信息。
- D. 目录价格低，可以自由购买。

3. RMAN被称为服务器管理恢复是因为它管理备份、还原和恢复的所有方面。

- A. True
- B. False

4. 可以在RMAN提示符或用RMAN命令行参数连接目录。为了在RMAN提示符下连接在RCAT数据库中的目录，用户名为rman817，应该输入：

- A. connect target rman817/rman@rcat;
- B. connect target rman817/rman@rcat as SYSDBA
- C. connect rman817/rman@rcat as catalog
- D. connect catalog rman817/rman@rcat

5. 介质管理层允许RMAN向磁盘和磁带写入备份。

- A. True
- B. False

答案

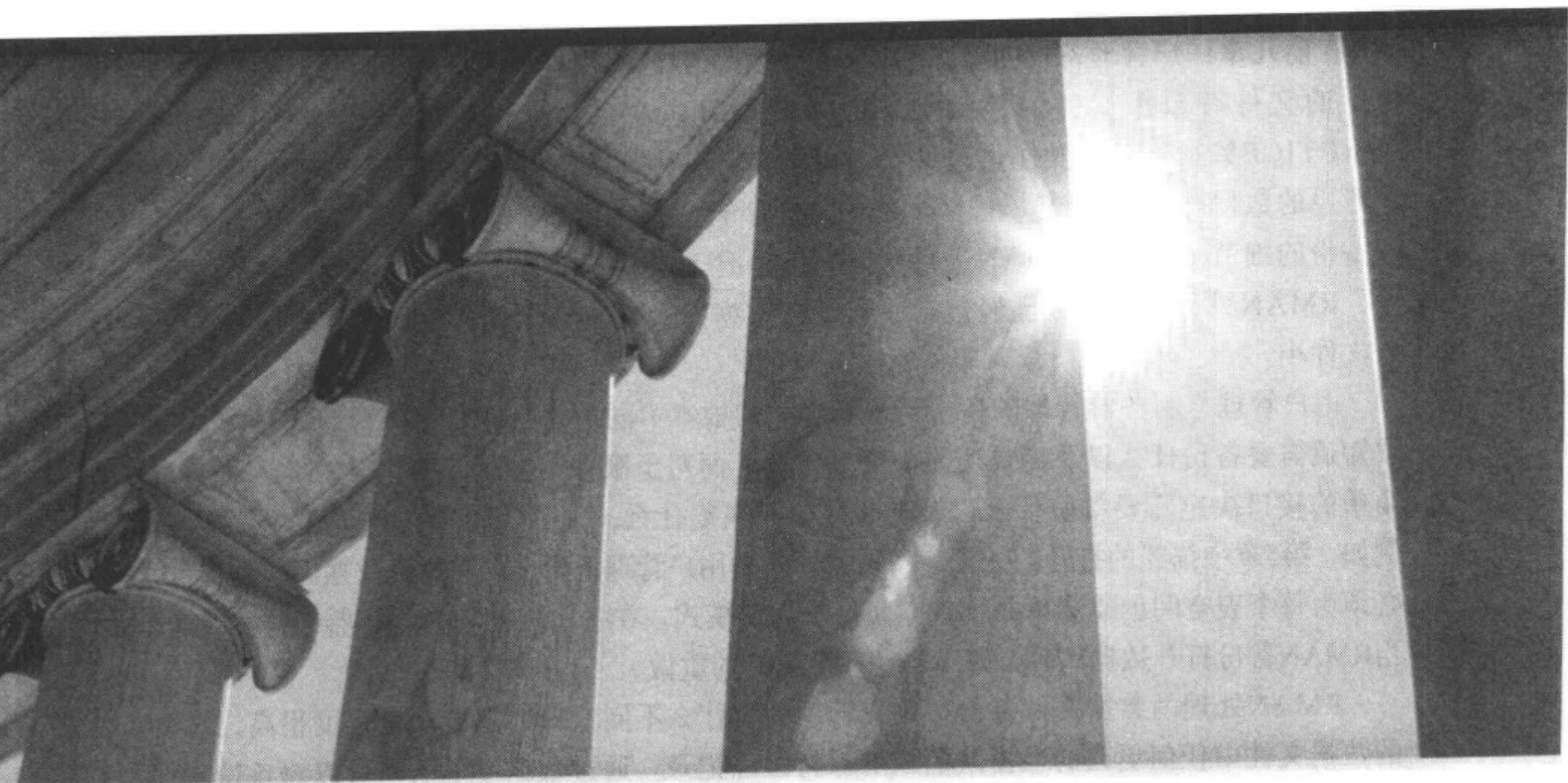
1. B。当创建目录时，数据库模式对象在目录用户模式下创建。

2. A, C。使用目录最大的优点是，如果目标控制文件丢失，使用RMAN备份会很容易。恢复目录将包含目标控制文件所没有的历史信息。

3. True。RMAN以及其所有相关的组件都与备份文件、时间、位置和介质保持一致。我们只要进行安装、连接、发布备份命令，然后等待，让RMAN去管理。

4. D。connect命令允许连接到目录或目标数据库。在关键字connect之后，应指定需要连接到哪里以及用于连接的用户标识。

5. False。MML对于向磁带读写备份是必须的，但对于向磁盘备份则不是必须的。



第12章 RMAN备份

在前几章用户管理备份的操作中，我们介绍了如何使用操作系统命令拷贝整个文件。涉及拷贝的文件有数据文件、控制文件以及归档日志文件。配合使用这些文件还原并恢复PRACTICE数据库。在操作过程中，用户必须自己管理备份与恢复操作，必须创建需要拷贝和还原的数据库文件清单。在本章中，我们将介绍RMAN如何使用服务器相关组件替用户处理备份的细节。当使用RMAN进行数据库备份时，RMAN知道需要对哪些文件进行操作。另外，RMAN只备份当前或已经包含了数据的数据库数据块。因此，备份的文件要比实际的数据文件小。

用户管理与服务器管理的备份与恢复之间的根本不同之处是：对用户管理的操作，DBA必须知道需要备份什么以及需要还原与恢复什么；而对于服务器管理的操作，RMAN将通过一个简单的接口决定需要备份什么以及需要还原与恢复什么，大部分管理任务被省略或简单化了。例如，第5章中描述的打开的数据库备份，如果是用户管理备份，则在拷贝数据库数据文件之前，必须为每个表空间创建清单并将表空间置于备份模式。在备份模式下，服务器产生额外的重做。当RMAN备份打开数据库时，并不自动产生额外的重做。

RMAN处理的数据文件备份与用户管理的有什么不同？RMAN智能化程度很高，对要备份的数据文件中任何更改的数据块的SCN保持跟踪记录。通过数据库的活动可以对备份文件的更改继续进行。在恢复阶段，RMAN将自动应用需要的重做并使文件处于一致的状态。备份期间，有可能出现被RMAN拷贝的数据块没有被DBW（数据库写入器）进程完全写入到磁盘的现象，从而导致数据块头部与尾部不一致。这种现象被称为“断裂”或“破碎”块。当检测到这样的数据块时，RMAN将重新读取该块，直到完成了完整的拷贝。因此，RMAN可以以最小的性能降低为代价对打开的数据库进行备份。

12.1 恢复管理器的备份选项

恢复管理器为数据库的备份提供了三种不同的方法：备份、映像拷贝以及代理拷贝。尽管这三种方法都能提供在恢复或数据库失效时所需要的同样的数据库备份结果，但每种方法都各有优缺点。

12.1.1 备份

RMAN备份在选择的备份介质（磁盘或磁带）上以自己的格式创建一个或多个物理文件，这些物理文件被称为“备份片”。备份可以由数据库、被选择的表空间、数据文件、控制文件、归档重做日志或多种混合体组成。

- 数据文件和控制文件可以合并在同一备份中。
- 归档日志文件必须与数据文件和控制文件分开备份。

RMAN创建的备份文件（备份片）不能被其他任何工具读取，因此只能用RMAN还原。为了解这些备份是如何被创建的，首先要理解用到的基本术语：

- **通道**：在第11章中已经介绍了，通道是从目标数据库到备份介质的通信途径。这个通道类型可以被定义为磁盘或‘SBT_TAPE’（SBT代表系统备份到磁带）。通道可以设置许多参数来规定备份片的大小、同时打开的文件数以及文件读取的速度（每秒的字节数）。在备份开始之前至少要分配一个通道，当然也可以使用多个通道。
- **备份集**：备份集就是当运行备份命令时生成的物理文件（备份片）的逻辑组合。至少要为一个备份创建一个备份集，但通常都使用多个备份集。可以使用filesperset参数来控制一个备份集中备份的文件数量或用setsize参数设置每个备份集的最大尺寸。如果分配了多个通道，RMAN将为每个通道创建一个备份集。RMAN也试图对备份集中的备份文件进行分割，这样每个备份集大小都一样而且每个通道的数量也都相同。一个数据文件/控制文件/归档日志文件只属于一个备份集，但一个备份集可以包含许多文件。
- **备份片**：备份片是RMAN在备份期间创建的物理文件。这个文件将包含来自目标数据文件、控制文件或归档重做日志得到的实际数据块。每个备份集至少包含一个备份片，但也可以包含多个。一个数据文件可以跨越多个备份片。
- **格式（format）**：格式是在备份期间赋予生成的备份片的文件名。当备份到磁盘时，还要提供备份片的全路径。对于磁带备份，只需要文件名。
- **标记（tag）**：标记是用于标记一个备份的一个有意义的名称。例如，Sunday level 0 备份或许意味着Sun-Lev0。RMAN可以直接使用这个标记从一个指定的备份中重建文件。标记的最大长度为30个字符，而且一个标记可以用于多个备份。
- **备份键值**：RMAN创建的每个备份都被赋予了用于确定身份的一个惟一序列号。

当执行backup命令时，请考虑以下问题：

- 1) 备份时数据库处于什么状态？（数据库状态）
- 2) RMAN备份需要保存到什么地方？（设置目的地）
- 3) 需要备份的是数据库的哪些部分？（数据库文件）
- 4) 需要拷贝数据库文件的哪些数据块？（增量级别）

对这四个问题的回答将决定如何使用RMAN的BACKUP命令来进行备份。下面讨论备份时需要考虑的这四部分。

1. 确定备份集数据库状态

首先，RMAN可以在两种不同的数据库状态之间进行备份。类似于用户管理的备份，可以在数据库关闭或打开时进行备份。

- **关闭的数据库备份** 类似于用户管理的关闭（冷，稳定的）数据库的备份，目标数据库没有打开。但不同于用户管理的备份，RMAN期望数据库处于加载状态。RMAN在关闭数据库备份时必须访问目标数据库控制文件以获得结构信息。
- **打开的数据库备份** 类似于用户管理的打开（热，不稳定的）数据库的备份，目标数据库处于打开状态。但RMAN的不同之处是：在备份期间没有将表空间置于备份模式，不会生成额外的重做。

2. 确定备份集目的地

接着，涉及到备份的目标集方面的术语。当执行RMAN备份命令时，RMAN将生成一个或

多个文件（备份片）。这些文件除了备份的数据块之外还存储了RMAN的控制信息。为RMAN创建的备份片赋予有意义的名称可以避免无人知道磁盘或磁带上的文件的来历或内容。通过增加format参数可以实现这个目的。创建文件名时可以使用许多替换变量，这样对于磁盘的备份是唯一的，而且对于磁带备份也是理想的惟一性标记。下面是一些可用的格式变量：

- %d 数据库名称。
- %n 为数据库名增加1个或多个x字母使其长度达到8个字符。
- %s 备份集编号。这些编号来源于目标控制文件，从1开始每次递增值为1。如果控制文件重新生成，则该编号回到1重新开始。
- %p 备份集内备份片的编码。第一个备份片的号码是1，而且在同一备份集中每个新的备份片递增值为1。
- %t 备份的时间值。该值是从以前某个参考时间点开始到现在的秒数。
- %u 一个8位字符的值，来自于备份集数字和备份的时间。
- %c 备份集中的备份片的拷贝数。除非指定为双份备份外，否则该值一直是1。
- %U 生成一个惟一的由%u_%p_%c组成的文件名。如果没有指定备份文件的格式，默认值是%U。

表12-1中列出了一些使用这些格式变量的示例。应该注意，只有为了备份到SBT_TAPE通道的文件才需要指定文件名。如果备份到磁盘则必须指定路径名和文件名。如果没有指定路径名，备份在通常情况下将被存储到\$ORACLE_HOME/dbs。

表12-1 名为PROD的数据库的备份格式结果

指定的格式	示例结果
dbinc0_%d_%s_%p_%t	dbinc0_PROD_21_1_447001901
tbsUSERS_%U	tbsUSERS_08dae506_1_1
al_%n_%t_%s_%p	al_PRODxxxx_447157468_41_3

警告 在选择文件名之前，应检查需要备份的介质的最大文件名长度。如果选择的文件名超过了这个限制，则最终可能会以同样的文件名覆盖另外一个文件。

3. 确定备份集数据库文件

下一步，定义属于RMAN备份范围的术语。在执行一个单独的backup命令时，需要指定包含在备份集中的数据库文件。RMAN BACKUP命令将只拷贝来自数据文件、控制文件和归档日志的数据块。可以应用下面这些备份选项：

- **整体数据库备份** 备份集中包含了目标数据库中所有的数据文件和控制文件的数据块。在备份命令之后用关键字DATABASE指定进行整体数据库备份。例如BACKUP (DATABASE)。
- **表空间备份** 指定表空间的所有数据文件的数据块都被拷贝到备份集中。在备份命令之后用关键字TABLESPACE指定进行表空间备份。例如：BACKUP (TABLESPACE SYSTEM, USERS)。
- **数据文件备份** 每个被命名的数据文件的数据块都被拷贝到备份集中。在备份命令后用关

键字DATAFILE指定进行数据文件备份。例如：BACKUP (DATAFILE 1,2,3,4)。

- **控制文件备份** 数据库控制文件中的所有数据块都被拷贝到备份集中。在备份命令之后用关键字CONTROLFILE指定进行控制文件备份。例如：BACKUP CURRENT CONTROLFILE。当对系统表空间进行备份时，控制文件自动包含在备份当中。因此一个完全的数据库备份也包含了当前的控制文件。
- **归档日志备份** 每个归档日志文件的所有数据块都被拷贝到备份集中。在备份命令之后用关键字ARCHIVELOG指定进行归档日志备份。尽管可以用同样的备份命令备份数据文件、控制文件和归档日志，但RMAN将为归档日志创建一个独立的备份集。例如：BACKUP (ARCHIVELOG ALL)。

4. 确定备份集的级别

最后，我们探讨RMAN备份级别的选项。当RMAN在备份期间拷贝数据块时，可以指定在备份时需要拷贝哪些数据块。我们知道，数据文件都是由数据块组成的，每个块都在被更新时赋予了一个系统更改号（SCN）。当进行备份时，每个块都被从数据文件中读出，但只有符合备份级别条件的数据块才被拷贝到备份集中。因为归档日志和控制文件中所有的数据块都将被拷贝，所以备份级别只属于数据文件备份。RMAN不从数据文件中拷贝空的数据块到备份集中，只有进行映像拷贝时才拷贝空的数据块。

12.1.2 RMAN 备份压缩

当在一个数据库中增加数据文件时，该文件中所有的数据块都被格式化为没有使用过的空数据块。Oracle通过在块的头部和尾部添加一些同样的结构值来完成格式化工作。这种块在包含数据时被分类。如果向表中插入记录然后又删除所有的行，甚至重新截断该表，则这些包含了删除数据的块仍然被认为是被应用过的块。因此RMAN也将备份这些没有包含当前数据的块。RMAN不备份那些从来没有包含过数据的空数据块。因此这样的备份要比原始的文件小，所以RMAN称之为“备份压缩”。注意在备份期间整个数据块都被拷贝，而备份片文件中的数据块没有被压缩。

注意 RMAN将拷贝文件头部，也就是每个数据文件的第一个数据块。

当RMAN进行备份数据文件时，有两种模式的操作：增量式和完全式。

1. 增量式备份（Incremental Backup）

采用增量式备份策略可以只备份上次增量备份之后更改后的数据块。这样可以大大减少备份的数据量，并因此可以减少进行备份和还原所需的时间和空间。

增量备份含有备份级别的概念。增量备份的级别是一个0到4之间的整数。当进行增量备份时，数据检查点SCN存储在目标控制文件中。随后的增量备份决定了需要拷贝那些与以前的增量备份级别相关的而且发生在SCN时间的数据块。备份级0是全集备份级别，在级别为0的备份中，要备份选中数据文件的所有使用过的数据块。

增量备份有两种类型：差异型和累积型。差异型备份拷贝最近进行的同级或低级增量备份以来所有改变的数据块。例如，在级别为1的差异型增量备份时，所有最近级别为1的备份以来

的数据块都要拷贝。如果以前没有进行过级别为1的备份，则所有最近级别为0的备份以来的数据块都要拷贝。当备份命令中使用了incremental关键字时，则差异备份是RMAN默认的增量备份类型。

累积型备份拷贝最近低级别增量备份以来的所有更改过的数据块。例如，在级别为2的累积增量备份时，自从上次级别为1的增量备份以来的所有数据块都要拷贝。如果没有进行过级别为1的备份，则自上次级别为0的增量备份以来的所有数据块都要拷贝。

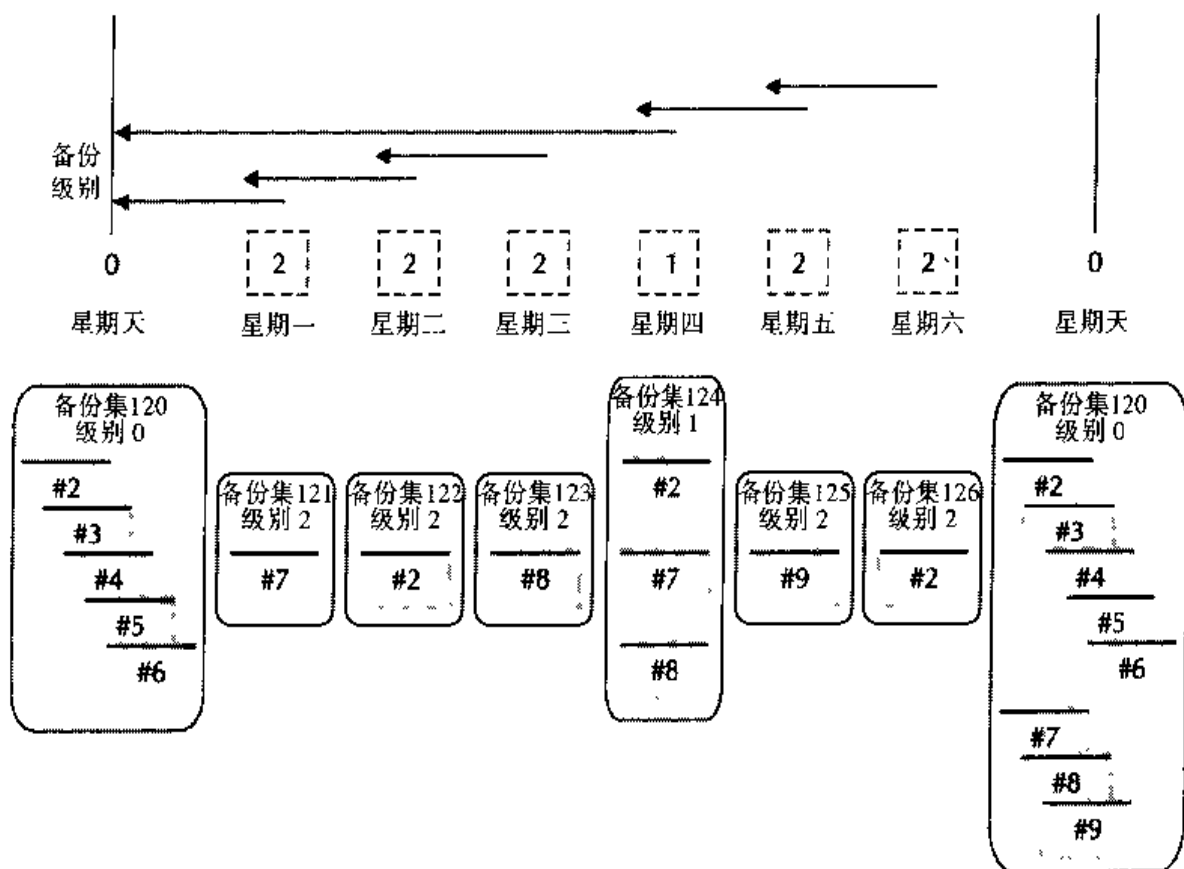


图12-1 差异增量备份基线示例

图12-1给出了不同增量备份的一个基线，有助于理解增量备份的不同之处。一个星期的每一天为数据文件生成一个备份集。实施这些备份的描述如下（这个简单的例子没有考虑当用户对数据进行更改时会引起的数据字典对象的更改）：

- **星期天** 进行级别为0的增量备份。数据块#2、#3、#4、#5和#6被拷贝到备份集120中。这些块组成了选为备份的数据文件中的所有使用过的数据块（尽管在一个实际的数据库中要查找只有5个使用过的数据块的数据文件需要很长时间）。级别为0的增量备份既不是累积型也不是差异型。这是作为增量备份策略的一个基准。
- **星期一** 进行级别为2的差异增量备份。自从星期天备份以来，数据文件中更改的块是#7，这个块中充满了数据。因此，这个数据块是惟一被拷贝到备份集121中的（增加了文件头以及来自数据字典的一些数据块或空间管理块）。
- **星期二** 进行级别为2的差异增量备份。自从星期一的备份以来，数据块#2中的一行数据

被更新了。因为这个数据块是惟一被更新的，所以只有这个数据块被拷贝到备份集122中。

- **星期三** 进行级别为2的差异增量备份。自从星期二的备份以来，数据块#8中插入了一行数据。因为这个数据块是惟一被更新的，所以只有这个数据块被拷贝到备份集123中。
- **星期四** 进行级别为1的差异增量备份。自从最近一次的级别为1或更低级的备份以来，向备份集123中拷贝所有更改过的数据块。自从级别0的备份以来，没有进行级别为1的备份。因此，自星期天的0级别备份以来所有更改的数据块都被包含在这个备份集中（#2、#7和#8）。
- **星期五** 因为自从星期四的级别1的备份以来，数据增加到了数据块#9，所以级别为2的增量备份将#9拷贝到备份集125中。
- **星期六** 因为自从星期五的级别2的备份以来，数据块#2中数据被更改了，所以级别为2的增量备份将#2拷贝到备份集126中。
- **星期天** 进行级别为0的增量备份将所有使用过的数据块拷贝到备份集127中。这时又开始新一轮的循环备份。

从图12-2中可以看出，累积增量备份类似于差异备份，但略有不同。这个图显示了累积增量备份的基线。一个星期的每一天为一个数据文件生成一个备份集。实施这些备份的描述如下：

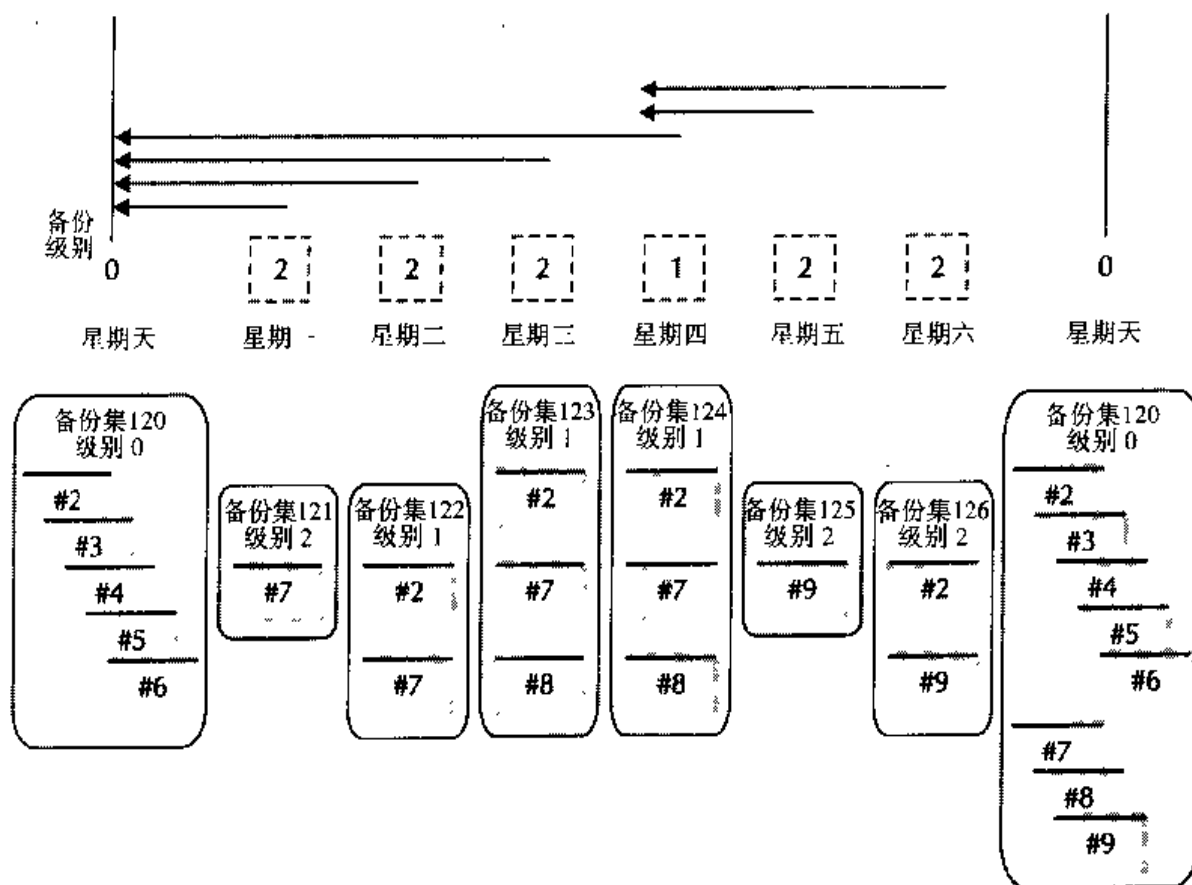


图12-2 累积增量备份基线示例

- **星期天** 进行级别为0的增量备份。数据块#2、#3、#4、#5和#6被拷贝到备份集120中。这些块组成了选为备份的数据文件中的所有使用过的数据块。级别为0的增量备份既不是累

积型也不是差异型。这是作为增量备份策略的一个基准的全备份。

- **星期一** 进行级别为2的累积增量备份。自从星期天备份以来，数据文件中更改的块是#7，其中充满了数据。因此，这个数据块是惟一被拷贝到备份集121中的数据块。
- **星期二** 进行级别为2的差异增量备份。自从星期一的备份以来，数据块#2中的一行数据被更新了。自从星期天备份以来的两个更改的数据块都被拷贝到备份集122中。
- **星期三** 进行级别为2的累积增量备份。自从星期二的备份以来，在数据块#8中插入行数据。自从星期天备份以来的三个更改的数据块都被拷贝到备份集123中。
- **星期四** 进行级别为1的累积增量备份。向备份集123中拷贝所有自最近一次级别为1备份以来更改过的数据块。自星期天的0级别备份以来所有更改的数据块都被包含在这个备份集中（#2、#7和#8）。
- **星期五** 因为自从星期四的备份以来，数据增加到了数据块#9，所以级别为2的增量备份将#9拷贝到备份集125中。
- **星期六** 因为自从星期四的备份以来，数据块#2和#9中数据被更改了，所以级别为2的增量备份将#2和#9块拷贝到备份集126中。
- **星期天** 进行级别为0的增量备份，将所有使用过的数据块拷贝到备份集127中。这时又开始新一轮的循环备份。

这两个图及其解释说明了RMAN是如何进行差异增量备份和累积增量备份的。

现在了解了增量备份，但RMAN是如何还原数据文件的呢？在第14章中有这部分详细的介绍，但简而言之，增量备份是逐个层层还原以便及时前滚数据文件。如果选择了差异增量备份策略（如图12-1），那么在星期六的级别2的增量备份之后，如果数据文件在星期六中午丢失了，则RMAN将采取以下备份进行还原：

1) **星期天的增量级别为0** 包含数据文件中所有使用过的数据块。作为后续还原工作的基准，必须还原级别为0的备份。

2) **星期二的增量级别为1** 还原所有自星期天备份被还原以来所有更改过的数据块。

3) **星期五的增量级别为2** 还原所有自星期二备份以来所有更改过的数据块。

4) **星期六的增量级别为2** 还原所有自星期五备份以来所有更改过的数据块。

5) **最后** 应用所有自星期二备份以来生成的重做日志。

这样，应用前面描述的差异增量备份策略使用3个备份用于将数据库还原并恢复到星期六的中午时的状态。

如果选择了累积增量备份策略（如图12-1），那么在星期六的级别2的增量备份之后，如果数据文件在星期六中午丢失了，则RMAN将采取以下备份进行还原：

1) **星期天的增量级别为0** 包含数据文件中所有使用过的数据块。作为后续还原工作的基准，必须还原级别为0的备份。

2) **星期二的增量级别为1** 还原所有自星期天备份被还原以来所有更改过的数据块。

3) **星期六的增量级别为2** 还原所有自星期二备份以来所有更改过的数据块。

4) **最后** 应用所有自星期二备份以来生成的重做日志。

这样，应用前面描述的累积增量备份策略使用3个备份用于将数据库还原并恢复到星期六中

午时的状态。

选择应用差异还是累积备份策略, 需要根据备份速度/大小与还原和恢复的速度的比较确定:

- 差异增量备份将比累积备份少拷贝数据块, 因此在较短的时间内生成较少的备份片。差异备份的缺点是: 因为需要更多的备份集, 所以还原时间较多。
- 因为在还原数据库文件时需要较少的备份集, 所以累积增量还原速度较快。而累积备份的缺点是: 需要对同样级别的备份拷贝相同的数据块。拷贝相同的数据块将意味着生成较大的备份片文件且备份需要较长时间。

因为备份集中只包含了使用过的和更改过的数据库块, 所以备份集将比映像拷贝要小。使用备份集的主要原因就是因为备份文件比映像拷贝和用户管理操作要节省空间。当使用RMAN备份命令保护数据库时, 应参考以下情况和建议:

- 可以从RMAN客户端进行备份, 但备份集总是在连接到服务器的磁带或磁盘上生成。
- RMAN并不备份SQL*Net配置文件、参数文件、告警日志和口令文件。
- 必须使用与备份的目标数据库相同版本的RMAN执行体。

2. 完全备份 (full backup)

对于完全备份, 每个数据文件中所有使用过的数据块都将被拷贝到数据集中。RMAN可以还原完全备份并应用归档重做日志将数据库及时回退到以前的状态。完全备份拷贝的数据块与增量级别为0的备份拷贝是一样的, 但却不是增量备份。增量备份选择拷贝的数据块是基于以前的增量备份 (非完全备份)。例如, 通过星期天增量级别为0的备份对一个数据文件进行了备份。在星期一, 进行完全备份, 在星期二进行了增量级别为1的备份, 则在星期二备份的数据块并没有考虑在星期一的备份, 而只备份在最近一次增量备份 (在星期天进行的备份) 以来所有更改过的数据块。

12.1.3 映像拷贝

拷贝命令将生成一个完整数据库文件的映像拷贝, 这个数据库文件可以是数据文件、控制文件或者归档日志文件。RMAN生成的文件拷贝不需要应用任何特殊的格式化就能与原始文件区别开来。因此, 当数据库打开或关闭时 (类似于第4章中的情况), 该文件可以当作操作系统拷贝的文件应用。惟一不同的是, 该拷贝自动在目标数据库控制文件和可选的恢复目录中注册了。因为RMAN保留了对更改数据块的SCN的记录, 所以包含有正在被拷贝的数据文件的表空间不需要设置为热备份模式。SCN用于确定恢复时需要应用的重做日志文件。用RMAN生成的数据文件拷贝可以被当作增量级别为0的备份, 该备份作为后续增量备份的基准。

级别为0的增量备份或完全备份生成了一个数据文件中所有使用过的数据块的备份。映像拷贝也完成同样的工作, 同时也包含了所有空的数据块。在应用任何其他增量备份或重做之前, RMAN可以应用映像拷贝、完全备份或级别为0的备份作为还原的基准。映像拷贝不能用任何其他增量级别 (1~4) 生成。RMAN只能在磁盘上生成映像拷贝, 不能使用SBT_TAPE通道生成映像拷贝。

映像拷贝与差异和累积增量备份相比有一个主要的优点: 可以在还原过程中快速应用映像拷贝。例如, 如果丢失了数据文件, 让RMAN切换到映像拷贝、应用重做, 则该数据文件又可以被访问了。假设对数据库其他部分需要进行同样数量的重做, 则这个过程比从磁盘或磁带上

的备份集中恢复数据文件要快得多。

但应用映像拷贝最大的缺点是，需要占用与原始空间一样大的磁盘空间。如果需要将映像拷贝保存到磁带上以节省磁盘空间，则需要在RMAN使用之前将其拷贝到磁盘上。使用备份比映像拷贝能更有效地利用空间资源。

12.1.4 代理拷贝

Oracle增加了一个称为代理拷贝的功能，以增强发布的介质管理层（MML）API。代理拷贝允许MML控制如何进行数据文件的读取和备份。RMAN向MML层传递需要备份的数据文件列表，MML决定如何进行最佳的备份。应用代理拷贝的优点是可以利用介质管理层的高级特性，例如从磁盘到磁带的直接传输。使用代理拷贝的一个缺点是当数据库打开时，每个要被备份的数据文件都要在内部被设置为热备份模式，这就意味着在备份期间需要生成更多的重做，类似于手工将表空间设置为热备份模式。Oracle自动处理每个数据文件的状态变化，因此需要管理的工作比热备份少。RMAN也对代理拷贝时的备份进行归类，所以RMAN所有的自动化工作仍然适用。

代理拷贝对RMAN来说是一种比较新的功能，而且也只被少数的MML支持。可以联系有关介质管理供应商来获得更多的信息。

12.1.5 备份类型比较

了解了RMAN可以进行的映像拷贝、备份集和代理拷贝之后，需要知道在各自的Oracle环境下，哪种类型是最佳的备份方式。每种备份类型都有自己的优缺点。如果愿意可以进行这三种类型的备份。表12-2提供了每种备份类型的优缺点一览表。

通过本章的学习，如图12-3所示，可以用RMAN COPY和BACKUP命令来保护自己的PRACTICE数据库了。可以生成数据库所有数据文件和控制文件的映像拷贝，也可以生成只包含了数据文件中已经使用过的数据块以及归档日志文件和控制文件中所有的数据块的备份集。

表12-2 RMAN备份类型的优缺点

备份类型	优 点	缺 点
备份	只包含了使用过或更改过的数据块，备份文件比较小。如果需要节约备份空间则生成备份集。备份集动态为备份决定数据库文件。可以被保存到磁盘或磁带上	因为从备份集还原需从一个或多个备份片中重新构造数据库文件，所以恢复时间较长
映像拷贝	可以快速还原文件拷贝。如果恢复时间很关键则可以采用映像拷贝	文件拷贝需要与数据库文件同样的大小，因此需要比备份集更多的空间 每个映像拷贝必须在备份命令中明确指定名称。生成动态脚本比较困难 映像拷贝只能写到磁盘上
代理拷贝	可以利用MML的高级特性。较少考虑备份片和备份集的大小	需要第三方供应商支持，增加支持复杂特性的文档不全面

该练习与第4章和第5章中的用户备份练习非常相似。这里将要用RMAN创建PRACTICE数据库中所有数据文件和控制文件的拷贝。

任务1：创建RMAN拷贝脚本

在数据文件拷贝命令中，必须明确要拷贝的每个文件，并为拷贝的文件指定名称。一个生成数据库文件列表的快捷方法就是运行REPORT SCHEMA命令。应用该列表生成RMAN的拷贝脚本。从输出结果中拷贝文件名并粘贴到文件b_copy.rcv中。根据下面列出的内容编辑该b_copy.rcv文件。该脚本将拷贝数据文件和控制文件（尽管RMAN的脚本可以是任何名称和文件扩展名，但最好将这些文件的扩展名设置为rcv（ReCoVery））。

```
run {
  allocate channel d1 type disk;
  copy
  datafile 1 to '/oradata/PRACTICE/backup/ch12/system01.dbf.bak',
  datafile 2 to '/oradata/PRACTICE/backup/ch12/rbs01.dbf.bak',
  datafile 3 to '/oradata/PRACTICE/backup/ch12/users01.dbf.bak',
  datafile 4 to '/oradata/PRACTICE/backup/ch12/temp01.dbf.bak',
  datafile 5 to '/oradata/PRACTICE/backup/ch12/tools01.dbf.bak',
  datafile 6 to '/oradata/PRACTICE/backup/ch12/indx01.dbf.bak',
  datafile 7 to '/oradata/PRACTICE/backup/ch12/users02.dbf.bak',
  current controlfile to '/oradata/PRACTICE/backup/ch12/backup.ctl';
}
```

分配完通道后，执行拷贝命令，RMAN将数据文件号映像拷贝到所提供的文件名。当前的控制文件也被拷贝。在该模块执行完成后将释放分配的通道。

任务2：运行RMAN拷贝脚本

连接到目标和目录数据库，运行刚才生成的脚本，然后用@符号来运行RMAN命令文件：

```
RMAN> @b_copy.rcv
```

在脚本运行期间，将会出现类似以下的信息：

```
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: d1
RMAN-08500: channel d1: sid=16 devtype=DISK
```

这些消息是在运行模块中分配的通道命令中得到的，而RMAN将其当作消息生成。在9i中，这些消息将被更改，因为并没有显示任何错误消息编号。这些消息说明RMAN成功地编译并执行了分配命令。通道的名称为d1，目标数据库PRACTICE的会话ID号为16，而生成的通道类型为DISK。

```
RMAN-08501: output filename=/oradata/PRACTICE/backup/ch12/system01.dbf.bak
recid=1 stamp=446229077
```

可以看到，拷贝命令为数据文件1生成了一个映像拷贝文件system01.dbf.bak。

在该脚本的结尾，会得到恢复目录的部分再同步：

```
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
```

```
RMAN-08005: partial resync complete
```

```
RMAN-08031: released channel: d1
```

在再同步期间，RMAN的拷贝记录将与恢复目录同步。在运行模块的最后，d1通道将从数据库自动释放。

任务3：验证RMAN Copy脚本

可以通过检查RMAN脚本的输出结果来验证映像拷贝，当然也可以通过其他方法来验证是否成功地创建了数据文件和控制文件的映像拷贝。首先检查/oradata/PRACTICE/backup/ch12/目录，这个目录下将有为所有文件生成的一个拷贝。接着，可以在RMAN命令提示符下运行list copy命令：

```
RMAN> list copy;
RMAN-03022: compiling command: list
List of Datafile Copies
Key File S Completion time Ckp SCN Ckp time Name
---
224 1 A 12-JAN-01 78045 12-JAN-01 /oradata/PRACTICE/backup/ch12/system01.dbf.bak
225 3 A 12-JAN-01 78047 12-JAN-01 /oradata/PRACTICE/backup/ch12/users01.dbf.bak
226 7 A 12-JAN-01 78051 12-JAN-01 /oradata/PRACTICE/backup/ch12/users02.dbf.bak
```

这个输出显示了RMAN从以前脚本拷贝的每个数据文件，还有以前进行的其他拷贝。每个数据文件都有自己的备份号（惟一标识）。应用备份编号可以维护每个映像文件拷贝。标志为S的栏目则提供了备份片的状态（AVAILABLE、UNAVAILABLE或EXPIRED）。注意检查点SCN以及每个文件的时间。在拷贝命令执行期间，每个数据文件都要经历检查点。如果这些映像拷贝在数据库打开时被还原，则必须应用重做，以保证数据库的其余部分与这些文件一致。LIST COPY命令并没有显示控制文件，这是因为默认显示的文件类型是数据文件。为了显示控制文件的拷贝，需要在LIST COPY命令中增加OF CONTROLFILE：

```
RMAN> list copy of controlfile;
```

现在已经用RMAN对一个打开的数据库进行了备份。

如何生成一个用来创建数据库所有数据文件和控制文件映像拷贝的RMAN脚本？在SQL*Plus中，可以从控制文件动态视图v\$datafile中获得数据库所有数据文件的一个列表。将查询的结果与其他所需要的RMAN命令一起组合到一个文件中。下面的命令可以在SQL*Plus中生成一个RMAN脚本文件。其中需要注意的是用SQL内建的函数INSTR和SUBSTR将每个文件名从其目录中分离出来。

```
set feedback off pagesize 0 heading off verify off
set linesize 100 trimspool on
define dir = '/oradata/PRACTICE/backup/user/ch12/'
define fil = '/tmp/b_copy_whole.rcv'
define div = '/'
spool &fil
prompt run {
prompt allocate channel d1 type disk;
prompt copy
select 'datafile ' || file# || ' to ' ||
```

```
'&dir.'||substr(name,(instr(name,'&div',-1)+1))||'.bak ',
from v$datafile;
prompt current controlfile to '&dir.backup.ctl; }';
spool off;
```

练习12.2: 创建恢复管理器的整体备份

在本练习中，将要生成一个关于整个PRACTICE数据库的全部数据块的备份，其中包括控制文件。将会在数据库打开和关闭时分别进行整体数据库的备份。如果没有将目标数据库设置为ARCHIVELOG模式，则必须在数据库加载但没有打开时进行备份操作。

任务描述	时间（分钟）
1. 生成RMAN打开的数据库备份脚本	5
2. 运行RMAN打开的数据库备份脚本	5
3. 验证RMAN打开的数据库备份脚本	5
4. 运行RMAN关闭的数据库备份脚本	5
总计时间	20

在练习过程中，将讲解执行整体数据库备份的基本命令。当运行该脚本时，屏幕上的输出将与list命令所得到的结果相对应。我将指出其中的对应部分，以便读者将备份的运行输出结果与list命令输出结果联系起来。

任务1: 生成RMAN打开的数据库备份脚本

在本任务中，将要在数据库打开时进行备份操作。下面显示的脚本与任务1中的脚本基本相同，只有一点例外：数据库是打开的而不是关闭的。

```
1.  run {
2.      allocate channel d1 type disk;
3.      backup database incremental 0
4.      format '/oradata/PRACTICE/backup/ch12/db_%d_%s_%p_%t'
5.      tag = 'WHOLE_INC0';
6.  }
```

- 第1行：backup命令必须包括在run命令中。Run命令必须在同一行（或下一非空、非注释行）中紧跟一个左花括号。Run命令标明了必须作为一个单元运行的代码段。所有在花括号中包含的命令组成了命令单元。
- 第2行：备份使用通道从数据库文件向备份介质拷贝数据块。给在运行单元中需要的备份通道分配一个名称。
- 第3行：backup命令确定了计划保护的数据库文件内容。在这一行中，将要为整个PRACTICE数据库备份数据块。当备份system表空间时，RMAN包括了一个当前的控制文件的备份。Backup database命令以第5行的分号作为结束。
- 第4行：format关键词告诉backup命令将备份片文件放置于何处以及备份片的名称是什么。本行语句通知RMAN在/oradata/PRACTICE/backup/ch12目录中生成一个文件。如果没有指定format，RMAN将以%U命名备份片并将其保存\$ORACLE_HOME/dbs（Linux环境）目录或\$ORACLE_HOME/database（Windows环境）目录下。

- 第5行: Tag可以提供容易辨认的名称,以便在RMAN部署中识别备份。虽然总可以通过备份集的数字编号来查找一个指定的备份集,但是tag有助于更简单地管理备份。Tag可以重用,如果再次运行该备份脚本,则有两个备份集都有WHOLE_INC0标记。
- 第6行: 这个右花括号与第1行中的左花括号相匹配。这个括号标志着run命令的结束。所有在run命令中的命令被编译并作为一个单元来执行。

这个脚本最有魅力的地方是:它如此简洁!只需要指定RMAN去备份数据库。RMAN查询目标数据库的控制文件为备份的数据文件生成一个列表,并在备份集中生成存储数据块的文件。RMAN知道需要拷贝哪些块并忽略那些不需要的块。RMAN在控制文件和目录中记录备份的动作。我们只需要生成并运行几个简单的命令,即可由服务器管理来完成全部工作。

任务2: 运行RMAN打开的数据库备份脚本

连接到目标数据库PRACTICE和目录数据库RCAT后,执行前面任务中的备份数据库脚本:

```
RMAN> @b_whole_inc0.rcv
```

当脚本运行时,将会有许多RMAN的消息。下面的一些消息是我备份时出现的输出结果。我将其中一些重要的输出消息用粗体字符标示出来,并将它们与刚才讨论的实际的命令和操作联系起来讲解。

```
RMAN-08500: channel dl: sid=14 devtype=DISK
RMAN-08008: channel dl: starting incremental level 0 datafile backupset
RMAN-08502: set_count=120 set_stamp=446988820 creation_time=12-JAN-02
RMAN-08522: input datafile fno=00001 name=/oradata/PRACTICE/system01.dbf
RMAN-08011: including current controlfile in backupset
RMAN-08522: input datafile fno=00002 name=/oradata/PRACTICE/rbs01.dbf
RMAN-08522: input datafile fno=00004 name=/oradata/PRACTICE/temp01.dbf
RMAN-08522: input datafile fno=00005 name=/oradata/PRACTICE/tools01.dbf
RMAN-08522: input datafile fno=00006 name=/oradata/PRACTICE/indx01.dbf
RMAN-08522: input datafile fno=00003 name=/oradata/PRACTICE/users01.dbf
RMAN-08522: input datafile fno=00007 name=/oradata/PRACTICE/users02.dbf
RMAN-08013: channel dl: piece 1 created
RMAN-08503: piece handle=/oradata/PRACTICE/backup/ch12/db_PRACTICE_120_1_44698882E
RMAN-08525: backup set complete, elapsed time: 00:00:57
RMAN-08005: partial resync complete
RMAN-08031: released channel: dl
RMAN-06400: database opened
```

这个输出显示了下面几个重要的问题:

- **sid=14** 当在PRACTICE数据库上分配了通道来执行备份时,连接的会话id是14。
- **starting incremental level 0 datafile backupset** 该备份是一个增量级别为0的备份。所有数据文件中全部使用过的数据块都将被拷贝到一个备份片中。
- **set_count=120** 备份集编号为120。这是该数据库的第120个RMAN备份集。注意:该编号与备份键值不同。
- **including current controlfile in backupset** 当前的控制文件与system表空间数据文件一起被拷贝。System数据文件拷贝的消息之后就是这个消息。

- `piece handle=/oradata/PRACTICE/backup/ch12/db_PRACTICE_120_1_44698882E`
这个文件就是实际包含了整个PRACTICE数据库数据块级备份的文件。在format语句中应用的替代变量被用于这个文件名（%d-数据库名=PRACTICE，%s-备份集数=3，%p-备份片编号=1，%t-备份时间=44698882E）。这个备份时间不是常见的时间格式。
- `elapsed time: 00:00:57` 备份用时为57秒。

任务3：验证RMAN打开的数据库备份脚本

前面已经运行了备份操作，下面使用LIST命令来检查已经运行的备份结果：

```
RMAN> list backup;
RMAN-03022: compiling command: list
List of Backup Sets
-----
Key          Recid      Stamp          LV Set Stamp  Set Count  Completion Time
-----
256          120        446988874      0  446988820    120        12-JAN-02

List of Backup Pieces
-----
Key Pc# Cp# Status      Completion Time      Piece Name
-----
257    1  1  AVAILABLE  12-JAN-02            ./db_PRACTICE_120_1_44698882E

List of Datafiles Included
-----
File Name                                     LV Type Ckp SCN      Ckp Time
-----
1    /oradata/PRACTICE/system01.dbf            0  Inc   78342      12-JAN-02
2    /oradata/PRACTICE/rbs01.dbf              0  Inc   78342      12-JAN-02
3    /oradata/PRACTICE/users01.dbf             0  Inc   78342      12-JAN-02
4    /oradata/PRACTICE/temp01.dbf              0  Inc   78342      12-JAN-02
5    /oradata/PRACTICE/tools01.dbf              0  Inc   78342      12-JAN-02
6    /oradata/PRACTICE/indx01.dbf              0  Inc   78342      12-JAN-02
7    /oradata/PRACTICE/users02.dbf             0  Inc   78342      12-JAN-02
```

这个输出提供了刚才生成的备份的大量信息：

- **List of backup sets** 备份集键值是120，该键是这个RMAN备份运行的惟一标识。注意，这个数字与实际运行备份所得到的输出是一致的。备份是在1月12日运行并完成的。LV是备份的级别，0级备份包含了每个数据文件中所有使用过的数据块。
- **List of backup pieces** 该备份集只有一个备份片。包含了数据块的实际的操作系统文件是/oradata/PRACTICE/backup/ch12目录下的db_PRACTICE_120_1_44698882E。RMAN脚本中的format语句产生了这个文件名。
- **List of datafiles included** 列出了这个备份中包括的每个数据文件。LV列显示了数据块拷贝的级别，而type则显示该备份是一个增量文件备份。可以浏览每个数据文件备份的检查点SCN值。

验证备份工作的最后一个步骤就是：在操作系统下，定位到/oradata/PRACTICE/backup/ch12目录下，在这里将有一个名为db_PRACTICE_120_1_44698882E的备份片文件。所有使用过的数据文件块和整个控制文件都包含在这一个文件中！

该文件的大小可能是60MB左右。请考虑：所有数据文件的总和大约是200MB，控制文件大

约是3MB，那么如何将203MB的大小的文件压缩到只有60MB的文件中呢？依据是只有使用过的数据块才被拷贝到这个备份片中。假定有20MB的索引表空间，因为只需要存储少量的索引，所以可能只用了20-50个数据块，而表空间中其他的数据块都没有被使用（从来没有使用）。因此，当RMAN只拷贝该数据文件中使用过的数据块时，没有拷贝多少数据块。这就是RMAN如何在其备份文件中节约空间的原因。

任务作完了！RMAN已经将整个数据库的每个数据块都备份到了一个文件中。

警告 当命名备份文件片时，应确保使用了惟一的文件名，否则就可能覆盖以前拷贝的备份片文件。许多RMAN用户指定format特殊标识%d_%s %p_ %t，这样就可以产生惟一的名称，其中包含了数据库名称、备份集编号、备份片编号以及编码过的备份操作的时间。

任务4：运行RMAN关闭的数据库备份脚本

当控制文件加载但数据库没有打开时，也可以用RMAN来备份关闭的数据库。如果数据库完整地关闭了，则所有数据文件都在其头部包含了一个一致的SCN值。在数据文件备份期间，这个数据文件的一致SCN将保存在RMAN备份的控制文件中，如下所示：

```
1.  shutdown immediate;
2.  startup mount;
3.  run {
4.      allocate channel d1 type disk;
5.      allocate channel d2 type disk;
6.      backup database incremental 0
7.          format '/oradata/PRACTICE/backup/ch12/db_%d_%s_%p_%t'
8.          tag = 'WHOLE_INC0_CLOSED';
9.  }
10. alter database backup controlfile to '/temp/backup.ctl' reuse;
11. alter database open;
```

这个文件中包含的11行语句是在PRACTICE数据库关闭状态下，对数据库的所有数据文件和控制文件执行增量基准数据块级备份时需要的。下面将对该脚本中的每一行以及每条命令所完成的功能进行描述。

当分配了几个通道时，RMAN试图创建同等大小的备份集文件。这样做的目的是在分配的通道中平均I/O的流量。RMAN假定被备份的每个数据文件中每个数据块都将被包含在备份集中。如果实际备份的块比预计的少，则这些备份集的大小就会不同。

RMAN对于关闭的数据库的备份命令与打开的数据库的命令是一样的。实际上，对前面的备份数据库的脚本文件增加SHUTDOWN、STARTUP和ALTER DATABASE命令就可以运行该脚本。为了说明问题，我将针对RMAN备份修改该脚本：

- 第1行：在备份一个关闭的数据库之前，该数据库必须被关闭。在RMAN提示符下，可以执行shutdown和startup等数据库控制命令。
- 第2行：在运行任何RMAN备份命令时，必须将数据库的控制文件加载。为什么要这样操作呢？因为所有的数据库模式信息都保存在目标控制文件中。同时，RMAN备份操作在该控制文件中进行存储，所以目标控制文件必须能被目标实例打开。

- 第4-5行：在这个备份过程中，RMAN将为备份打开两个通道。每个通道在目标数据库上创建一个会话，并且在磁盘上打开一个备份片。多个通道允许RMAN以并行方式执行数据块拷贝操作。使用多个通道使RMAN提高向磁盘或磁带的输出速度。在这个PRACTICE数据库，一个通道足够了。（我为了演示的目的使用了两个通道。）
- 第7-8行：注意，FORMAT和TAG关键字的顺序不同。在backup命令中，可以交换这两个关键字的顺序。
- 第10行：在RMAN提示符下，可以运行其他数据库控制命令。在这一行中，将一个控制文件的备份拷贝保存在备份目录下。这个alter database命令与RMAN的backup命令不同，因为这个控制文件的拷贝没有被记录到目标数据库控制文件或目录表中。
- 第11行：当完成备份后，可以为用户打开该数据库。

为了运行刚才生成的备份脚本，启动RMAN并连接到目标数据库PRACTICE和目录数据库RCAT上。使用@标示符来运行刚才创建的脚本：

```
RMAN> @b_whole_inc0_closed.rcv
```

当运行完成这个关闭数据库的备份脚本后，可以使用list命令来查看备份集编号、备份片和备份的文件。尤其要注意的是，一个备份命令生成了两个备份集和两个备份片，原因是为该备份分配了两个通道。

注意，这个输出与任务2的输出基本相同，不同的是这里生成了两个备份集并写入到了各自的备份片中：

```
RMAN-08502: set_count=121 set_stamp=446987639 creation_time=12-JAN-02
RMAN-08522: input datafile fno=00002 name=/oradata/PRACTICE/rbs01.dbf
RMAN-08522: input datafile fno=00004 name=/oradata/PRACTICE/temp01.dbf
RMAN-08522: input datafile fno=00005 name=/oradata/PRACTICE/tools01.dbf
RMAN-08522: input datafile fno=00006 name=/oradata/PRACTICE/indx01.dbf
RMAN-08502: set_count=122 set_stamp=446987639 creation_time=12-JAN-02
RMAN-08522: input datafile fno=00001 name=/oradata/PRACTICE/system01.dbf
RMAN-08011: including current controlfile in backupset
RMAN-08522: input datafile fno=00003 name=/oradata/PRACTICE/users01.dbf
RMAN-08522: input datafile fno=00007 name=/oradata/PRACTICE/users02.dbf
RMAN-08503: piece handle=/oradata/PRACTICE/backup/ch12/db_PRACTICE_121_1_44698763E
RMAN-08013: channel d2: piece 1 created
RMAN-08503: piece handle=/oradata/PRACTICE/backup/ch12/db_PRACTICE_122_1_44698763E
```

这个输出显示生成了两个备份集和备份片。SYSTEM和USERS表空间数据文件包含在备份集122的备份片db_PRACTICE_122_1_44698763E中，其他表空间的数据文件则被保存在备份集121的备份片db_PRACTICE_121_1_44698763E中。其中一个备份集总共耗时50秒完成，而另一个耗时47秒。

备份过程中，RMAN从几个数据文件中拷贝使用过的数据块。在备份结束时，PRACTICE数据库的控制文件被更新。最后，目录被PRACTICE的控制文件同步，这样就可以包括新创建的备份记录了。

检查备份路径下这个备份的实际备份片文件名。注意备份集122比备份集121要大，因为传

送到每个通道/备份集的数据文件包含不同的使用过的数据块。

简而言之，已经为备份PRACTICE数据库中的所有数据文件以及当前的控制文件生成了RMAN备份脚本，可以运行该脚本并观察其输出结果。然后，用RMAN LIST命令、用SQL查询目录数据库并检查操作系统上的文件来验证备份的结果。真是可喜可贺，已经完成了第一个RMAN备份！

注意 如果用RMAN在一个关闭的数据库上执行备份，则不需要执行恢复就可以恢复并打开该数据库。可以这样做的原因是：在备份期间数据文件是一致的，因此，还原时这些数据文件也是一致的。如果在一个打开的数据库上用RMAN进行备份，则当这个备份被还原后需要恢复，以使文件保持一致。

练习12.3：创建恢复管理器的增量式备份

前面讲述的数据文件备份功能都需要拷贝整个数据文件或数据文件中所有使用过的数据块。对于像PRACTICE这样的小型数据库，拷贝整个文件或所有数据块并不会占用很多空间、花费很长的时间。然而，很少有类似PRACTICE这样小的产品数据库。有大量数据文件的大型数据库的备份需要很长时间并占用大量空间。通常情况下，数据库的大量数据在两次备份之间是保持不变的。Oracle现在提供了一种功能，只拷贝最近修改过的数据块，即数据块级的增量备份。只要需要，就可以用基本的增量级别为0的备份和需要的后续级别的增量备份一起还原与恢复数据库。因为，不需要重复备份那些没有更改的数据块，所以在增量级别备份中生成的备份片将节省大量的空间。

任务描述	时间（分钟）
1. 生成RMAN增量备份脚本	5
2. 运行RMAN增量备份脚本	5
3. 验证RMAN增量备份脚本	5
4. 运行另一个RMAN增量备份脚本	5
总计时间	20

在上一个练习中，在数据库打开状态下，为整个PRACTICE数据库以及控制文件生成了一个增量级别为0的基本备份。在本练习中，将生成一个累积增量备份级为1的备份。接着，将要建立一个增量级别为2的备份。级别1的备份将拷贝自级别0备份以来所有更改过的数据块。而级别2的备份将拷贝所有自级别1备份以来所有更改过的数据块。

任务1：生成RMAN增量备份脚本

下面代码所包含的命令将对整个PRACTICE数据库执行差异增量备份：

```
run {
  allocate channel d1 type disk;
  backup
    incremental cumulative level = 1
    database
    format ò/oracle/PRACTICE/backup/ch12/db_&d_&s_&p_&t1
```



```

tag = 'WHOLE_INC1';
}

```

上面的命令与本章前面给出的脚本非常相似。除了tag名称和备份片文件名的format不同以外，还有一个不同点：这个脚本包括了一个通知RMAN这个数据文件的备份是一个累积增量级别1的备份。因为这个脚本需要对整个数据库备份，所以所有数据文件中已经更改过的数据块都将被拷贝到备份集的备份片中。

任务2：运行RMAN增量备份脚本

与前面的过程类似，连接到目标数据库PRACTICE和目录数据库上执行备份操作：

```
RMAN> @b_whole_inc1.rcv
```

当执行b_whole_inc1.rcv时，这个备份需要占用相同的时间，但其备份片文件却要小得多。这是因为包含了更改过的数据字典的数据块很少，也没有SCOTT表，TINA.DATE_LOG中只插入了几行记录。

任务3：验证RMAN增量备份脚本

再次使用list命令来检查备份的结果并检查操作系统以确认备份片文件是否存在。是否看到哪里有一个样例模板了？这个命令的输出除了以下部分外，其他都与它们的备份命令的输出命令相同：

- 键值和备份片名称不同
- incremental列的数值不同

任务4：运行另一个RMAN增量备份脚本

在讨论RMAN归档日志备份以前，需要运行另一个类似本练习的备份。运行增量级别为2的备份。为了完成这个任务，生成一个名为b_whole_inc1.rcv文件的拷贝文件b_whole_inc2.rcv。打开这个文件并将所有的‘1’修改为‘2’。例如：在b_whole_inc2.rcv中：

- incremental = 1 修改为 incremental = 2
- tag = 'WHOLE_INC1' 修改为 tag = 'WHOLE_INC2'

在级别1和级别2备份期间没有任何数据块更改，所以没有再次拷贝数据块。

练习12.4：创建恢复管理器的归档日志备份

当PRACTICE数据库继续操作时，就会在归档路径下产生归档日志文件。这些文件绝对需要保护，否则就不能将数据库恢复到过去某个时间点或当前时间状态。在第5章中介绍了如何以用户管理的方法管理这些归档文件的备份，那么如何使用RMAN来管理这些至关重要的文件呢？使用BACKUP ARCHIVELOG命令可以将这些归档日志文件拷贝到备份集的一个备份片中。当通知RMAN来备份这些文件时，可以告诉RMAN通过日志序列范围或日志生成的时间范围等限制条件来选择需要备份的文件，或者可以将它们全部拷贝到一个备份集中。

RMAN也允许对这些文件执行强制管理。作为数据库的操作，重做生成并存储在归档日志文件中。有些情况下，这些文件将会占满磁盘。RMAN的一个DELETE INPUT选项可以从磁盘上删除已经拷贝过的归档日志文件。应在一个谨慎归档备份策略下使用这个选项，可以放心地使用RMAN来保护这些文件。

任务描述	时间 (分钟)
1. 创建RMAN归档日志备份脚本	5
2. 运行RMAN归档日志备份脚本	5
3. 验证RMAN备份脚本	5
总计时间	15

在这个练习期间，将会看到备份的基本命令并删除PRACTICE数据库的归档日志文件。我将讨论这些基本的命令以及如何应用这些命令来保护并清理这些PRACTICE数据生成的文件。

任务1：创建RMAN归档日志备份脚本

很容易使用RMAN来备份归档文件：只需要简单地创建一个RMAN的命令来备份归档日志。RMAN可以按如下的方式完成所有的工作：

```
run {
  allocate channel d1 type disk;
  sql "ALTER SYSTEM ARCHIVE LOG CURRENT";
  backup archivelog all
    format '/oradata/PRACTICE/backup/ch12/ar_%d_%s_%p_%t';
}
```

在这些精炼的run命令中，定义了需要备份归档日志的路径、磁带或磁盘。然后，可以有选择地确定当前联机的重做日志文件是否已经归档了。最后，通知RMAN拷贝所有的归档日志到一个备份集的备份片中。备份片的位置是/oradata/PRACTICE/backup/ch12。

任务2：运行RMAN归档日志备份脚本

连接到目标数据库和目录数据库后运行这个脚本：

```
RMAN> @b_archive.rcv
```

在这个脚本命令的执行过程中，会得到由RMAN产生的输出消息，这些消息显示了备份片的名称以及正在被备份的归档序列。

```
RMAN-08502: set_count=124 set_stamp=446989243 creation_time=12-JAN-02
RMAN-08014: channel d1: specifying archivelog(s) in backup set
RMAN-08504: input archivelog thread=1 sequence=126 recid=1 stamp=446985969
RMAN-08504: input archivelog thread=1 sequence=127 recid=2 stamp=446986041
RMAN-08504: input archivelog thread=1 sequence=128 recid=3 stamp=446989240
RMAN-08013: channel d1: piece 1 created
RMAN-08503: piece handle=/oradata/PRACTICE/backup/ch12/ar_PRACTICE_7_1_446989243
```

任务3：验证RMAN备份脚本

运行LIST命令并查看v\$表，检查该文件的输出路径。

```
RMAN> list backupset of archivelog all;
```

这个命令将列出所有刚才为归档日志文件所创建的备份。如果生成了大量的备份，则输出结果会滚屏输出，显示拷贝了的归档日志序列的序列号以及每个日志文件的SCN域。列出这些归档备份以后，需要检查操作系统的文件。应该在目录/oradata/PRACTICE/backup/ch12中保存备份输出所产生的备份片文件。在下面部署一节中，我将讲述如何将归档备份合并到一个示例备份策略中。

多归档目的地的RMAN备份

如果目标数据库有两个或更多的归档目的地，则备份归档命令将只能备份其中一个（通常是第一个）目的地。为了备份每个目的地的归档日志文件，需要为每个归档目的地分别创建一个通道。然后，当向分配的通道备份每个目的地时，需要增加LIKE关键字。跟随LIKE关键字后的字符串应该包括归档目的路径和%通配符，以备份该路径下的任何归档日志文件。

12.2 RMAN的部署

前面了解了如何方便地进行增量和归档日志备份，现在来讨论如何将增量置于一个示例规划中。这个示例的规划包括每日数据文件、控制文件以及归档日志的备份策略。当部署增量备份策略时，需要平衡一旦数据库失效后用备份资源（磁盘空间、磁带驱动、机器容量等）进行恢复所需的时间。一个称为标准的备份情况需要在周末进行增量级别为0的备份。然后，在整个星期内，需要进行不同的级别1或2的备份。这样每周循环可以使每一周都有一个基准增量备份以及每周内少量的增量备份。例如，可以在数据库的活动处于低谷时进行备份工作（例如，午夜2:00）：

- 星期天 清晨 0级别备份（运行b_whole_inc0.rcv脚本）
- 星期一 清晨 2级别备份（运行b_whole_inc2.rcv脚本）
- 星期二 清晨 2级别备份（运行b_whole_inc2.rcv脚本）
- 星期三 清晨 2级别备份（运行b_whole_inc2.rcv脚本）
- 星期四 清晨 1级别备份（运行b_whole_inc1.rcv脚本）
- 星期五 清晨 2级别备份（运行b_whole_inc2.rcv脚本）
- 星期六 清晨 2级别备份（运行b_whole_inc2.rcv脚本）

有了这个累积增量备份部署，就可以用至少三个备份集来还原与恢复工作。例如，如果在星期六中午发生了数据库失效，则RMAN restore命令将需要星期天、星期二和星期六的备份集。为了运行累积增量备份，需要在本练习描述的脚本中的incremental后增加CUMULATIVE关键字。

当为归档日志文件部署备份策略时，必须绝对确保已经获得了每个备份的归档日志文件的至少一个（最好多个）拷贝。还必须清理归档日志的目的地以防止写满文件系统以及中断数据库操作。RMAN有一个比较谨慎的部署选项DELETE INPUT，由此可以完成两个任务。归档日志备份策略在删除输入之前已经为每个归档日志文件做了三份拷贝。下面是如何为PRACTICE数据库实现每个归档日志文件的三份拷贝：

- 1) 在新的备份策略的前一两天内对PRACTICE数据库应用ARCHIVELOG ALL选项以实现为所有已经存在的归档日志文件做基准备份（运行b_archive.rcv）。
- 2) 每天为包括有最近两天的重做的所有已经存在的归档日志文件做备份（运行b_archive_2days.rcv）。
- 3) 每天为包括有从五天以前到前两天的重做日志文件进行备份（运行b_archive_delete3.rcv）

这个策略确保由数据库创建的每个归档日志文件都有至少三份拷贝。在执行这个归档日志备份策略时需要下面两个脚本：b_archive_2days.rcv 和 b_archive_delete3.rcv。

```
run {
    allocate channel d1 type disk;
```

```

sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
backup archivelog
  from time 'SYSDATE-2'
  format '/oradata/PRACTICE/backup/ch12/ar_%d_%s_%p_%t';
}

```

当为磁盘分配了一个通道并归档了当前联机重做日志后，备份自两天以前生成的所有归档日志。运行该脚本，则每天将备份前两天的归档日志文件。这个脚本没有删除备份过的归档日志。

```

run {
  allocate channel d1 type disk;
  sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
  backup archivelog
    from time 'SYSDATE-3' until time 'SYSDATE-2'
    format '/oradata/PRACTICE/backup/ch12/ar_%d_%s_%p_%t'
    delete input;
}

```

当运行archive_delete时，DELETE INPUT将通知RMAN在成功备份之后删除归档日志文件。当运行b_archive_delete3.rcv时，会出现如下的输出：

```

RMAN-08071: channel d1: deleting archivelog(s)
RMAN-08514: archivelog filename=/oradata/PRACTICE/archive/126.arc recid=125 stamp
           =443360938

```

该输出结果显示了被删除的文件。当这个脚本运行完成后，查看操作系统中的归档存放路径，被删除的文件126.arc将不再出现在磁盘上。为了解释地更清楚，请参看图12-4，注意备份集和归档文件。数据库每天生成3个重做日志，每天的下午一点运行两个备份脚本：b_archive_2days.rcv和b_archive_delete3.rcv。

- 星期二的备份 备份集1包括归档日志1、2、3。没有归档日志被删除。
- 星期三的备份 备份集2包括归档日志1-6。没有归档日志被删除。
- 星期四的备份 备份集3包括归档日志4-9。备份集4拷贝了归档日志1-3并从磁盘上将其删除。
- 星期五的备份 备份集5包括归档日志7-12。备份集6拷贝了归档日志4-6并从磁盘上将其删除。
- 星期六的备份 备份集7包括归档日志10-15。备份集8拷贝了归档日志7-9并从磁盘上将其删除。

这个备份和删除文件的过程是持续不断执行的。

技巧 当选择在一定的时间范围内备份时，任何在这个时间范围内的归档日志（包括重做）都将被包含在这个备份中。

如果要查看一个特定的归档日志文件已经被备份了多少次，则可以用如下方式的RMAN list命令获得：

```
RMAN> list backupset of archivelog like '%archive/126.arc'
```

有了这些生成的脚本，就可以在Linux操作系统中使用cron（或其他规划程序）将它们进行规划。Linux上的crontab入口可能类似以下的显示：

```
#min hour date month day command
```

0	1	*	*	0	<path>/b_whole_inc0.sh #Sunday
0	1	*	*	1	<path>/b_whole_inc2.sh #Monday
0	1	*	*	2	<path>/b_whole_inc2.sh #Tuesday
0	1	*	*	3	<path>/b_whole_inc2.sh #Wednesday
0	1	*	*	4	<path>/b_whole_inc1.sh #Thursday
0	1	*	*	5	<path>/b_whole_inc2.sh #Friday
0	1	*	*	6	<path>/b_whole_inc2.sh #Saturday
0	2	*	*	*	<path>/b_archive_2days.sh #Every Day
30	2	*	*	*	<path>/b_archive_delete3.sh #Every Day

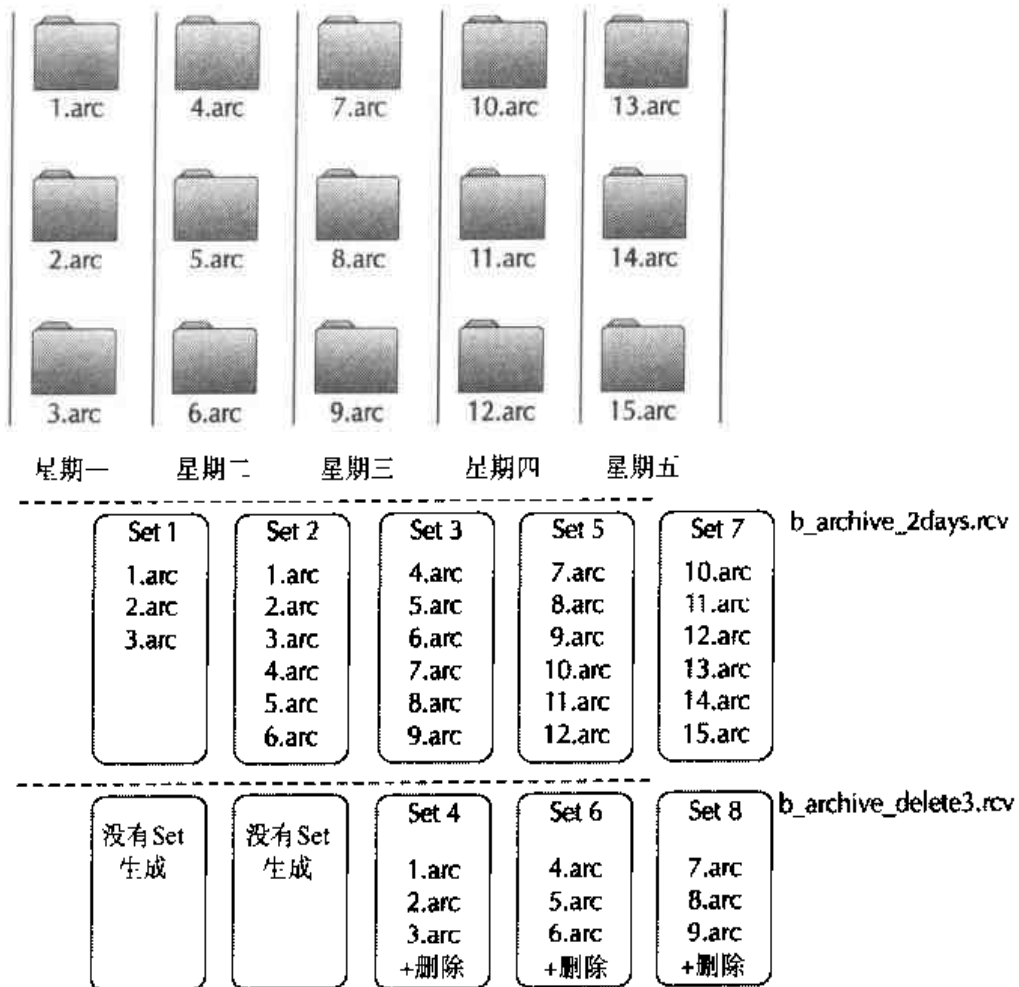


图12-4 归档日志备份时间线

Cron将安排这些脚本在一星期内运行以完成RMAN的备份策略。每个脚本都将用两个参数来调用RMAN: cmdfile和log。同时, 每个脚本也可以调用包含了连接到目标数据库和注册录数据库信息的RMAN脚本。例如, 安排了一个计划来完成整个数据库的增量级别为0的备份, 可能类似b_whole_inc0.sh:

```
$ORACLE_HOME/bin/rman cmdfile=<path>/b_whole_inc0.rcv log=/tmp/b_archive_2days.log
```

在b_whole_inc0.rcv的开始部分增加一个名为connect.rcv的连接脚本:

```
@connect.rcv
```

这个connect.rcv包含了两个连接命令:

```
connect target sys/practice@practice
connect catalog rman817/rman@rcat
```

在调用connect.rcv前面的两个@表示这个脚本将从当前脚本运行的路径下运行。同时，在b_whole_inc0.rcv中增加一个退出语句以便在完成一个安排的工作后退出。这个例子说明了如何在Linux上部署一个RMAN增量备份策略：

在Windows系统下，可以使用AT安排这些RMAN任务：

```
at 1:00 /every:Sunday      cmd /c <path>\b_whole_inc0.bat
at 1:00 /every:Monday      cmd /c <path>\b_whole_inc2.bat
at 1:00 /every:Tuesday     cmd /c <path>\b_whole_inc2.bat
at 1:00 /every:Wednesday  cmd /c <path>\b_whole_inc2.bat
at 1:00 /every:Thursday   cmd /c <path>\b_whole_inc1.bat
at 1:00 /every:Friday      cmd /c <path>\b_whole_inc2.bat
at 1:00 /every:Saturday   cmd /c <path>\b_whole_inc2.bat
at 2:00 cmd /c <path>\b_archive_2days.bat
at 2:30 cmd /c <path>\b_archive_delete3.bat
```

类似于以上显示的Linux脚本，RMAN的备份命令可以从Windows批处理文件中运行。

12.3 RMAN的备份的性能

可以监视RMAN的备份工作的运行并充分利用机器的资源。在对RMAN备份的操作调整之前，首先监视该备份是否按照期望的动作以及在可接受的时间范围运行了。如果备份进行地很迅速则不需要进行调整。为了提高备份的性能，需要在磁带或磁盘设备中平衡，调整RMAN提供的选项。

12.3.1 监视备份与拷贝工作

当进行大型备份工作时，需要了解RMAN是否正在进行备份工作。为了监视备份和拷贝命令的进度，可以在目标数据库上打开另一个SQL*Plus会话并查看进度。首先，可以通过检查V\$PROCESS 和 V\$SESSION数据字典视图来确定RMAN是否连接到了目标数据库上：

```
LINUX> sqlplus sys/practice@practice
SQL> SELECT sid, spid, client_info
2   FROM v$process p, v$session s
3   WHERE p.addr = s.saddr
4   AND client_info LIKE '%id=rman%';
```

应该从这个查询中看到RMAN的连接情况：一个连接是为查询目标数据库，另一个连接是为当前运行的备份分配通道。然后可以查询V\$SESSION_LONGOPS视图来检查备份的进度情况，当RMAN运行时，RMAN将其进度数据发布到这个字典视图中。在备份运行时可以看到% Complete列数值的增加：

```
SQL> SELECT sid, serial#, context, sofar, totalwork,
2   round(sofar/totalwork*100,2) "% Complete"
3   FROM v$session_longops
4   WHERE opname LIKE 'RMAN:%'
```

```
5 AND opname NOT LIKE 'RMAN: aggregate';
```

这两个查询语句在跟踪长时间的RMAN备份时非常方便。

12.3.2 调整性能的技巧

PRACTICE数据库是一个很小的数据库，所以不需要提高其备份的性能。但是在一个实际产品级系统中，对大型或异常繁忙的数据库进行备份时需要仔细考虑。在备份时应充分利用RMAN的并行性并配置通道来获得适当的吞吐率。

- 如果希望备份运行速度快，则应在备份期间多分配几个通道，这样备份或拷贝命令就可以并行执行了。当使用多个通道时必须考虑有多少磁带设备或磁盘可以并行使用。但如果应用了比磁带设备多的通道或者太多的磁盘通道，则会引起I/O子系统的阻塞。
- 如果希望限制一个繁忙的Oracle系统的工作负载，则可以用set limit channel <name> read rate = <integer>来限制通道的读取速度。这将会减少RMAN输入到I/O子系统的读取量。
- 如果希望将一个大型的文件的备份调整到一个限制的文件大小之内，则通过限制通道 <name> Kbytes = <integer>来限制通道创建文件的大小。例如，如果磁盘的操作系统最大文件限制为2GB，则设置为Kbytes = 2097152（即 $2 \times 1024 \times 1024$ ）。
- 在备份期间，同时进行太多的备份将会引起硬件设备上的吞吐瓶颈。这个可以使用sar -d或NT性能检测器等通用的文件检测工具来检测。为了限制同时备份的文件数量，需要限制每个通道同时能打开的文件数，默认值为32，但也可以在备份运行单元中设定限制属性。

在本章中始终谈到，增量备份运行比完全备份要快很多。但是，在备份期间应该注意，不管执行的是什么级别的增量备份，都需要读取每个块以判断是否要进行备份。增量备份只拷贝较少的数据块，因此减少了整体的拷贝时间。

12.4 疑难解答

如果运用RMAN进行备份时遇到了什么问题，可以参考下面的消息和注解。

1) RMAN-04014: startup failed: ORA-01078: failure in processing system parameters 当启动一个远程数据库时，RMAN却使用本地数据库初始化文件来打开该数据库。将参数文件拷贝到RMAN运行的本地硬盘上。在startup命令中增加pfile选项：

```
RMAN> startup pfile=initPRACTICE.ora
```

在Windows NT环境下RMAN是在%ORACLE_HOME%\dbs目录下，而不是%ORACLE_HOME%\database目录下。也可以增加一个用IFILE参数指向pfile的路径的参数文件。

2) ORA-27040: skgfrcre: create error, unable to create file

OSD-04002: unable to open file

O/S-Error: (OS 3) The system cannot find the path specified. 如果为了备份，使用format设置了一个路径但却没有找到，则备份命令将失败并返回该错误代码。请确认format命令确实指向了已经创建好的路径。

3) RMAN-06004: ORACLE error from recovery catalog database:

RMAN-20242: specification does not match any archive log in the recovery catalog 如果在备份期间没有任何归档日志文件与指定的文件匹配, 则返回该错误。忽视该错误, 这个错误在 Oracle 9i 中已经不存在了。

数据块损坏检测

RMAN 提供了三种不同的方法用于在备份期间检测数据块的损坏情况。对于从磁盘读取数据块到缓冲区等这种正常的数据库操作, Oracle 总是通过数据块头部的部分结构与数据块尾部的结构相匹配进行检查。如果没有匹配成功, 则再经过几次读取该块后, 该块就要被报告为损坏了。RMAN 也执行这种数据块的检查。

默认情况下, RMAN 也对每个数据块所包含的校验和 (使用 init.ora 中的参数 DB_BLOCK_CHECKSUM 生成) 进行校验。当备份数据块时保存该校验和的值。如果校验和被重新计算后并没有匹配, 则该块也被报告为损坏。如果数据块没有包含校验和, 则 RMAN 将自行计算一个数值并存储在备份片的该数据块中。当数据块还原时, 这个校验和就要用来检验还原的数据块是否损坏了。在备份命令中指定 NOCHECKSUM 就可以关闭这个特性。但不提倡这样做, 因为当数据块损坏问题发生时也需要进行告警。

可以增加更进一步的数据块检测程序, 称为逻辑数据块检测。在备份命令中指定 CHECK LOGICAL 就可以应用这个特性了。逻辑检测在数据块中确保所有的结构 (行偏离、可用空间、行号等) 都是正确的。默认情况下这种检测都关闭了, 因为进行备份时会增加时间开销。如果要对产品级数据库备份应用, 则事先应测试其影响程度。

默认情况下, 只要检测到一个损坏的数据块 RMAN 就会停止执行。而这个可以在 RMAN 运行单元内 (对所有文件默认都是 0) 通过使用 SET MAXCORRUPT 进行更改。也可以对每个文件设置一个极限值, 但必须谨慎否则会生成许多带有损坏了的数据块的备份文件。当达到这个极限值时, 如果再检测到下一个则会引起备份的终止。当从带有坏块的备份还原数据块时, 则同样的块也将是坏块, 并且当有查询访问到该块时会报告一个错误 (通常是 ORA-1578)。

如果 RMAN 在 MAXCORRUPT 范围之内找到了坏块, 则这些坏块将记录在目标控制文件和恢复目录中。可以在目标数据库的 V\$BACKUP_CORRUPTION 或 V\$COPY_CORRUPTION 视图中看到这些记录。在恢复目录数据库的 BACKUP_CORRUPTION 和 RC_COPY_CORRUPTION 视图中也有同样的记录。这些视图的每一行记录代表了一个备份集和发现损坏块的范围。例如, 如果 RMAN 发现数据块 90、92 和 93 损坏了, 则在 V\$BACKUP_CORRUPTION 视图中会出现如下结果:

RECID	STAMP	SET_STAMP	SET_COUNT	PIECE#	FILE#	BLOCK#	BLOCKS	MAR
3	449013231	449013230	22	1	2	90	1	YES
4	449013231	449013230	22	1	2	92	2	YES

这说明有一个备份集 (#22) 中从数据块 90 开始有一个损坏的块, 从 #92 开始有两个损坏的块。如果检查该目标数据库的 alert.log 则也会发现这三个块的损坏消息。

控制文件中的损坏部分在备份集仍然需要时一直保存着。而恢复目录中的内容则一直保持

到这些备份的设置信息被删除为止。如果损坏的位置是固定的但又进行了新的备份,则V\$BACKUP_CORRUPTION中不会出现新的记录。保留已有的损坏块记录可以检验RMAN是否还原了带有损坏块的数据文件。

12.5 小结

本章中主要介绍了RMAN COPY和BACKUP命令,详细讲解了RMAN提供的每种备份类型的细节(备份、映像拷贝和代理拷贝)以及各种方法的使用原则。用COPY命令生成了数据文件的映像拷贝,用BACKUP命令在磁盘或磁带上生成了包含有数据文件、控制文件或归档日志的文件。还介绍了如何在产品级环境中部署一个简单的增量备份策略。全章中一直使用LIST命令来显示如何确定RMAN的备份是否已经正确地记录在控制文件和恢复目录中。用RMAN进行备份非常简单,当对RMAN的各组件(第11章)和RMAN命令有了清晰的了解之后,就可以指定成功的数据库保护策略并让服务器管理来完成这些工作。如果需要更详细地了解RMAN备份的细节,可以参阅《Oracle 8i Recovery Manager User's Guide and Reference》的第1、2、5章的内容。同时也可以参考Oracle服务器安装时提供的示例脚本。这些脚本保存在\$ORACLE_HOME /rdbms/demo*.rcv目录下。

习题

既然已经完成了这个内容繁多但却非常重要的一章的学习,那么通过下面的几个问题来检验一下学习成果:

1. RMAN的什么命令将生成数据库文件块的拷贝?
 - A. Copy
 - B. Backup
 - C. Allocate
 - D. Run
2. 使用RMAN进行数据文件的映像拷贝时,数据库必须被关闭但要被加载。
 - A. True
 - B. False
3. 如果计划拷贝数据库中所有的数据文件,可以使用BACKUP DATABASE命令。这种备份被称为____备份。
 - A. 完全
 - B. 完成
 - C. 整体
 - D. 增量
4. 如何备份最近两天生成的归档日志文件?
 - A. 备份两天的归档日志
 - B. 从sysdate-2开始备份归档日志
 - C. 备份sysdate-2的归档日志
 - D. 使用RMAN不能完成这个工作

5. 为什么增量备份策略节省备份片的空间?
- A. 只有使用过的和更改过的数据块被拷贝
 - B. 日常增加时进行备份
 - C. 只有部分数据块被拷贝
 - D. 拷贝时对数据块进行了压缩处理

答案

1. B。BACKUP命令读取数据文件、控制文件和归档日志的数据块并拷贝到磁带或磁盘上。生成的文件被称为备份片。备份集是包含了整个备份片的集合。

2. False。可以在数据库打开或关闭时进行数据库文件的映像拷贝。

3. C。整体数据库备份将在备份集中包括所有的数据文件。而完全数据库备份在备份集中只包括所有使用过的数据块。

4. B。应用时间条件，可以指定在备份集中需要的归档日志文件的开始范围。

5. A。RMAN在备份期间只拷贝更改的数据块。对于级别为0的备份或完全备份，则所有使用过的数据块都被拷贝。对于级别为1或大于1的增量备份，只拷贝相对最近备份以来更改过的数据块。RMAN拷贝整个数据块但并不进行压缩处理。



第13章 RMAN目录维护

RMAN目录维护这一章比较重要，因为作为一名数据库管理员，必须能对数据库进行备份以确保数据库始终处于受保护的状态，而不出现任何失效的情况。进行自动备份操作时，需要关注以下几个问题：

- 经常备份的内容是什么？
- 可以完全恢复数据库吗？
- 已经有了一个归档日志文件的良好备份吗？
- 是否有不能恢复的数据库操作妨碍了数据库的恢复？
- 哪些备份是孤立的并可以被删除？
- 恢复需要哪些备份才能实现？
- 可以进行快速恢复吗？

RMAN备份配置完成并成功运行后，仍然需要给予密切关注。作为DBA，必须验证备份工作正顺利地进行着，不再需要的旧备份都被删除。在第12章中，我推荐了一种典型的增量备份策略。当日常自动备份进行时，这些备份一直在累积。图13-1显示了几个星期之后日常备份是如何被建立起来的。这个图还显示了在星期天进行基准增量备份之后，日常数据库的增量备份情况。每天的归档日志也被备份了。如果每个备份操作都完好地按计划进行，则意味着数据库已经得到保护，DBA可以高枕无忧了。

为了精确，需要经常检查备份工作的情况以及数据库是否按计划得到了保护。有时会因为缺少磁盘空间、硬件错误等问题导致自动备份的失败，或者RMAN备份工作很成功但磁带介质管理器却发生了问题。如果在紧急情况下出现备份没有完成或者不可用等情况，那问题就严重了，甚至可能危及到DBA的职位。

本章将描述一些关键命令和概念，并讲述如何通过RMAN目录来验证备份以及删除不再需要的旧备份。

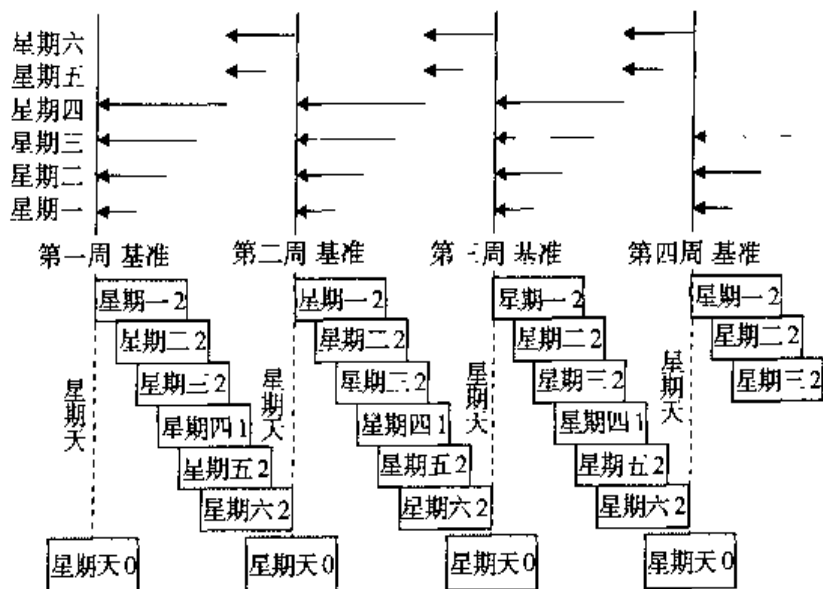


图13-1 PRACTICE数据库的RMAN备份基线

在本章中，会出现许多用于检查和维护PRACTICE数据库目录备份的RMAN命令，因此我将对其中一些命令进行简要介绍。在本章中的练习中，可以通过以下三个方面来探讨一个成功的RMAN部署：

- 1) 如何通过已有的脚本来运行RMAN任务？（参见“恢复管理器脚本”一节。）
- 2) 如何确定数据库已经被RMAN备份保护起来了？（参见“恢复管理器备份确认”一节。）
- 3) 如何清理过时的、不需要的脚本，释放备份存储空间，并保持目录处于一个便于管理的大小？（参见“恢复目录的清理”一节。）

13.1 RMAN的命令

RMAN提供了几个用于维护备份的命令。虽然在第11、12章中已经应用了其中一些命令，但在这里将进行更详细地讲解。下面的各小节包括了对LIST、REPORT、CHANGE、CROSSCHECK和VALIDATE命令的介绍。

13.1.1 LIST

LIST命令将使RMAN读取目录或控制文件来显示备份的有关细节。如果列出一个单一的备份集，将会看到生成的备份片、时间、SCN值、文件名等属于该备份的信息。一般情况下，列表的结果输出到屏幕上，如果在命令行中或RMAN提示符下指定了消息日志文件，那么结果都被输出到该文件中。如果是列出备份的内容，则会看到有关数据库的形态、RMAN备份或者RMAN拷贝的信息。如果需要显示特定备份的细节，则可以限定备份键值、备份标识、时间范围、设备类型或指定的匹配模式。也可以查询备份内部特定的对象、特定的表空间、数据文件、控制文件以及归档日志等的详细情况。

13.1.2 REPORT

REPORT命令对于需要操作的数据库备份都高亮显示。与LIST命令类似，发布的REPORT命令将通知RMAN读取目录或目标控制文件并收集备份的有关信息。REPORT的结果将在屏幕上显示。这些REPORT的信息能确保了解是否可以快速恢复数据库以及找到不再需要的陈旧备份。也可以采用如下方式运行报告（report）命令以获得需要的信息：

- 需要备份什么文件？
- 哪些文件没有最近的备份？
- 哪些文件因为不可恢复的操作的原因而不能被恢复？

Report将会找到一个孤立的备份，该备份已经被执行了多次而且已经被新的备份孤立了。冗余（Redundancy）选项定义了一个备份在被孤立以前能被保留的次数。对孤立备份的报告默认冗余值是一个备份。

13.1.3 CHANGE

CHANGE命令帮助管理已知RMAN备份的状态。对于每个文件，RMAN都分配了一个可维护的通道来访问磁盘或磁带上的文件。CHANGE命令有几个更改目录和备份文件的选项：

- **Delete** 从操作系统（磁盘或磁带）上删除备份片文件，并标记该备份记录为已删除（deleted）。如果使用的是磁带介质管理层（MML），RMAN需要从磁带上删除备份片，但又不期望自己直接进行操作。
- **Available** 为了还原，将映像拷贝或备份片的状态修改为Available。默认情况下，所有的备份都被设置为Available。
- **Unavailable** 为了还原，将映像拷贝或备份片的状态修改为Unavailable。
- **Uncatalog** 从目录中删除备份。

13.1.4 CROSSCHECK

CROSSCHECK命令验证RMAN备份中已知的备份片文件是否保存在了磁盘或磁带上。Crosscheck可以作为CHANGE命令的一个选项执行。当对磁盘上的备份使用CROSSCHECK命令时，RMAN读取目录查找该备份所在的位置。在crosscheck操作期间必须创建一个维护通道。对于每个文件，RMAN使用维护通道确保文件的存在。如果备份文件被找到，RMAN还需要验证备份片头部的正确性。如果对磁带上的备份使用crosscheck命令，RMAN将只查询介质管理器。如果介质管理器返回结果是“该文件不存在”，则RMAN标记该文件为Expired或者Deleted。

13.1.5 VALIDATE

VALIDATE命令确定一个或多个备份集是否能被还原或者是否已经损坏。Validation比crosscheck包含了更多的验证功能。当执行该命令时，RMAN将查找介质上所有的备份片并扫描每个文件。当读取备份片中的每个数据块时，已经存储的备份操作的校验和将被重新计算并进行验证。如果每个文件的校验和都正确，则RMAN确认该备份集的所有备份片都处于有序状态而且有效。备份片验证时的任何错误都将被写入到告警日志文件、服务器跟踪文件、V\$BACKUP_CORRUPTION或V\$COPY_CORRUPTION动态视图中。备份集中可以包含数据文件、控制文件以及归档日志文件。不仅可以对指定的备份集进行验证，而且可以在RESTORE命令中使用Validate选项模拟还原数据库或部分数据库。最后，RMAN可以使用UNTIL参数设定期望的恢复停止点，验证恢复是否可以在前一个时间点完成。

除了这些管理备份与目录的命令以外，还可以使用CREATE、REPLACE以及DELETE等命令存储RMAN备份以及还原资料库中的脚本。在第12章中曾将RMAN命令记录在文件中，然后再运行该文件来执行这些命令。当然也可以将RMAN命令存储在恢复目录中，然后再调用这些存储的脚本来执行一系列RMAN命令。

13.2 恢复管理器脚本

恢复管理器命令可以在RMAN提示符下运行或通过脚本执行。也可以将脚本保存在资料库中，然后在RMAN提示符下执行这些存储的脚本。每个注册的数据库必须有属于自己的存储脚本；脚本不能在不同数据库之间共享。例如，如果有两个数据库在目录中注册并且都要运行一个名为b_whole_inc0的脚本，则必须在目录中分别为每个数据库创建自己的脚本。存储的脚本可以被创建、替换、执行、删除或列表显示。

练习13.1: 存储脚本

在本练习中将要学习如何将前面章节中使用过的脚本文件作为存储脚本保存。

任务描述	时间 (分钟)
1. 创建并替换存储脚本	5
2. 列表并显示存储脚本	5
3. 执行并删除存储脚本	5
总计时间	15

在本练习中, 将要使用第12章中创建的命令文件, 并作为脚本存储在RMAN目录中。然后列出并打印这些脚本, 并执行这些脚本。

任务1: 创建并替换存储脚本

采用以下方式, 用上一章中使用的文件创建备份存储脚本b_whole_inc0.rcv、b_whole_inc1.rcv、b_archive_2days.rcv和b_archive_delete3.rcv:

```
create script b_whole_inc0 {
    allocate channel d1 type disk;
    backup
        incremental level = 0 cumulative
        database
        format '/oradata/PRACTICE/backup/ch13/db0_%d_%s_%p_%t'
        tag = 'WHOLE_INC0';
    release channel d1;
}
create script b_whole_inc1 {
    allocate channel d1 type disk;
    backup
        incremental level = 1 cumulative
        database
        format '/oradata/PRACTICE/backup/ch13/db1_%d_%s_%p_%t'
        tag = 'WHOLE_INC1';
    release channel d1;
}
create script archive_log_current {
    sql "ALTER SYSTEM ARCHIVE LOG CURRENT";
}
create script b_archive_2days {
    allocate channel d1 type disk;
    execute script archive_log_current;
    backup archivelog
        from time 'SYSDATE-2'
        format '/oradata/PRACTICE/backup/ch13/ar_%d_%s_%p_%t';
    release channel d1;
}
create script archive_delete3 {
    allocate channel d1 type disk;
```



```

execute script archive_log_current;
backup archivelog
  from time 'SYSDATE-3' until time 'SYSDATE-2'
  format '/oradata/PRACTICE/backup/ch13/ar_%d_%s_%p_%t'
  delete input;
release channel d1;
}

```

上述脚本定义将完成与第12章中用于备份的脚本文件同样的任务。下面是关于scripts.rcv内容的一些说明：

- 在存储脚本中，不需要运行块。脚本中的命令将在载入时被编译然后作为一个整体执行。
- 存储脚本的语法在保存到目录中时进行检查。这种校验保证了任何存储脚本都没有语法错误。在格式化语句中存储脚本不检查子目录是否存在。
- 存储脚本可以调用其他存储脚本。注意这个存储脚本被命名为archive_log_current。该脚本被其他两个脚本以EXECUTE SCRIPT命令调用。
- 所有RMAN的存储脚本执行完成后，分配的通道并没有自动释放，所以应该显式地释放这些通道。
- 当存储脚本文件运行时，生成脚本的命令第一次生成脚本。也可以使用REPLACE SCRIPT命令以新的脚本代替已经存在的脚本。
- 存储脚本可以从许多文件中或在RMAN提示符下生成；不需要同时载入到一个脚本文件中。
- 如果替换的脚本不存在或需要进行替换，则使用REPLACE SCRIPT命令而不是CREATE SCRIPT来生成这个存储脚本。

为了生成存储脚本，必须连接到恢复目录以及目标数据库。尽管可以在RMAN命令提示符下输入每一行命令来生成存储脚本，但最好还是运行scripts.rcv文件。

```

LINUX> export ORACLE_SID=PRACTICE
LINUX> rman
RMAN> connect target sys/practice;
RMAN> connect catalog rman817/rman@rcat;
RMAN> @scripts.rcv

```

如果只是为了在恢复目录中增加数据，为什么还要连接到目标数据库呢？因为在目录中创建的脚本只用于该目标数据库。（如果还有其他的数据库也注册在该目录中，则当连接到另一个目标数据库时还需要运行scripts.rcv。）当替换脚本文件运行时，这个独立的脚本将进行语法检查并保存储在目录数据库中。脚本文件中的每个存储脚本都有类似以下的信息显示：

```

RMAN-03022: compiling command: replace script
RMAN-03023: executing command: replace script
RMAN-08086: replaced script b_whole_inc0

```

如果输入错误或者RMAN不能编译这些命令，则RMAN将显示错误并说明该文件不符合规则的信息。修改脚本中不正确的内容后再次运行。如果没有错误出现，则该脚本就在目录中生成成了。

任务2：列表并显示存储脚本

如何知道脚本已经在目录中生成了？在RMAN提示符下没有一个比较方便的方法显示所有存储的脚本。有一个名为RC_STORED_SCRIPT的视图，其中每一行分别记录了目录数据库中的每一个存储的脚本。可以用host执行RMAN以外的命令，具体方法如下：

```
RMAN> host;
LINUX> sqlplus \nolog
SQL> connect rman817/rman@rcat
2 FROM rc_stored_script
3 WHERE db_name = 'PRACTICE';
DB_NAME    SCRIPT_NAME
-----
PRACTICE   archive_log_current
PRACTICE   b_whole_inc0
PRACTICE   b_whole_inc1
PRACTICE   b_archive_2days
PRACTICE   b_archive_delete
SQL> exit;
LINUX> exit;
```

退出SQL*Plus，返回到RMAN提示符下。在命令提示符下，再次输入exit，又同退到RMAN提示符下。

可以在RMAN下使用PRINT命令打印存储脚本的内容。下面将显示每一个存储脚本的命令内容：

```
RMAN> print script archive_log_current;
RMAN> print script b_whole_inc0;
RMAN> print script b_whole_inc1;
RMAN> print script b_archive_2days;
RMAN> print script b_archive_delete3;
```

为PRACTICE数据库创建的脚本将显示在屏幕上。因为刚刚创建了这些脚本，所以以这种方式在屏幕上打印并没有什么意义。也可以查询RC_STORED_SCRIPT_LINE目录视图来显示脚本的内容。

任务3：执行并删除存储脚本

RMAN EXECUTE SCRIPT命令在RMAN提示符下运行存储脚本。尝试运行刚才生成的一个脚本来执行一次完全数据库备份。运行存储在目录数据库中的b_whole_inc0脚本来备份PRACITCE数据库：

```
RMAN> run { execute script b_whole_inc0; }
```

脚本执行命令必须被置于运行块中。如果运行几个不同的脚本，则需要置于单独的运行块中。为了说明，进行一次整体数据库增量级别1的备份以及最近两天之内生成的归档日志文件的备份。

```
RMAN> run {
2> execute script b_whole_inc1;
3> execute script b_archive_2days;
4> }
```

为了删除存储脚本，使用REMOVE SCRIPT命令并指定脚本的名称：

```
RMAN> delete script b_whole_incl;
```

在本练习中，只创建了少量的脚本。对于大型的部署规划，将会创建诸如备份、维护以及恢复等多种任务的很多复杂脚本，关于这些脚本的一个优秀示例可以在\$ORACLE_HOME/rdbms/demo目录下找到，几个以deom开头而以.rcv结尾的文件包括了Oracle提供的示例脚本，打开这些文件并修改格式等语句以适应自己的环境。从存储脚本运行备份操作的一个最大优点就是集中存放。分离的脚本文件不需要在服务器上到处维护，而RMAN也可以运行在没有驻留目标数据库的服务器上。使用存储脚本意味着不需要在其他机器上拥有同样的拷贝。

13.3 恢复管理器备份确认

数据库得到保护是否就可以防止失效？是否可以使用服务器管理的备份按商业要求进行数据库的恢复？这些问题对于成功的RMAN部署是一个关键的测试。我们知道，必须测试并验证这些备份以确保可以用于还原与恢复Oracle数据库。这些测试对于大型的数据库需要花费大量时间和磁盘空间。RMAN中是否有一种方法可以验证目标数据库已经被当前的备份保护起来了？答案是，有。是否可以发现数据库的某些部分由于不可恢复的操作而没有受到保护？答案是，可以。是否可以确认目录中注册的备份仍然保存在磁盘或磁带上？答案是，最好相信它！

练习13.2：确认备份

通过RMAN和SQL*Plus有许多方法可以对备份进行确认。为了完成这些工作可能需要用到LIST、CHANGE、REPORT和VALIDATE命令。也可以查询目标控制文件和目录来检查当前备份的详细清单。通过这些方法，将可以确认对PRACTICE数据库已经进行了充分的备份。

任务描述	时间（分钟）
1. 用LIST确认备份	10
2. 用REPORT确认备份	10
3. 用SQL*Plus确认备份	10
4. 交叉检查备份	10
5. 验证备份	10
总计时间	50

这个练习主要提供了几种用来验证使用已有RMAN备份可以恢复数据库的方法。

任务1：用LIST确认备份

在这个任务中，将介绍可以列出的信息类别和数据库文件以及如何过滤不需要的内容。在介绍如何用LIST命令来确认备份之前，先通过示例来讲解LIST命令的各个选项。

1) 种类 LIST命令将显示RMAN相关备份信息的三种类别中的一种。可以查询RMAN的数据库形态（Incarnation）信息、备份（backup）以及映像拷贝（image copy）：

- Incarnation 显示数据库的形态，包括重置日志的SCN和日期/时间值。每个数据库形态的数据库键值以及形态键值都不相同。
- Backup 显示数据库文件的备份，包括备份集号、备份级别（增量号或完全备份）以

及完成时间。每个备份集都有自己的键值。默认情况下，备份列表显示备份集和数据文件，而不显示归档日志和控制文件。

- **Copy** 显示数据库文件的映像拷贝，包括检查点SCN、时间、文件名以及该拷贝完成的日期/时间值。每个映像拷贝的键值都不相同。默认情况下，该拷贝列表显示数据文件的映像拷贝，而不显示归档日志和控制文件。

2) **数据库文件及过滤器** 当列表备份与拷贝时，应指定RMAN需要显示的数据库文件类型：数据文件、控制文件或者归档日志文件。如果没有指定显示的文件类型，默认情况下备份与拷贝的列表将显示数据文件。对于备份与拷贝信息，可以设置需要显示的备份发生时的条件范围（过滤器）。为了查看不同的数据库对象列表（数据文件、表空间、控制文件、归档日志文件或者数据库），应在LIST BACKUP或COPY命令上增加以下这些关键字：

- **database** 显示目标数据库所有数据文件的备份或映像拷贝。
- **Tablespace** 在一个单引号字符串中显示表空间的所有数据文件的备份或映像拷贝。可以列表多个数据文件。
- **Datafile** 在一个单引号的字符串内以文件号或文件名形式显示数据文件的备份或映像拷贝。可以列表多个数据文件。
- **Controlfile** 显示目标数据库中的控制文件的备份或映像拷贝。
- **Archivelog** 显示归档日志文件的备份或映像拷贝。

3) **数据文件** 为了以不同的方式说明SYSTEM和USERS表空间数据文件的备份与映像拷贝的清单，在RMAN提示符下输入以下命令：

```
# List backup sets & pieces for the SYSTEM and USERS tablespaces
RMAN> list backup;
RMAN> list backup of database;
RMAN> list backup of tablespace 'SYSTEM', 'USERS';
RMAN> list backup of datafile 1, 3, 7;
RMAN> list backup of datafile
2>      '/oradata/PRACTICE/system01.dbf',
3>      '/oradata/PRACTICE/users01.dbf',
4>      '/oradata/PRACTICE/users02.dbf';
# List image copies for the SYSTEM and USERS tablespaces
RMAN> list copy;
RMAN> list copy of database;
RMAN> list copy of tablespace 'SYSTEM', 'USERS';
RMAN> list copy of datafile 1, 3, 7;
RMAN> list copy of datafile
2>      '/oradata/PRACTICE/system01.dbf',
3>      '/oradata/PRACTICE/users01.dbf',
4>      '/oradata/PRACTICE/users02.dbf';
```

当列表该数据库的备份与拷贝时，将得到组成该数据库所有部分的全部备份与映像拷贝。

4) **控制文件** 为了查看控制文件的备份与映像拷贝，可以执行只输出控制文件的list命令：

```
RMAN> list backup of controlfile;
```

```
RMAN> list copy of controlfile;
```

注意，这个备份列表的输出包括了字符串“Controlfile included”，这说明该控制文件包含在SYSTEM表空间的备份中。这个列表的输出与数据文件列表的输出不同。

5) 归档日志 对归档日志列表的过滤选项与数据文件和控制文件都不同。必须在list命令中明确指定需要哪个归档日志文件的清单。使用关键字Archivelog将指定一个归档日志文件备份或映像拷贝的列表。列表的结果可以是所有的归档日志文件（ALL）、匹配一种归档日志模式（LIKE）或者指定文件范围的列表。范围类型必须属于以下范畴之一：

- 时间 可以用FROM TIME、UNTILE TIME或者二者结合的关键字指定范围，后跟单引号的日期字符串。日期字符串可以是字符形式、与TO_DATE函数结合的字符串形式、SYSDATE的日期算式和其他类似LAST_DAY、ADD_MONTHS等的日期函数。FROM TIME将会包括从指定时间开始以来进行重做更改的归档重做日志文件。UNTILE TIME则包括从指定时间作为最后的重做更改时间以来的归档重做日志文件。
- SCN SCN跨越了重做日志文件，可以用关键字FROM SCN、UNTIL SCN或二者结合的关键字来指定范围，后跟一个整数值。FROM SCN将会包括具有大于或等于SCN值的归档重做日志文件。UNTIL SCN将会包括具有小于或等于该SCN值的归档重做日志文件。
- 日志序号 重做日志序列的值是针对重做日志文件的。可以用关键字FROM LOGSEQ、UNTIL LOGSEQ或二者结合的关键字来指定范围，后跟一个整数值。FROM LOGSEQ将会包括具有大于或等于该日志序列号的归档日志文件。UNTIL LOGSEQ将会包括具有小于或等于该日志序列号的归档日志文件。

为了解释如何列表归档日志备份与拷贝，则请参看如下示例：

```
# List archive log backup sets & pieces
RMAN> list backup of archivelog all;
RMAN> list backup of archivelog like '%oradata%';
RMAN> list backup of archivelog from time 'SYSDATE - 14';
RMAN> list backup of archivelog until time 'SYSDATE - 7';
RMAN> list backup of archivelog
2>   from time 'SYSDATE - 14'
3>   until time 'SYSDATE - 7';
RMAN> list backup of archivelog from SCN 100000;
RMAN> list backup of archivelog until SCN 110000;
RMAN> list backup of archivelog from SCN 100000 until SCN 110000;
RMAN> list backup of archivelog from logseq 130;
RMAN> list backup of archivelog until logseq 139;
RMAN> list backup of archivelog from logseq 130 until logseq 139;
```

上面的示例说明了如何查看RMAN的备份以及RMAN的映像拷贝的归档日志文件。罗列的几个示例有助于理解如何定义范围模式。当连接到PRACTICE目标数据库以及目录数据库时，可以运行这些命令。调整SCN值以及日志序列号则可以看到不同的输出结果。为了查看映像拷贝归档日志文件，请用关键字COPY代替BACKUP。

注意 为LIST命令指定数据库对象是互斥的。例如，不能在一个列表中同时观察数据文件和控制文件。可以显示数据文件、控制文件或者归档日志之一的备份。

6) **日期格式选项** 当用RMAN指定一个日期时，就会得到不同的选项。请参看表13-1中的日期示例，说明了如何在列表（或者其他使用日期参数的RMAN命令）中用RMAN建立日期值。

表13-1 RMAN日期字符串选项

日期字符串	值
SYSDATE	当前的日期时间
TRUNC (SYSDATE)	当天午夜
SYSDATE	比当前日期时间少7天
'13-JAN-2002'	如果会话的值是NLS_DATE_FORMAT = 'DD-MON-YYYY'，则是2002年1月13日午夜
TO_DATE ('13-JAN-2002 13:00', 'DD-MON-YYYY HH24:MI')	不论NLS_DATE_FORMAT的值是什么，该时间是2002年1月13日下午1时
ADD_MONTHS(SYSDATE, 1)	当前日期时间之后的一个月
ADD_MONTHS(SYSDATE, -1)	当前日期时间之前的一个月
LAST_DAY(SYSDATE, 1)	当前日期时间前一天
NEXT_DAY('SYSDATE', 'SUNDAY')	当前日期时间的下一个星期天的日期

7) **更多的过滤器选项** 既然已经建立了需要显示的列表的数据库对象，可以基于几个条件来过滤输出。如果使用了RMAN一段时间但没有删除孤立的备份，则输出结果会超额。这个列表可以使用过滤规则得到解决。过滤规则包括：

- **Completion time** 列表输出的结果中只出现被限制在时间范围内的备份与映像拷贝。用AFTER关键字指定时间下限，用BEFORE关键字指定时间上限。用BETWEEN关键字指定时间范围。可以提供类似表13-1限制的时间规划。
- **Tag** 列表输出结果中只出现被限制在特定的tag名称的备份与映像拷贝。如果没有提供tag名称，则将包括所有被标识的备份与映像拷贝。
- **Recoverable** 列表的结果只限于那些可以被恢复的数据库。也可以有选择地为可恢复性指定停止点。这个选项将显示所有适于基准备份的备份以及适于恢复的正确形态号。例如，如果进行了一次级别为3的增量备份而没有基准（增量级别为0、完全备份或映像拷贝）备份，则该备份不能用于恢复。
- **Device type** 列表的结果只限于那些具有磁盘或'SBT_TAPE'类型的备份或映像拷贝。
- **Like** 列表的结果只限于那些含有的数据文件匹配了文件名模式的映像拷贝。 '%' 通配符可以匹配任何字符序列而 '_' 字符则只匹配一个字符。

这些列表的过滤器可以与其他过滤规则一起使用，方法如下所示：

```
# List backup sets of the system datafile with filter criteria
RMAN> list backup completed after 'SYSDATE - 14';
RMAN> list backup completed before 'SYSDATE - 7';
```

```

RMAN> list backup completed between 'SYSDATE - 14' and 'SYSDATE - 7';
RMAN> list backup tag = WHOLE_INC0;
RMAN> list backup recoverable;
RMAN> list backup device type disk;
RMAN> list copy like = '%system%'; #image copies only for like filter
RMAN> # Combine filters
RMAN> list backup completed after 'SYSDATE - 14'
2>          tag = WHOLE_INC0
3>          device type disk
4>          recoverable;

```

这些示例说明了如何使用过滤规则来限制列表的输出。为了查看映像拷贝的清单，可以用COPY代替关键字BACKUP。也可以将这些过滤器与其他数据库对象的选项组合使用，比如为了查看RBS表空间本月完成的、tag名为“WHOLE_INC0”的磁盘备份，可以按如下方式操作：

```

RMAN> list backup of tablespace 'RBS'
2>      completed after 'LAST_DAY(ADD_MONTHS(SYSDATE,-1))+1'
3>      TAG = WHOLE_INC0
4>      bdevice type disk;

```

如果该备份存在，则该list命令会列出所需要的结果。注意，过滤规则的顺序是无关的，而且每个过滤条件不需要在同一行书写。

8) 使列表工作 当以这些选项列出了备份与映像拷贝后，又如何确认PRACTICE数据库已经得到了保护？根据生成的列表可以确认得到的备份与期望的策略一致。考虑一个认同了第12章中描述的简单策略的备份场景。这个事情要求可以恢复最近21天中任何时刻的数据库。可以采用如下描述的增量备份策略：

- 星期日 累积增量级别为0的备份，tag为SUN_LEVEL0
- 星期一 累积增量级别为2的备份，tag为MON_LEVEL2
- 星期二 累积增量级别为2的备份，tag为TUE_LEVEL2
- 星期三 累积增量级别为2的备份，tag为WED_LEVEL2
- 星期四 累积增量级别为1的备份，tag为THU_LEVEL2
- 星期五 累积增量级别为2的备份，tag为FRI_LEVEL2
- 星期六 累积增量级别为2的备份，tag为SAT_LEVEL2
- 每天 备份所有最近两天的归档日志
- 每天 备份所有超过三天的归档日志并从磁盘上删除
- 每天 将当前的控制文件拷贝到每个整体数据库增量备份中。

采用这种备份策略，每个数据库文件中的备份输出是什么？每个归档日志是什么？每个控制文件呢？假设超过21天的备份都被删除了，下面这个列表将显示期望看到的输出结果。

```

RMAN> # One backup set each day, total of 21
RMAN> list backup completed after 'SYSDATE - 21';
RMAN> # 21 backup sets completed in the last 21 days
RMAN> list backup of controlfile completed after 'SYSDATE - 21';

```

```

RMAN> # 3 backup sets completed on Sunday
RMAN> list backup completed after 'SYSDATE - 21' tag = SUN_LEVEL0;
RMAN> # Repeat above for other tag values
RMAN> # 42 backup sets - two backups daily for 21 days
RMAN> list backup of archivelog all completed after 'SYSDATE - 21';
RMAN> # For a specific archive log, should have 3 backups
RMAN> list backup of archivelog from logseq 130 until logseq 130
2>    completed after 'SYSDATE - 21';
RMAN> # Make sure we can recover to 21 days previous
RMAN> list backup recoverable until time 'SYSDATE - 21';

```

这些命令的列表输出结果很长，因此，使用RMAN的log命令来运行。然后再检查日志文件的输出。查看列表输出文件中是否有期望获得的备份清单。

```

LINUX> rman log=/tmp/list.out
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @list.rcv
RMAN> exit
LINUX> vi /tmp/list.out

```

任务2：用REPORT确认备份

LIST命令显示得到了什么内容，而REPORT则显示需要什么。有四个report的选项，其中两个（Need backup以及Unrecoverable）可方便地用于确认当前的备份是否可以恢复。

- **Need backup** 说明要达到完整地备份，数据文件还需要适合某些条件（日期、增量级别或冗余数）的新备份。恢复将需要归档重做日志和增量备份。根据这个报告可以确定数据库中的数据文件是否还需要更多的增量备份或其他重做的时间。同时也可以报告哪些数据文件没有足够的冗余备份。
- **Unrecoverable** 说明需要备份的数据文件中的数据块发生了不可恢复的操作（即NOLOGGING操作）。
- **Obsolete** 说明不再需要的备份可以被删除。对于该报告可以指定额外的备份数（冗余备份）。例如，如果觉得两个冗余备份比大多数情况下的一个冗余备份更合适，则可以报告所有比这两个冗余备份还要早的备份。
- **Schema** 显示目标数据库的数据库物理模式。使用目录时，可以在指定的条件（时间、日志序列或SCN）下观察物理模式。

因为report具有分析功能，所以用于特殊report的参数属于REPORT生成类型。在本练习中，将查看每个report并讨论它们如何在备份策略中发现漏洞。

1) **Need Backup报告** Need Backup选项显示了一定日期范围内还没有备份的数据库的内容。

```

RMAN> report need backup days 2 database;
RMAN-03022: compiling command: report
Report of files whose recovery needs more than 2 days of archived logs
File Days Name
-----
1      2    /oradata/PRACTICE/system01.dbf

```



```

2      2      /oradata/PRACTICE/rbs01.dbf
3      2      /oradata/PRACTICE/users01.dbf
4      2      /oradata/PRACTICE/temp01.dbf
5      2      /oradata/PRACTICE/tools01.dbf
6      2      /oradata/PRACTICE/indx01.dbf
7      2      /oradata/PRACTICE/users02.dbf

```

这个报告显示了在两天之内还没有备份的数据文件，并说明完全恢复将需要所有最近两天的重做日志的应用。这个报告显示了最早的数据文件还没有备份。即使今天除了一个数据文件以外所有的数据文件都备份了，而那个文件最近两天之内没有备份，则该数据文件将出现在报告中。

完全恢复需要很长时间的另一个原因就是，需要应用太多的增量备份。假设需要还原20个增量备份，尤其是从磁带上，将要花费大量时间。Need backup报告中的incremental选项将显示用于完全恢复必须应用多少增量备份。如果目标数据库的恢复需要超过三个增量备份则该报告将发出如下警告：

```

RMAN> report need backup incremental 3 database;

```

警告 Need backup选项报告中不显示从来没有被备份过的数据文件。所以当Need backup没有任何结果时不要被蒙蔽了；也许根本就没有任何备份。为了实现完全备份与恢复策略，必须确保数据库中的所有数据文件都在增量备份开始之前进行了至少一次备份。确保数据库的所有部分都得到备份是DBA的责任。

2) Unrecoverable报告 对目标数据库进行Unrecoverable报告将会验证被操作更改的数据块是否可以应用重做记录。如果需要另一个增量备份以防止对一个NOLOGING操作应用重做时出现问题，RMAN将会给出报告：

```

RMAN> report unrecoverable;

```

如果这个报告有结果显示，则应该进行另一次增量（或完全）备份来拷贝不能被重做应用恢复的数据库数据块。

3) Schema报告 Schema报告对于长时间恢复或不可恢复数据块并不给出警告。可以比较目标数据库与当前数据文件或查看模式是否一致：

```

RMAN> report schema;
RMAN> report schema at time 'SYSDATE-7';
RMAN> report schema at time "TO_DATE('01/13/2002','MM/DD/YYYY')";
RMAN> report schema at scn 1000;
RMAN> report schema at logseq 131;

```

任务3：用SQL*Plus确认备份

对目录用户增加SQL命令产生的结果类似于LIST命令的结果。使用SQL来查询目录，查看描述了备份集、备份片以及组成当前备份的数据文件的目录内容。也可以使用备份集标识或备份集键值来查找所需要的信息。查询也可以针对目标控制文件或目录用户表。查询这些视图的最大优点是：可以控制输出的表现形式。列表具有固定的输出格式，而SQL可以有多种格式化的方法。表13-2中显示了包含备份与映像拷贝信息的关键视图。

表13-2 RMAN备份与拷贝的重要数据字典与目录视图

目标数据库视图	目录视图	描述
V\$BACKUP_SET	RC_BACKUP_SET	RMAN备份集信息
V\$BACKUP_PIECE	RC_BACKUP_PIECE	RMAN备份片文件信息
V\$BACKUP_DATAFILE	RC_BACKUP_DATAFILE	RMAN备份文件信息
V\$DATAFILE_COPY	RC_DATAFILE_COPY	RMAN映像拷贝文件
V\$BACKUP_CONTROLFILE	RC_BACKUP_CONTROLFILE	RMAN备份控制文件信息
V\$CONTROLFILE_COPY	RC_CONTROLFILE_COPY	RMAN控制文件的映像拷贝
V\$BACKUP_REDOLOG	RC_BACKUP_REDOLOG	RMAN备份归档日志信息

控制文件视图中的列和数据与目录视图中的非常相似，但也有以下一些差别：

- 目录视图有一个列的内容是标记数据库和数据库的形态，而目标数据字典视图没有。控制文件将只含有当前形态的编码。
- 目录视图含有一个备份键值，是作为备份与拷贝的内部RMAN目录索引号。控制文件视图中则没有这个键值。
- 目标控制文件与目录可能含有对方没有的记录行。如果超过时限，则目标控制文件中的备份记录将不再出现在v\$视图中。如果从目录中删除了备份，则这些备份不会出现在目录视图中，但仍然保留在目标控制文件中。

1) 备份集与备份片 可以在目标数据库和目录数据库中看到RMAN进行的备份集。可以查询这些视图如下，显示备份集与备份片的信息：

```
SQL> connect sys/practice@practice
SQL> SELECT recid, set_count,
2         decode(backup_type, 'D', 'FULL',
3                        'I', 'INCREMENTAL',
4                        'L', 'LOGS') Type,
5         incremental_level ILevel, completion_time Completed
6         FROM v$backup_set;
SQL> connect rman817/rman@rcat
SQL> SELECT recid, set_count,
2         decode(backup_type, 'D', 'FULL',
3                        'I', 'INCREMENTAL',
4                        'L', 'LOGS') Type,
5         incremental_level ILevel, completion_time Completed
6         FROM rc_backup_set
7         WHERE db_key =
8         (SELECT max(db_key) FROM rc_database
9         WHERE name = 'PRACTICE');
```

为了目录用户与目标控制文件比较备份片，可以分别对V\$BACKUP_PIECE和RC_BACKUP_PIECE视图执行类似的查询。这些视图包含了重要的列，如handle（磁盘或磁带的备份片文件名）和tag（备份片的标识名称）。

2) 数据文件 除了RMAN LIST命令以外，还可以对目录中的视图进行查询以获得数据文件拷贝和备份的详细情况。运行下面的查询就可以得到与以上RMAN LIST COPY和LIST BACKUP

命令类似的信息：

```
SQL> connect system/practice@practice
SQL> SELECT file#, completion_time time,
2      checkpoint_change# change#, name
3      FROM v$datafile_copy;
SQL> SELECT file#, completion_time time,
2      checkpoint_change# change#, set_count
3      FROM v$backup_datafile;
SQL> connect rman817/rman@rcat
SQL> SELECT file#, completion_time time, checkpoint_change# change#, name,
2      cdf_key key,
3      decode (Status, 'A','AVAIL','U','UNAVAIL','D','DELETED') Status
4      FROM rc_datafile_copy
5      WHERE db_name = 'PRACTICE';
SQL> SELECT file#, completion_time time, checkpoint_change# change#,
2      bdf_key key, set_count,
3      decode(status, 'A','AVAIL','O','UNUSABLE','D','DELETED') Status
4      FROM rc_backup_datafile
5      WHERE db_name = 'PRACTICE';
```

3) 备份控制文件 在V\$CONTROLFILE_COPY、V\$BACKUP_CONTROLFILE、RC_CONTROLFILE_COPY和RC_BACKUP_CONTROLFILE视图中可以找到控制文件映像拷贝和备份。这些目录视图提供了比LIST命令更多的关于控制文件备份的信息。在SQL*Plus中使用SQL语句，可以查看其他的列并可以格式化输出的结果。

4) 备份归档日志 除了LIST BACKUP ARCHIVELOG命令外，还可以向目录用户查询以显示指定备份的信息。以目录拥有者的身份启动一个SQL*Plus会话并运行类似上一个任务中的查询语句（从V\$BACKUP_REDOLOG中查询）。

为了方便SQL语句检查所有备份的归档日志文件以及每个文件中的出现次数，可以用以下的SQL来查询恢复目录：

```
SQL> connect system/practice@practice
SQL> SELECT sequence#, count(*)
2      FROM v$backup_redolog
3      GROUP BY sequence#;
SQL> connect rman817/rman@rcat
SQL> SELECT sequence#, count(*)
2      FROM rc_backup_redolog
3      WHERE db_name = 'PRACTICE'
3      AND status = 'A'
4      GROUP BY sequence#;
```

因为采用了第12章中描述的归档日志备份策略，所以应该在每个归档日志文件中看到三个备份。如果序列号中有中断，则表示那个归档日志文件就没有RMAN备份。

注意 这里的查询语句比较简单。一旦熟悉了这些视图，就可以按照自己的RMAN部署来定制这些查询。同时，如果在目录中有多个数据库和数据库的形态注册，则应该使用

db_key和dbinc_key两个列作为条件，确保查询的语句中的where部分包括了正确的数据库和形态。

任务4：交叉检查（crosscheck）备份

了解备份信息的关键是确定当前的备份情况是否已经处于有序状态。当使用list和report时，只能得知RMAN对于当前备份的了解程度。如果在操作系统下删除了备份片，那么控制文件和目录将都不知道这个操作。因此应经常进行交叉检查以确认目录包含了有关备份片的正确信息。应注意的是交叉检查是对资源敏感的操作，尤其是备份都在磁带设备上时。当进行交叉检查时，RMAN确定备份集是否存在。如果需要在磁带上交叉检查大量的备份，将会使磁带子系统始终处于繁忙、加载、卸载和磁带定位等状态下。一些MML查询语句将查询自己的目录并向RMAN报告，而对于为了检查文件是否存在于物理磁带上并不敏感。因为PRACTICE数据库备份都保存在磁盘上，而且没有太多的备份，所以交叉检查操作会很快完成。

注意 维护（maintenance）通道并不需要类似于备份、还原、恢复通道那样的名称。

下面的命令将检查目标控制文件和目录已知的所有备份：

```
RMAN> allocate channel for maintenance type disk;
RMAN> crosscheck backup;
RMAN> release channel;
```

输出结果如下：

```
RMAN-08074: crosschecked backup piece: found to be 'AVAILABLE'
RMAN-08517: backup piece
handle=/oradata/PRACTICE/backup/cn13/db_PRACTICE_130_1_446974818 recid=19
stamp=446974822
```

单词“available”意思是该备份片被找到。可以用tag选项限制对带有tag或特殊的tag名称的备份的交叉检查（crosscheck）。最后，可以定义备份所属的数据库对象。

警告 在具有许多RMAN磁带备份的产品级系统上必须谨慎地使用交叉检查（crosscheck）命令，否则就会占用磁带子系统、MML库并阻碍备份的运行。

```
RMAN> crosscheck backup of datafile 1;
RMAN> crosscheck backup completed between 'SYSDATE-7' and 'SYSDATE';
RMAN> crosscheck backup completed between '01-JAN-2002' and '30-JAN-2002';
RMAN> crosscheck backup tag WHOLE_INC0;
RMAN> crosscheck backup of archivelog like '%/100.%';
RMAN> crosscheck backup of archivelog from logseq 100 until logseq 110;
RMAN> crosscheck backup of archivelog from 'SYSDATE-7' and 'SYSDATE';
```

为了交叉检查（crosscheck）特定的备份集或映像拷贝，可以在change命令中使用crosscheck选项，而不仅仅使用crosscheck命令：

```
RMAN> change backupset 311 crosscheck;
RMAN> change datafilecopy 545 crosscheck;
```

如果找到备份集，则结果如下：

```
RMAN-08074: crosschecked backup piece: found to be 'AVAILABLE'
RMAN-03022: compiling command: change
```

```

RMAN-06154: validation succeeded for datafile copy
RMAN-08513: datafile copy
filename=/oradata/PRACTICE/backup/ch12/system01.dbf.bak recid=1
stamp=446972315

```

最希望看到的信息就是VALIDATION SUCCEEDED。

任务5：验证备份

RMAN正是因为可以在restore命令中提供了validate选项才具有了模拟还原的能力。这种验证的方法使RMAN可以选择用于还原的备份集或拷贝。RMAN也可以使用VALIDATE BACKUPSET命令来验证指定备份集的还原性。这两种还原验证方法都可以对整个数据库、表空间、数据文件、控制文件或归档日志文件应用。

警告 RMAN的验证并不测试还原（应用增量备份和重做），因此，验证并不是一个综合的测试。可以验证还原，然后只验证增量备份的还原。

1) 还原验证 首先验证整个数据库的还原并让RMAN选择适用的备份集和拷贝。这种验证方法在一个运行块中使用了还原（restore）命令。验证备份集和拷贝的还原。

下面的示例验证了磁带上的备份控制文件、所有表空间和归档重做日志的还原操作：

```

RMAN> run {
    allocate channel d1 type disk;
    restore database validate;
    restore archivelog all validate;
}
RMAN-08096: channel d1: starting validation of datafile backupset
RMAN-08502: set_count=40 set_stamp=447343243 creation_time=01-DEC-01
RMAN-08023: channel d1: restored backup piece 1
RMAN-08511: piece handle=/oradata/backup/ch13/DB_PRACTICE_40_1_447343 243
tag=WHOLE_INC0 params=NULL
RMAN-08098: channel d1: validation complete
RMAN-08096: channel d1: starting validation of datafile backupset
RMAN-08502: set_count=41 set_stamp=447343280 creation_time=01-DEC-01
RMAN-08023: channel d1: restored backup piece 1
RMAN-08511: piece handle=/oradata/backup/ch13/DB_PRACTICE_41_1_447343 280
tag=WHOLE_INC1 params=NULL
RMAN-08098: channel d1: validation complete

```

如果输出信息说明验证完成了，则说明验证成功；否则将会显示错误的消息。

2) 验证备份集 第二种验证方法需要指定备份集的细节。对PRACTICE数据库进行两个增量备份。例如，备份集130是级别为0的增量备份，而备份集131的级别为1。这两个增量备份都包含一个控制文件的拷贝：

```

RMAN> run {
    allocate channel d1 type disk;
    validate backupset 130, 131;
}

```

验证期间，将读取备份片并模拟还原操作。如果出现：

RMAN-08024: channel chl: restore complete, 则说明还原可以工作, 否则将显示错误消息。

为了模拟失败的验证, 将备份片重命名为一个新的文件名, 并验证备份集, 这时将得到一个错误。然后, 将任何字处理程序文档命名为备份片的原始文件名。再次验证备份集, 又将得到其他的错误。删除刚才拷贝的用于模拟损坏的备份片的字处理的文件, 并将原始的备份片恢复到初始的名称。最终验证将产生一个成功的消息。

13.4 恢复目录的清理

假设每天进行一次RMAN数据库备份。一年之后, 将得到365份数据库备份以及365份归档日志备份。除非有计划要恢复到364天以前的时间点, 否则根本不需要一年以前的备份。如何删除这些陈旧的备份? 答案是, 应用change命令。

练习13.3: 清除冗余的备份

在本练习中, 将会介绍删除不再需要的陈旧备份的3种方法。

任务描述	时间 (分钟)
1. 查找孤立的备份	5
2. 删除孤立的备份	5
3. 删除陈旧的备份	5
4. 查找过期的备份	5
5. 删除过期的备份	5
总计时间	25

为了清理陈旧的备份, RMAN提供了带有delete选项的change命令。使用report、交叉检查 (crosscheck) 和SQL可以首先确定当前恢复不需要的孤立备份。一旦确定后, 就删除这些孤立备份。

任务1: 查找孤立备份

在完全恢复期间, RMAN将使用最近的备份或映像拷贝来还原数据库文件。当需要进行时间点恢复时, RMAN还原离时间点最近的备份。因为有最近的备份而恢复不再使用的其他备份就是孤立备份。如果选择的恢复时间比最近的备份要早, 则可能会需要这些独立备份。因此拥有少量的额外备份也是有道理的。然而, 如果积累了太多的备份, 则应该选择部分备份进行删除。

RMAN有一个报告可以显示超过了冗余范围的孤立备份集。或许公司更合适三个星期的备份保持范围。这样就没有人会让人DBA将数据库恢复到三个星期以前的状态。

查找已经孤立的以及21天 (三个星期) 前生成的备份:

```
RMAN> report obsolete redundancy 1
2>      until time 'SYSDATE-21' device type disk;
```

该报告返回的任何备份或拷贝都可以被删除; 不再需要这些备份或拷贝。因为默认的冗余值为“1”, 所以可以忽略冗余部分。

```
RMAN> report obsolete until time 'SYSDATE-21';
```

这个命令将只保留21天以来进行的一个备份拷贝。一旦发生了21天前的备份是增量级别为1的备份的情况，为了使用这些备份，将需要22天前进行的增量级别为0的备份，而这个备份有可能已经删除了。这样将危害这种21天的备份持续策略，因为不能将数据库恢复到21天前的状态。

如果这个问题没有产生什么麻烦，并且只能在磁带或磁盘上保留21天的备份（不管具体内容），则都可以使用list命令来显示21天前的备份列表：

```
RMAN> list backupset complete before 'SYSDATE-21';
```

警告 如果在某个时刻进行完全或增量级别为0的备份，并创建了所有的增量备份或者归档日志文件，则只能恢复到这个时刻。因此，如果不确定是否还要恢复到过去某个时间点，则应保留一个旧版本和所有的归档日志。应根据业务的需求决定备份的数量。如果磁带循环使用的周期是三个月，则只能将备份恢复到三个月以前。

任务2：删除孤立的备份

一旦确定了要删除的备份集，如何使用RMAN来删除这些备份？可以通过分配删除通道并使用change命令来删除备份集或映像拷贝。例如，如果REPORT命令确定需要删除三个备份集（131、132和133），则可以使用以下的命令从操作系统中删除这些备份片，并将其在目录中的状态修改为已删除：

```
RMAN> # Report obsolete backups on disk.
RMAN> report obsolete redundancy 4 device type disk;
RMAN> # Allocate a channel of type delete.
RMAN> allocate channel for delete type disk;
RMAN> # Delete the backup set(s).
RMAN> change backupset 131, 132, 133 delete;
RMAN> # Release the allocated delete channel.
RMAN> release channel;
```

这个操作的缺点是必须运行report并根据report的输出结果生成删除命令。下一个任务将介绍删除陈旧备份的另一个方法。

为了查找并删除孤立的备份，运行几个增量级别为0的备份：

```
RMAN> run {execute script b_whole_inc0; };
RMAN> run {execute script b_whole_inc0; };
RMAN> run {execute script b_whole_inc0; };
RMAN> run {execute script b_whole_inc0; };
RMAN> report obsolete redundancy 4 device type disk;
```

这个obsolete报告将显示孤立的备份，删除以上显示的孤立备份。这些备份文件将从磁盘和目录中删除。

任务3：删除陈旧的备份

可以手工方式查找备份集编号并逐个删除。然而为了自动化删除陈旧的备份集，可以从恢复目录视图中查询备份集并将change命令spool到一个文件中。在RMAN下运行该文件将会删除陈旧的备份。生成的RMAN脚本文件需要下而包含了SQL*Plus命令的代码段：

```
define fil = '/tmp/delete_copies.rcv'
spool &fil
```

```
prompt allocate channel for delete type disk;;
SELECT 'change backupset '||bs_key||' delete;'
  FROM rc_backup_set
 WHERE completion_time > SYSDATE - 21;
prompt release channel;;
spool off;
```

当连接到目标数据库时，在RMAN提示符下调用/tmp/delete-copies.rcv将使目录删除21天以前生成的备份集。

注意 Oracle提供了一个脚本，该脚本使用report命令从恢复目录中找到孤立的备份然后自动删除这些备份。这个脚本位于Linux和NT的\$ORACLE_HOME/rdbms/demo/rman1.sh中。这是一个Unix shell脚本，如果需要在NT下运行，则必须按照DOS脚本进行编辑或者使用Unix shell环境。为了更改冗余的天数，可以简单地做相应修改。

任务4：查找过期的备份

大多数介质管理器都有文件保存期限。例如，如果一个文件超过了三个月而介质管理器的保存期也设置为三个月，则该文件将被标注为过期。介质管理器将不再对它进行访问，而且这些文件将被其他新的备份删除。将保持期与RMAN的crosscheck选项结合使用可以删除陈旧的备份。对于需要备份保留三个星期然后将RMAN备份片文件写入到磁盘的用户，可以在Linux操作系统下删除超过了三个星期的备份片：

```
LINUX> find /oradata/PRACTICE/backup/ -mtime -21 -print | xargs rm
```

一旦这些文件被删除，可以对备份集执行交叉检查：

```
RMAN> allocate channel for maintenance type disk;
RMAN> crosscheck backupsets;
RMAN> release channel;
```

所有被操作系统删除的文件都将被交叉检查crosscheck命令标记为过期。

警告 应该仔细操作，确保没有从磁盘上删除近期可能需要的备份。为了安全起见最好在其他地方保存几份以前的备份，或者在执行交叉检查（crosscheck）命令之前将备份转移到其他路径下。接着，对目录进行查询并完全确信不再需要这些备份了，然后再删除。

对于用RMAN向磁带上备份的用户，可以发布CROSSCHECK命令检查是否还可以访问这些被MML标注为过期的备份。如果MML标注它们过期了，则RMAN在目录中也将它们置为“过期”。

任务5：删除过期的备份

在这一任务中，你将删除前一任务中已经被标注为“过期”的孤立备份集和映像拷贝。为了删除物理的备份以及将资料库中的记录修改为DELETED，可以使用以下命令：

```
RMAN> delete expired backup;
```

当删除备份片后，运行report和list命令，确保可以将数据库恢复到三个星期以前的状态：

```
RMAN> # Make sure we can recover to 21 days previous
RMAN> list backup recoverable until time 'SYSDATE - 21';
```


Oracle 9i的保留策略

在Oracle 9i中，RMAN有了一个新的功能：保留策略。类似于介质管理器对磁带定义的保留策略，RMAN提供了根据恢复窗口或备份冗余为备份、映像拷贝或代理拷贝定义保留策略的能力。对于恢复窗口，RMAN将所有在当前定义的时间范围内不再用于恢复数据库的备份都定义为孤立备份。对于备份冗余，RMAN将所有备份的个数超过了定义次数的备份都标记为孤立备份。在目录中不删除任何对象，备份只被标记为孤立。然后，可以使用DELETE OBSOLETE命令删除这些陈旧的孤立备份。

13.5 疑难解答

如果在进行维护目录和RMAN备份时遇到了问题，可以参考以下列举的错误及注解：

1) RMAN-01005: syntax error: found "identifier" : expecting one of: "double-quoted-string, disk, equal, single-quoted-string"

RMAN-01008: the bad identifier was: diskx

RMAN-01007: at line 4 column 29 file: scripts.rcv

当在目录中创建存储脚本时，如果脚本中的命令不符合语法则会出现这个错误。检查文件中的行列，校正错误的语法并再次创建脚本文件。

2) RMAN-06091: no channel allocated for maintenance (of an appropriatetype)

change、crosscheck以及validate命令都需要适当类型（磁盘或者sbt_tape）的维护通道。在管理备份之前，首先要分配该通道。

3) RMAN-10035: exception raised in RPC: <file> is not a backup piece

这个消息表示存在一个该名称的备份片文件，但不是一个有效的备份片。查找正确的备份片文件或者删除该备份片。

13.6 小结

本章主要介绍了如何列出备份和映像拷贝保存在目录或目标控制文件中的信息。讲解了用RMAN生成的四个报告（need backup、unrecoverable、obsolete、schema）。通过交叉检查（crosscheck）和验证（validate）选项可以确认备份是否存在以及是否可以被成功还原。Change命令可以删除并管理已有的数据库备份和映像拷贝。为了详细了解不同的list和report选项及输出，请参看《Oracle8i Recovery Manager User's Guide and Reference》的第10章，该书的第3、4章则讲解了RMAN的资料库以及如何生成列表以及报告。

习题

请回答以下问题，测试本章所学的知识点。

1. RMAN中的哪两个命令可以用来查看目标控制文件和目录的内容？

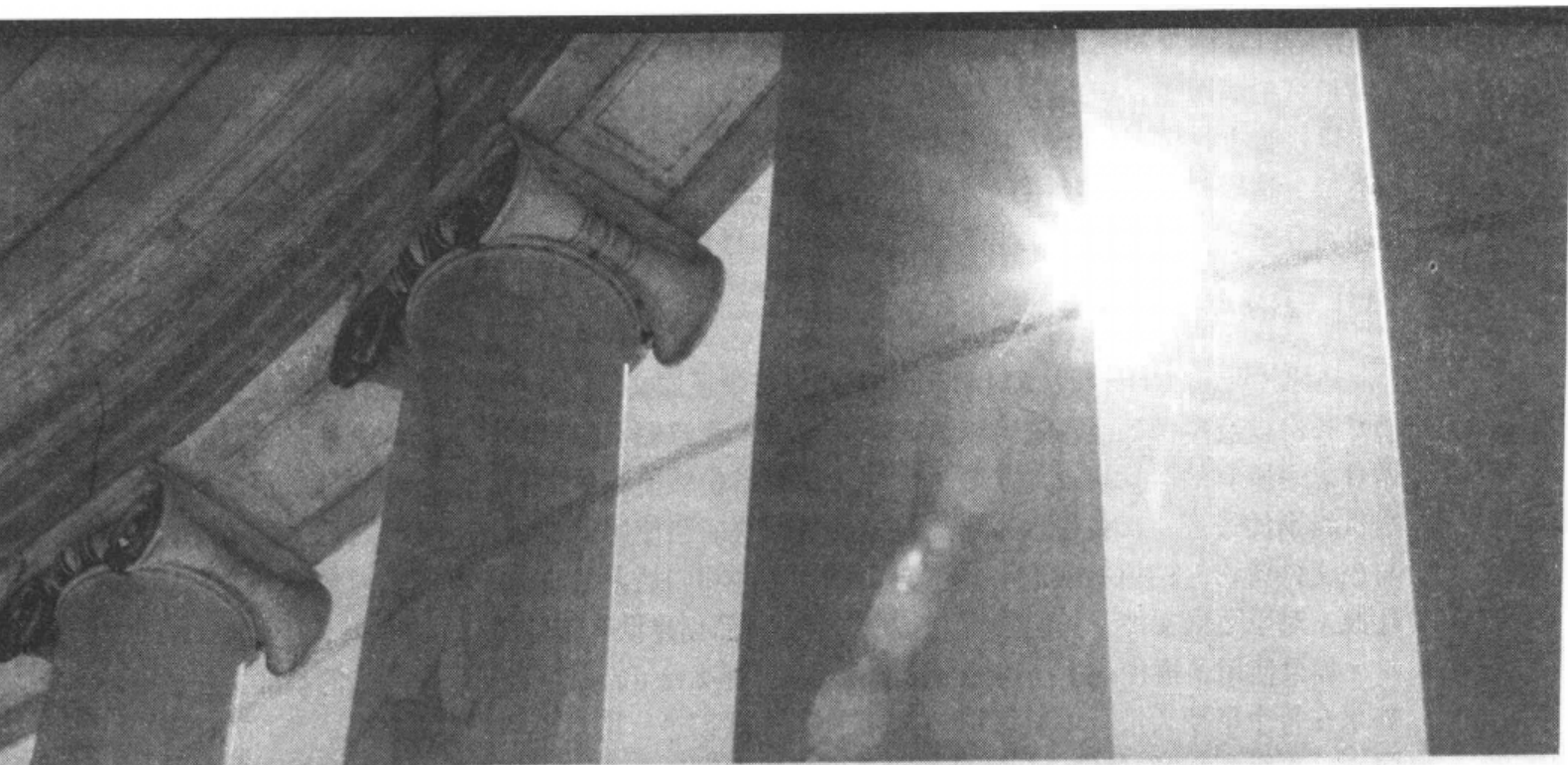
A. Select

B. Report

- C. List
 - D. Display
2. 哪个目录视图显示了RMAN生成的备份?
- A. v\$BACKUP_SETS
 - B. RC_COLA_SETS
 - C. v\$BACKUP_PIECES
 - D. RC_BACKUP_SETS
3. 对于RMAN, DELETE命令将从RMAN目录和物理介质中删除备份。
- A. True
 - B. False
4. RMAN存储的脚本可以被在同一目录模式中注册的同一目标数据库中的其他存储脚本调用。
- A. True
 - B. False
5. RMAN的validate操作与crosscheck操作有什么不同?
- A. crosscheck操作是互相比较备份; validate针对目录模式进行比较备份。
 - B. crosscheck操作检查备份介质上的文件并读取备份片的头部; validate则从备份介质中读取整个备份片文件并执行校验和验证。
 - C. crosscheck操作和validate完成同样的工作。
 - D. crosscheck在银行账户之间传送资金; validate为停车付费。

答案

- 1. B、C。REPORT和LIST命令格式化地显示从目录中获得的输出。
- 2. D。RC_BACKUP_SETS视图显示所有注册数据库的备份集的目录信息。目标数据库的v\$BACKUP_SETS视图只显示包含在控制文件中的备份集信息。
- 3. False。带有DELETE选项的CHANGE命令将从目录和备份介质中删除备份和映像拷贝。在CHANGE或DELETE命令之前必须分配相应介质类型(磁盘或SBT_TAPE)的维护通道。
- 4. True。存储脚本可以执行目录中同一目标数据库的其他存储脚本。
- 5. B。crosscheck和validate执行同样的功能,但是validate为了模拟还原的完整性需要读取整个备份片文件。因此,validate是一种更彻底的、资源敏感的测试。



第14章 RMAN 恢复

在第11章中为使用恢复目录配置了RMAN，在第12章中对PRACTICE数据库进行了备份，在第13章中学习了如何维护自己的RMAN目录。这一章中将应用RMAN来恢复PRACTICE数据库，以检查上述这些工作是否能协调统一。本章中的一些失效现象可能与第4、5章中的某些现象类似。这样就可以比较用户管理恢复与服务器管理恢复。用RMAN将一个失效的数据库调整返回到有序的工作状态涉及到两个重要的命令：还原（restore）与恢复（recover）。

14.1 RMAN还原

还原（restore）是指从RMAN所创建的备份中还原文件。这些还原的文件可以被使用或通过增量备份以及重做等应用进行恢复。在第12章中我解释了RMAN将备份一个控制文件或归档重做日志中所有的数据块。对于数据文件，则只备份被使用过的数据块，或者只备份根据定义的增量级别的最近一次备份以来所更改过的数据块。当使用RMAN还原文件之后，这个文件都会与该文件备份时的原始状态一样。对于控制文件和归档重做日志，则需要对所有的数据块进行还原。对于数据文件，只还原使用过的数据块，而创建没有备份的空数据块。

如果使用了第12章中描述的增量备份策略，那么restore命令将只还原准确的基准文件。例如，如果在每个星期天进行增量级别0的备份而每个星期进行增量级别1和2的备份，则RMAN将还原最近一次级别为0的备份。后续的增量备份需要在恢复（recover）命令中应用。

Restore命令使用的RMAN基准备份可以是以下这些数据文件备份类型的一种：映像拷贝、完全备份以及增量级别为0的备份。具体选择哪个数据文件基准备份很大程度上决定于文件的创建时间以及需要恢复到的时间。RMAN将选择比较新的备份并减少对恢复的需求。

当还原数据文件时，RMAN可以还原一个单一数据文件、具有几个名称的文件、一个或多个数据表空间的所有数据文件，或者数据库中的所有数据文件。当指定RMAN还原一个数据文件时，RMAN将会寻找最适合的备份或拷贝来还原该数据文件。

当还原数据库文件时，数据库应该处于什么状态呢？表14-1为该问题提供了一个快速解答。基本原则是：当一个文件正在被还原时，这个数据库文件不能被实例使用，因为当一个数据库文件正在被还原时该实例不能对其进行写入操作。在任何数据库文件还原期间，目标数据库的实例必须已经启动。当从备份集中还原任何非系统数据文件时，为了确保成功还原，这些数据文件（或表空间）必须处于脱机状态。如果是还原任何系统表空间数据文件，则数据库必须以加载（mount）模式启动。当还原一个控制文件时，实例必须以非加载（nomount）模式启动（因为数据库被加载后，该控制文件就要被使用）。归档日志在目标数据库加载或打开时都可以进行还原。

表14-1 还原操作时的数据库状态

数据库文件	数据库状态
任何非系统数据文件	加载或打开，但这些文件或表空间必须脱机
系统数据文件	加载
数据库（所有数据文件）	加载
归档日志	非加载、加载或打开
控制文件	非加载

为什么RMAN要求数据库在数据文件和归档日志还原期间处于加载状态？因为RMAN备份和数据库结构信息保存在控制文件中，而且RMAN需要访问它们。应该注意，除非预先进行了处理，否则默认情况下还原操作将替代已经存在的数据库文件。

RMAN的恢复

一旦数据文件还原之后，就可以发布恢复（recover）命令将数据库回退到以前的状态。RMAN将使用来自恢复目录或目标控制文件的信息来确定需要哪种增量备份以及应用哪些重做日志。

RMAN应用恢复操作需要经过四个独立的阶段，分别是：

- 1) 目标控制文件有时被更新来反映存储在恢复目录中的最新的信息。日志文件记录和数据文件记录经常在这时候被控制文件更新。这些动作发生在一个备份控制文件被还原以后。
- 2) 应用所有可以被应用的增量备份。
- 3) 检查所有还原的数据文件是否存在以及是否可以被恢复。不应该出现的数据文件（在恢复停止时间以后创建的数据文件）将从数据库中删除。在磁盘上查找需要的重做日志，将数据库回退。如果找到了这些重做日志就进行应用。
- 4) 磁盘上不存在的或需要用于恢复的重做日志文件都将被还原到磁盘上，并应用到数据文件上。

每个增量备份都建立在前一个增量备份之上，所以只需要更改被替换的数据块。如果可以这样操作，则RMAN会选择还原增量备份，而不是从归档日志文件中应用重做操作（应用增量备份比应用重做操作速度快）。一旦没有更多的增量备份可以应用，就需要进行重做操作将数据文件返回到恢复的状态。恢复操作一直持续到所有的重做信息都应用到了指定的数据文件上，或者到达了指定的停止点。默认情况下，RMAN执行完全恢复（所有数据文件都回退到当前联机重做日志文件中最后一次改变的时间）。如果控制文件也还原了，则RMAN不能应用联机重做文件，所以只能进行不完全恢复。恢复操作也可以持续到以UNTIL关键字指定的一个时间点。

技巧 当从备份的归档日志恢复时，归档日志首先被还原到磁盘上，然后再应用。在恢复之前应该检查磁盘空间是否足够保存这么多归档日志文件。

为了便于统一理解还原、恢复和增量备份，请参看图14-1。该图描述了第12章中讨论的增量备份策略。假设在星期六早晨级别为2的增量备份执行之前有一个数据文件丢失了，当RMAN还原并恢复该丢失的数据文件时，需要执行下列步骤进行完全恢复：

- 1) **还原数据文件** 应用备份集120，RMAN可以从星期天进行的整个数据库增量级别为0的备份中还原整个丢失的数据文件。这个基准备份包括了该数据文件所有使用过的数据块。该级别为0的备份作为后续还原工作的基准必须被还原。
- 2) **恢复数据文件** 一旦通过基准备份还原了数据文件，RMAN将使用适当的增量备份来恢复该数据文件。第一个将被应用的增量备份是备份集124（星期四的增量备份，级别为1）。备份集121、122和123并不需要，因为这些备份集中所有更改过的数据块已经都包含在备份集124中。然后，RMAN发现备份集125包含的数据块可以应用到这个已经还原的数据文件上，并能使其恢复到当前状态。如果备份集125中没有该数据文件的任何更改过的数据块，则该备份集不会被应用。接下来，RMAN发现不再需要其他的增量备份。RMAN为介质恢复查找需要的重做日志文件。如果这些文件在磁盘上，则从磁盘上读取。如果没有找到任何归

档文件，则RMAN将查找包含了所需重做文件的映像拷贝或备份集，并将其还原到磁盘上。还原完成之后，通过当前联机重做日志文件，这些文件就可以用于完全恢复了。

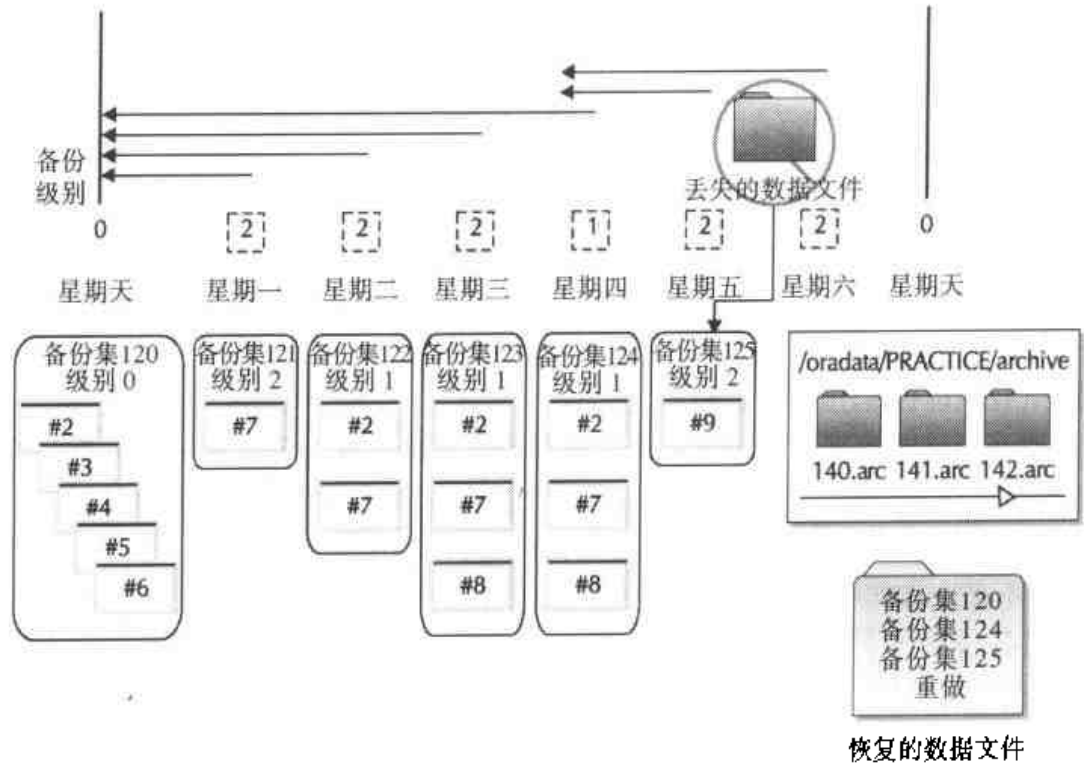


图14-1 使用RMAN的增量数据块级恢复

默认情况下，RMAN将会把所有数据文件还原到该文件备份时的初始位置。例如，RMAN 将把/oradata/PRACTICE/users01.dbf还原到目录/oradata/PRACTICE下，文件名为users01.dbf。归档日志在默认情况下被还原到归档转储目的路径下，控制文件则还原到数据库初始化文件（`init.ora`）中的CONTROL_FILES参数规定的路径下。可以在一个运行块中通过set属性来更改还原文件的位置。Set属性将在一个运行块中为命令定义数值（运行块是以“run”单词开始，以花括号结束范围内的一系列命令集合）。Set属性也可以指定恢复的时间点以及归档日志文件被还原以及恢复期间读取的位置。表14-2提供了运行块set属性用于还原和恢复选项的快速一览表。请注意，set命令与运行块中的set属性不同。在RMAN提示符下的set命令可以在表空间时间点恢复和复制数据库命令中定义文件名和其他事项。

表14-2 在RMAN运行块中用于恢复的set选项

Set关键字	描 述
为数据文件设置新名称	为特定的数据文件指定新的文件名和路径。如果没有定义该项，则RMAN将文件还原到其当前的路径和文件名
设置归档日志路径	为还原和恢复命令的归档日志文件定义不同的路径。如果没有定义该项，RMAN使用目标init.ora文件中定义的第一个归档存储目的的路径作为定义
Set until	为恢复定义停止点。不完全恢复可以完成到特定时间、日志序列号或者SCN。如果没有定义该项，RMAN将还原基准备份，并用增量备份和重做将所有数据文件恢复到当前状态

当使用RMAN进行还原和恢复时，环境变量起了重要的作用。有时需要设置环境变量使会话的字符集与数据库的字符集保持一致。设置NLS_LANG环境变量使它与目标数据库的字符集匹配。如果在RMAN的LIST、REPORT、SET UNTIL等命令中要使用特定的时间，则应该设置NLS_DATE环境变量与命令的日期字符串匹配。也可以在脚本中使用TO_DATE函数指向特定的日期。

```
LINUX> export NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1
LINUX> export NLS_DATE_FORMAT=YYYY-MM-DD:HH24:DD:SS
WINNT> set NLS_DATE_FORMAT=YYYY-MM-DD:HH24:MI:SS
WINNT> set NLS_LANG=AMERICAN_AMERICA.WE8ISO8859P1
```

本章中将使用第12章中的备份脚本文件生成的两个新的备份（也可以使用第12章的练习生成的备份）。图14-2描述了将要进行的操作。通过使用这些备份，RMAN可以恢复PRACTICE数据库。本章练习进行还原和恢复整个数据库、USERS表空间的数据文件，以及错误的删除表命令之后的不完全恢复操作。

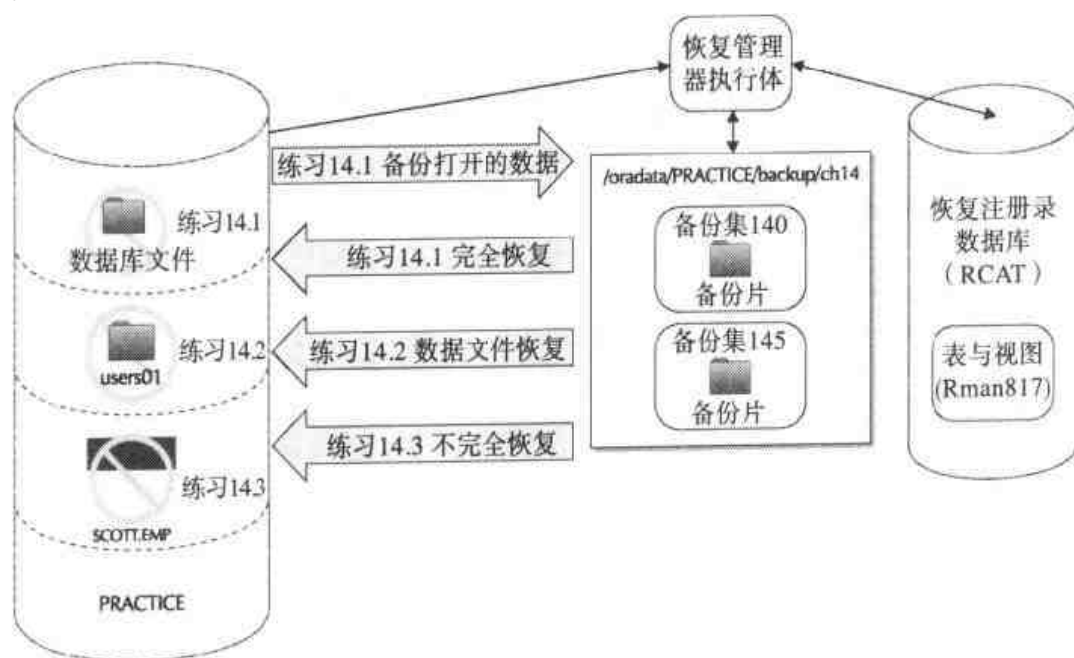


图14-2 对PRACTICE数据库用RMAN进行恢复练习

14.2 RMAN的还原与恢复

前面对还原和恢复命令有了大致的了解，现在将PRACTICE数据库作为目标数据库，RCAT数据库作为目录数据库，完成以下练习以便获得一些实际的经验。

在下面的练习中，使用整个数据库的一个备份来还原与恢复部分数据库。例如，可以使用整个数据库备份来还原TOOLS数据文件。但反之则不成立：不能只使用TOOLS表空间的备份来还原整个数据库。

练习14.1：还原整个数据库

假设遇到了灾难性的失效，丢失了数据库中所有数据文件，但控制文件和联机重做日志却完整无缺。那么如何使用RMAN备份从这样的失效中进行恢复呢？下面这个练习将讲述如何进

行RMAN还原与恢复。

任务描述	时间（分钟）
1. 执行整个数据库备份	5
2. 模拟PRACTICE失效	5
3. 还原与恢复数据库	10
4. 验证还原与恢复的数据库	5
总计时间	25

在该练习中，要执行两个整体数据库备份。然后，在PRACTICE数据库中删除所有数据文件。通过整体数据库备份，可以还原并恢复PRACTICE数据库的所有数据文件。

任务1：执行整个数据库备份

用第12章中生成的脚本创建两个PRACTICE数据库的备份。第一个进行的备份是一个整体数据库增量级别为0的备份，其中包含了当前的控制文件。在进行第二个备份之前，打开一个SQL*Plus会话更改数据库中一个表的数据。提交该更改后，返回到RMAN提示符下，运行第二个备份操作。第二个备份是整体数据库增量级别为1的备份，只拷贝自刚才级别为0的备份以来更改过的数据块。输入以下命令：

```
LINUX> export ORACLE_SID=PRACTICE
LINUX> rman
RMAN> connect target sys/practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @b_whole_inc0.rcv
RMAN> host;
LINUX> sqlplus /nolog
SQL> connect sys/practice
SQL> DELETE FROM tina.date_log WHERE create_date > SYSDATE;
SQL> INSERT INTO tina.date_log VALUES (SYSDATE+(365+14));
SQL> COMMIT;
SQL> exit
LINUX> exit
RMAN> @b_whole_inc1.rcv
```

警告 在RMAN和SQL*Plus中使用host命令可以提供一个运行操作系统命令的途径。应谨慎地返回到RMAN或SQL*Plus。否则就会混淆到底是哪个程序在运行并且发生了错误。

当这些命令运行结束时，将得到以下这些备份和数据。除非修改了备份脚本，否则备份片都将位于/oradata/PRACTICE/backup/ch12目录下。

- **备份：增量级别0** 第一个备份是运行了第12章中名为b_whole_inc0的备份脚本文件得到的。这个备份为PRACTICE数据库生成了一个独立的备份集，其备份片包含了所有数据文件中使用过的数据块和当前的控制文件。这个备份集的备份集号为140（这是生成时的号），备份片的键值为141。
- **表数据** 当第一个备份完成后，TINA.DATE_LOG表将做如下修改：在create-date列中插

入了一行表示未来14年日期的记录。表数据的更改引起了TOOLS表空间数据文件某些地方数据块的改变（假定该表仍然位于TOOLS表空间中）。被更改的数据块将在后续的增量备份中被选中。

- **备份:增量级别1** 第二个备份是运行了第12章中名为b_whole_incl的备份脚本文件得到的。这个备份生成了一个独立的备份集，其备份片包含了所有数据文件中使用过的数据块和自上次备份以来更改过的当前控制文件。TINA.DATE_LOG表更改过的数据块（以及其他）被写入到这个备份中。这个备份集的编号为145，备份片键值为146。

注意 为什么要生成两个备份？两者之间有何变化？在任务3中还原数据库时，RMAN将提取存储在第一个备份集中的数据块。然后，在恢复期间就可以应用到还原的数据文件中。

任务2：模拟PRACTICE失效

在操作系统中从/oradata/PRACTICE目录下删除所有的数据文件。联机重做日志和控制文件保持不变。在Linux上，可以在数据库打开的情况下删除这些文件。在Windows中，需要关闭数据库后才能删除数据库文件。为了不考虑操作系统而执行同样的操作，应该从RMAN中关闭数据库，然后在各自的操作系统下删除所有数据库文件。

```
LINUX> rman
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> shutdown abort
RMAN> host;
LINUX> rm /oradata/PRACTICE/*.dbf
LINUX> exit
```

既然数据库文件已经删除了，那么在启动数据库时就会遇到错误。当实例启动后并加载了控制文件后，该实例将查找编号为1的数据文件（system01.dbf）。因为该文件不存在，所以RMAN将显示一个错误消息。这个消息与试图用SQL*Plus启动数据库得到的消息是一样的：

```
RMAN> startup
RMAN-06193: connected to target database (not started)
RMAN-06196: Oracle instance started
RMAN-06199: database mounted
RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-06003: ORACLE error from target database:
ORA-01157: cannot identify/lock data file 1 - see DBWR trace file
ORA-01110: data file 1: '/oradata/PRACTICE/system01.dbf'
RMAN-06097: text of failing SQL statement: alter database open
RMAN-06099: error occurred in source file: krmk.pc, line: 4228
```

这时，实例启动了，数据库控制文件加载了，但数据库没有被打开。

任务3：还原与恢复数据库

在RMAN提示符下，列出任务1中生成的两个备份。使用tag和时间（最近7天）选项，执行

下列list命令列出这两个增量备份：

```

RMAN> list backup completed after 'sysdate-7' tag = WHOLE_INC0;
RMAN> list backup completed after 'sysdate-7' tag = WHOLE_INC1;

```

请注意列出的备份级别、set数以及备份片的名称。

应用RMAN，使用以下脚本还原并恢复PRACTICE数据库：

```

# Restore and Recover Database using RMAN
shutdown abort;
startup mount;
run {
    allocate channel d1 type disk;
    restore database;
    recover database;
}
alter database open;

```

恢复脚本内容不多，但每一行都完成了许多工作。下面将详细解释这些命令。在还原整个数据库时，数据库必须处于没有被打开的状态。Shutdown abort是有意义的，因为磁盘上不再有数据文件。

- Run restore还原与恢复recover命令必须位于运行块之内。该块内的所有命令都在功能上加以分组。
- Allocate 分配的通道将在restore命令中读取备份片文件并重新生成数据文件。在应用增量备份和重做进行恢复数据文件时也需要这个通道。
- Restore 通过RMAN目录可以知道整个数据库有哪些文件存在。通过这个信息，restore database命令会找到最近的基准备份（完全备份、映像拷贝或级别为0的增量备份）。
- Recover 当数据文件还原之后，recover命令将寻找并应用任何后续的增量备份。备份片中的数据块将被写入到已经从基准备份还原的数据文件中。因为这个练习场景中包括了级别为0的备份和级别为1的备份，所以级别为1的备份将被用于回退包含在其备份片中的更改的数据块。当应用了增量备份之后，需要进行重做应用，以便使数据文件返回到当前包含有控制文件的状态。RMAN将寻找并应用恢复（如果需要）所需要的归档日志文件，然后使用当前的联机重做日志文件。很可能第一次恢复时只需要当前联机重做日志文件，这是因为PRACTICE数据库上没有发生更多的动作。

当RMAN完成还原与恢复后，RMAN将释放d1磁盘通道。最后就可以为用户打开数据库了。运行前面讲述的脚本中的命令，并观察RMAN如何工作。下面是RMAN命令的输出：

```

RMAN> @r_whole.rcv
RMAN> # Restore and Recover Database using RMAN
RMAN> run {
2>     allocate channel d1 type disk;
3>     restore database;
4>     recover database;
5> }
RMAN-03022: compiling command: allocate

```

```

RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: dl
RMAN-08500: channel dl: sid=14 devtype=DISK
RMAN-03022: compiling command: restore
RMAN-03022: compiling command: IRESTORE
RMAN-03023: executing command: IRESTORE
RMAN-08016: channel dl: starting datafile backupset restore
RMAN-08502: set_count=140 set_stamp=447707604 creation_time=14-JAN-02
RMAN-08089: channel dl: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00001 to /oradata/PRACTICE/system01.dbf
RMAN-08523: restoring datafile 00002 to /oradata/PRACTICE/rbs01.dbf
RMAN-08523: restoring datafile 00003 to /oradata/PRACTICE/users01.dbf
RMAN-08523: restoring datafile 00004 to /oradata/PRACTICE/temp01.dbf
RMAN-08523: restoring datafile 00005 to /oradata/PRACTICE/tools01.dbf
RMAN-08523: restoring datafile 00006 to /oradata/PRACTICE/indx01.dbf
RMAN-08523: restoring datafile 00007 to /oradata/PRACTICE/users02.dbf
RMAN-08023: channel dl: restored backup piece 1
RMAN-08511: piece handle=/oradata/PRACTICE/backup/ch14/db_PRACTICE_140_1_44770760L
RMAN-08024: channel dl: restore complete
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-03022: compiling command: recover
RMAN-03022: compiling command: recover(1)
RMAN-03022: compiling command: recover(2)
RMAN-03023: executing command: recover(2)
RMAN-08039: channel dl: starting incremental datafile backupset restore
RMAN-08502: set_count=145 set_stamp=447707833 creation_time=14-JAN-02
RMAN-08089: channel dl: specifying datafile(s) to restore from backup set
RMAN-08509: destination for restore of datafile 00001: /oradata/PRACTICE/system01.dbf
RMAN-08509: destination for restore of datafile 00002: /oradata/PRACTICE/rbs01.dbf
RMAN-08509: destination for restore of datafile 00003: /oradata/PRACTICE/users01.dbf
RMAN-08509: destination for restore of datafile 00004: /oradata/PRACTICE/temp01.dbf
RMAN-08509: destination for restore of datafile 00005: /oradata/PRACTICE/tools01.dbf
RMAN-08509: destination for restore of datafile 00006: /oradata/PRACTICE/indx01.dbf
RMAN-08509: destination for restore of datafile 00007: /oradata/PRACTICE/users02.dbf
RMAN-08023: channel dl: restored backup piece 1
RMAN-08511: piece handle=/oradata/PRACTICE/backup/ch14/db_PRACTICE_145_1_4477078L
RMAN-08024: channel dl: restore complete
RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
RMAN-08054: starting media recovery
RMAN-08055: media recovery complete
RMAN-03022: compiling command: recover(4)
RMAN-08031: released channel: dl
RMAN> alter database open;
RMAN-03022: compiling command: alter db
RMAN-06400: database opened
```

```
RMAN> **end-of-file**
```

请注意输出的结果并特别留意以下几个重要的信息：

- **channel d1: sid=14 devtype=DISK** 当分配了一种磁盘类型的通道时，该通道将产生一个连接目标数据库实例的连接。这个连接在目标实例上获得一个系统标识（SID），这个标识可以通过SQL*Plus连接后在v\$session中查询得到。
- **set_count=140** restore命令寻找到的最近的增量级别为0的数据库备份是备份集140。RMAN通过该备份集的备份片重新构建PRACTICE数据库的所有数据文件。
- **db_PRACTICE_140_1_44770760L** 这个备份集140的单一备份片包含了还原需要的所有数据块。这个操作系统文件将被查询、读取，并用于重新构建每个数据文件。
- **restore complete** 一旦所有数据文件的还原完成，RMAN将会发出通知。还原完成之后，数据文件就会像它们被备份时的一样。一旦数据文件被从基准备份中还原，restore命令就完成了。
- **set_count=145** 在恢复输出中，备份集145的应用是作为恢复过程的一部分进行的（即使输出结果中表示备份是被还原的）。第二个用于恢复的备份集是增量级别为1的备份。对增量级别为1的备份的应用是恢复命令产生的结果。这种更改对应了表TINA.DATE_LOG在任务1中对数据块的更改。那个被修改的数据块以及其他数据块都包含在该数据集中。现在，RMAN将把这个更改的数据块存放到刚从备份集还原的数据文件中。
- **db_PRACTICE_145_1_4477078L** 备份集145完全包含在这个备份片文件中。通过通道d1从该文件中读取数据块并应用到刚才还原的数据文件中。
- **restore complete** 当第二个备份集被应用后，RMAN将会通知这个增量数据块的还原工作已经完成。一旦这个还原完成，这个数据文件就会拥有与b_whole_incl备份集脚本运行时一样的数据。
- **starting media recovery** 当所有最近的增量备份都被应用之后，就需要进行重做以将数据库恢复到指定的恢复停止时间的状态（默认是恢复到当前/完全恢复）。将要读取重做日志，并应用到还原的数据文件上。重做工作包括前滚数据库修改操作，回退没有提交的修改操作。
- **recover(1) through recover(4)** RMAN的恢复包括4个步骤；因此会看到recover(1) through recover(4)。如果目录数据库中的新数据必须被反映到控制文件中，则第一阶段将更改控制文件；第二阶段还原并应用增量备份；如果日志文件保存在磁盘上则第三阶段应用重做信息；如果磁盘上没有日志文件，则第四阶段还原日志文件，并向数据文件应用重做信息。
- **media recovery complete** 当介质恢复将数据库的当前状态输入到控制文件中时，RMAN将会通知：数据库已准备好，可以打开了。

完全恢复之后，不需要重置重做日志文件就可以打开数据库。

任务4：验证还原与恢复的数据库

检查还原的数据文件是否存在于操作系统上。在操作系统提示符下查找/oradata/PRACTICE目录下的内容。其中的数据文件应该与任务2中删除这些数据文件之前一样。打开一个SQL*Plus

会话，检查TINA.TIME_LOG表。在表中查找最大的create_date以确认最新的修改是否存在。

```
SQL> SELECT max(create_date) FROM tina.date_log;
```

如果看到了一个未来14年的日期记录，就验证该表已经进行了最新的数据更改。

怎么样，用RMAN进行恢复非常容易吧！

练习14.2：还原并恢复数据文件

如果个别的数据文件发生丢失或损坏，则数据库处于打开状态时可以用RMAN轻松地恢复该文件（只要该文件不属于系统表空间也不包含有激活的回退段）。RMAN可以对一个或多个表空间中的数据文件进行还原与恢复。

任务描述	时间（分钟）
1. 模拟PRACTICE失效	5
2. 还原并恢复数据文件	5
3. 验证表空间恢复	5
4. 用备份的归档日志还原与恢复	5
总计时间	20

在本练习中将要对数据库的于集执行完全恢复，与前一个练习类似。但是不同的是，在数据库仍然处于打开状态时，只还原并恢复单个表空间的一个数据文件。

任务1：模拟PRACTICE失效

在中断数据库之前，需要向SCOTT的一个表中增加一行记录：用插入语句在department表中增加一个部门。然后再增加一个部门，但回退后一个更改。一旦还原与恢复完成，将会出现Support部门但却没有MIS部门。

```
SQL> connect sys/practice@practice;
SQL> INSERT INTO dept (deptno, dname, loc) VALUES (50, 'SUPPORT', 'ATLANTA');
SQL> COMMIT;
SQL> INSERT INTO dept (deptno, dname, loc) VALUES (60, 'MIS', 'DENVER');
SQL> ROLLBACK;
SQL> ALTER SYSTEM SWITCH LOGFILE;
SQL> ALTER SYSTEM SWITCH LOGFILE;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

接下来，从USERS表空间中删除第一个数据文件，在操作系统中删除该文件。因为Windows操作系统对于联机的数据文件加锁了，所以该数据文件在删除之前应先脱机。

```
SQL> ALTER TABLESPACE USERS OFFLINE NORMAL;
SQL> host;
LINUX> rm /oradata/PRACTICE/users01.dbf
LINUX> exit
SQL> ALTER TABLESPACE USERS ONLINE;
ERROR at line 1:
ORA-01157: cannot identify/lock data file 3 - see DBWR trace file
ORA-01110: data file 3: '/oradata/PRACTICE/users01.dbf'
```

下一个任务中，将演示如何恢复该表空间。

任务2：还原并恢复数据文件

当需要恢复一个数据文件时，可以用下面的命令来恢复单个数据文件。包含该数据文件的表空间先脱机，还原并恢复该数据文件，然后再将该表空间联机。

```
sql 'alter tablespace users offline immediate';
run {
    allocate channel d1 type disk;
    restore datafile 3; # Specify the datafile number or name
    recover tablespace users;
    sql 'alter tablespace users online ';
}
```

下面是对这个恢复情况各个命令的简要解释：

- **restore** RMAN应用目录或者目标控制文件查找最新的映像拷贝、完全或已经还原的增量级别为0的备份的数据文件。通过使用适当的备份片，就还原了该数据文件。
- **Recover** 一旦通过增量级别为0的备份片文件还原了丢失的数据文件，该文件必须被恢复到数据库的当前状态。恢复命令首先确定是否从增量备份中向数据文件拷贝了任何数据块。当向还原的数据文件还原数据块时，RMAN读取重做日志（归档日志以及联机日志），并将这些重做应用到USERS表空间的数据文件上。除了当前还原的数据文件外，其他的数据文件都是当前状态，所以只有对刚才还原的/oradata/PRACTICE/users01.dbf文件和/oradata/PRACTICE/users02.dbf文件应用重做。增加的Support和MIS部门并没有包含在RMAN备份中，这些记录包含在重做日志文件中。因此，通过重做操作将进行插入Support部门的操作。而插入MIS部门和后续的回退操作也被重做流所应用。

在为了将表空间或数据文件恢复到脱机之前，最好先检查这样操作将会影响哪些对象以及短时间的不连续性。下面两个SQL语句都可以完成这个工作：

```
SQL> rem For offlining a tablespace:
SQL> SELECT owner, segment_name, segment_type
2     FROM dba_extents
3     WHERE tablespace_name = '<tablespace being recovered>'
4     ORDER BY owner, segment_type, segment_name;
SQL> rem For offlining a datafile:
SQL> SELECT owner, segment_name, segment_type, count(*) extents
2     FROM dba_extents
3     WHERE file_id = <file number being offlined>
4     GROUP BY owner, segment_name, segment_type;
```

与前面的练习类似，在RMAN命令行提示符下运行列出的脚本文件：

```
LINUX> rman
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @r_datafile.rcv
```

当还原与恢复脚本运行时，其输出类似于前面屏幕上的输出结果。我摘录了几个片段来说

明在脚本运行时，RMAN进行的操作：RMAN为还原与恢复数据文件分配通道；RMAN选择编号为140的备份集来还原user01.dbf数据文件；可以从v\$datafile控制文件的file#中得知该文件的数据文件编号为00003。

```
RMAN-08016: channel d1: starting datafile backupset restore
RMAN-08502: set_count=140 set_stamp=447680630 creation_time=14-JAN-02
RMAN-08089: channel d1: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00003 to /oradata/PRACTICE/users01.dbf
```

该数据文件被还原后，恢复命令继续工作。RMAN选择所有最近的增量备份应用到数据文件中。当确认145号备份集中包含了这个数据文件的备份之后，RMAN读取这个备份片文件并为此新建的还原文件应用所有更改过的数据块。

```
RMAN-08039: channel d1: starting incremental datafile backupset restore
RMAN-08502: set_count=145 set_stamp=447680803 creation_time=14-JAN-02
RMAN-08089: channel d1: specifying datafile(s) to restore from backup set
RMAN-08509: destination for restore of datafile 00003:
/oradata/PRACTICE/users01.dbf
```

一旦RMAN向还原的数据文件应用了所有需要的增量备份之后，介质恢复就开始工作。接下来，通过介质恢复就一定能将还原的数据文件返回到数据库其他部分的当前状态。在恢复期间，必须要读取并应用重做日志文件使还原的数据文件保持一致。我增加了三个日志切换命令，确保在恢复期间获得归档日志文件。当最后一部分备份被应用之后，需要当前重做日志文件序列号的重做操作。在这个输出表中，最近的增量备份开始时的当前联机重做日志文件的序列号为141。所有的重做日志文件（包括当前的联机重做日志文件）都必须应用到该还原的数据文件上。

```
RMAN-06050: archivelog thread 1 sequence 1 is already on disk as file
/oradata/PRACTICE/archive/141.arc
RMAN-06050: archivelog thread 1 sequence 2 is already on disk as file
/oradata/PRACTICE/archive/142.arc
RMAN-06050: archivelog thread 1 sequence 3 is already on disk as file
/oradata/PRACTICE/archive/143.arc
...
RMAN-08055: media recovery complete
```

RMAN将找到磁盘上的这些归档日志文件。一旦所有的日志文件都被应用后，将出现介质恢复完成的消息。这些工作完成之后，该表空间就成功地恢复到了联机状态，可以投入使用了。

当然也可以还原丢失了一个数据文件的整个表空间。还原整个表空间将导致RMAN覆盖该表空间所有已经存在的数据文件。

```
sql 'alter tablespace users offline immediate';
run {
  allocate channel d1 type disk;
  restore tablespace users;
  recover tablespace users;
  sql 'alter tablespace users online';
}
```


任务3：验证表空间恢复

成功地对丢失的USERS表空间数据文件进行还原与恢复后，可以在操作系统下检查RMAN还原的文件。同样也可以查询SCOTT的department表，这时应该看到Support部门而找不到MIS部门。

```
SQL> connect scott/tiger;
SQL> select * from dept;
DEPTNO DNAME          LOC
-----
10 ACCOUNTING        NEW YORK
20 RESEARCH           DALLAS
30 SALES              CHICAGO
40 OPERATIONS         BOSTON
50 SUPPORT            ATLANTA
```

任务4：用备份的归档日志还原与恢复

在前面三个任务中，介绍了如何用增量备份和磁盘上已经存在的归档日志文件还原与恢复一个单一的数据文件。如果在恢复期间有一些恢复所需要的归档日志文件并没有在磁盘上找到，又该如何处理呢？如果归档日志文件存在于可用的备份或映像拷贝中，则RMAN为了恢复工作将还原并应用这些所需的归档文件。

为了试验RMAN如何向磁盘上还原归档日志文件和在恢复期间使用还原的文件，本练习将重复任务1-3并恢复一个丢失的数据文件。但这次要走一个小弯路：在几个日志切换命令之后向磁盘备份归档日志文件。备份完成之后，重新命名这些最新的归档日志文件。当RMAN进行恢复操作时，将会在磁盘上查找这些重命名的归档日志文件。如果RMAN没有找到，则将从磁盘上的备份还原这些归档文件。通过使用这些还原的归档文件，RMAN就完成了恢复还原的users01.dbf数据文件的任务。（本任务中只重新命名而不删除这些文件，是为了防止某些错误的发生。）

```
RMAN> connect target sys/practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @b_archive_2days.rcv
RMAN> host;
LINUX> cd /oradata/PRACTICE/archive
LINUX> mv 141.arc 141.arc.bak
LINUX> mv 142.arc 142.arc.bak
LINUX> mv 143.arc 143.arc.bak
LINUX> exit
```

第12章中，我讲解了一个名为b_archive_2days.rcv的脚本，这个脚本主要是为了备份最近两天之内生成的归档日志文件。在该脚本的执行过程中，会出现以下的消息：

```
RMAN-08502: set_count=148 set_stamp=447945049 creation_time=08-DEC-01
RMAN-08014: channel dl: specifying archivelog(s) in backup set
RMAN-08504: input archivelog thread=1 sequence=141 recid=105 stamp=447941127
RMAN-08504: input archivelog thread=1 sequence=142 recid=106 stamp=447941449
RMAN-08504: input archivelog thread=1 sequence=143 recid=107 stamp=447945030
```

这意味着位于归档转储目的路径的归档日志被拷贝到了备份集中。当备份这些文件时，会显示它们的重做日志文件序列号。

备份完成之后，将这些归档文件删除，RMAN在磁盘上就找不到这些文件了。为了响应r_datafile.rcv中的恢复命令，RMAN必须使用执行脚本文件b_archive_2days.rcv产生的归档文件的备份。接着删除USERS表空间的第一个数据文件。在操作系统中，删除该文件。因为Windows操作系统对于联机的数据文件都加锁，所以在删除该数据文件之前应先将表空间脱机：

```
SQL> ALTER TABLESPACE USERS OFFLINE NORMAL;
SQL> host;
LINUX> rm /oradata/PRACTICE/users01.dbf
LINUX> exit
SQL> ALTER TABLESPACE USERS ONLINE;
ERROR at line 1:
ORA-01157: cannot identify/lock data file 3 - see DBWR trace file
ORA-01110: data file 3: '/oradata/PRACTICE/users01.dbf'
```

已经备份的归档文件和磁盘上的归档文件都被删除了，RMAN恢复时就不能使用这些文件了。运行本练习任务2中的恢复命令。

```
LINUX> rman
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @r_datafile.rcv
```

在还原过程中，因为在最近一次任务后没有进行其他新的备份，所以恢复将使用与前面应用相同的数据文件的增量备份。与前面的恢复不同的是：恢复所需要的归档日志文件需要从备份中还原。这时将出现一个说明归档日志文件正被还原到默认路径的消息。读取名为ar_PRACTICE_148_1_447946582的备份片，并用于创建/oradata/PRACTICE/archive目录下的归档日志文件141.arc、142.arc以及143.arc。一旦恢复命令的这个还原部分完成，还原的归档日志文件将被用于恢复已被还原的数据文件。

```
RMAN-08017: channel dl: starting archivelog restore to default destination
RMAN-08022: channel dl: restoring archivelog
RMAN-08510: archivelog thread=1 sequence=141
RMAN-08022: channel dl: restoring archivelog
RMAN-08510: archivelog thread=1 sequence=142
RMAN-08022: channel dl: restoring archivelog
RMAN-08510: archivelog thread=1 sequence=143
RMAN-08023: channel dl: restored backup piece 1
RMAN-08511: piece
handle=/oradata/PRACTICE/backup/ch12/ar_PRACTICE_148_1_447946582
tag=null params=NULL
RMAN-08024: channel dl: restore complete
RMAN-06050: archivelog thread 1 sequence 2 is already on disk as file
/oradata/PRACTICE/archive/142.arc
RMAN-06050: archivelog thread 1 sequence 3 is already on disk as file
/oradata/PRACTICE/archive/143.arc
```

```

RMAN-08515: archivelog filename=/oradata/PRACTICE/archive/141.arc thread=1
sequence=141
RMAN-08515: archivelog filename=/oradata/PRACTICE/archive/142.arc thread=1
sequence=142
RMAN-08515: archivelog filename=/oradata/PRACTICE/archive/143.arc thread=1
sequence=143
RMAN-08055: media recovery complete
RMAN-08031: released channel: dl

```

在本例中，归档日志文件被备份到148备份集中。RMAN知道这个备份及其名为ar_PRACTICE_148_1_447946582的备份片文件。作为恢复的一个步骤，这个归档文件将被还原到默认的归档转储路径下。一旦所需要的归档日志被还原后，归档日志文件将被应用于还原的数据文件使其前滚，这样RMAN才能在出现归档文件在磁盘上的情况下进行恢复操作。这也是RMAN强大功能的又一有力说明。服务器管理的恢复操作保留了对备份的跟踪记录，能快速调用这些备份进行还原与恢复Oracle数据库。不需要人工记录备份的位置，RMAN已经处理了所有细节。

因为RMAN在归档存储路径下还原了重新命名的归档日志文件，所以可以删除前面重新命名的原始归档日志文件：

```

LINUX> rm 141.arc.bak
LINUX> rm 142.arc.bak
LINUX> rm 143.arc.bak

```

练习14.3：不完全数据库恢复

为了恢复用户或开发者偶然删除的表，可以执行第4、5章中的不完全数据库恢复。如果使用RMAN，则在数据库关闭时通过发布还原与恢复数据库的命令就可以恢复所有的数据文件。通过指定时间、日志序列号或者SCN可以设置RMAN的恢复停止点。根据停止点，RMAN将选择最合适的备份以及映像拷贝并将数据库还原到与停止点的要求最近的状态。重做操作也持续到定义的停止点。当RMAN完成了这些工作之后，剩下的就是重置日志并打开数据库。因为是不完全的恢复，所以需要重置日志。

任务描述	时间（分钟）
1. 模拟PRACTICE失效	5
2. 不完全还原与恢复数据库	10
3. 验证表空间的恢复	5
总计时间	20

在本练习中，将演示RMAN如何将数据库恢复到过去某一时间点。我将讲解如何创建一个环境以提示将要执行的是不完全恢复。本练习将删除一个表并使用本章前面进行的备份将PRACTICE数据库恢复到该表被删除之前的某一时间点。

任务1：模拟PRACTICE失效

在本任务中，将要从PRACTICE数据库中删除一个名为SCOTT.EMP的表。为了确认能如期地进行不完全恢复，需要对TINA.DATE_LOG表进行清理和设置。在PRACTICE数据库的Tina表中插

入日期标志时间。在前几章中，为了演示，都在该表中插入了未来的日期记录。在删除employee表之前，删除这些未来的日期记录，只保留过去的日期记录。当删除employee表之后，在TINA.DATE_LOG表中插入一行表示未来14年的记录。最后，进行三次日志切换。这些日志切换确保稍后进行的恢复操作使用归档的重做日志。

```
SQL> CONNECT sys/practice
SQL> SET TIME ON
12:01:02 SQL> DELETE FROM tina.date_log WHERE create_date > SYSDATE;
12:02:12 SQL> DROP TABLE SCOTT.EMP;
12:03:23 SQL> INSERT INTO tina.date_log VALUES (SYSDATE+(365*14));
12:03:27 SQL> COMMIT;
12:03:27 SQL> SELECT MAX(create_date) FROM tina.date_log;
```

在12:01:02，DATE_LOG表中任何有关未来日期的记录都被删除了。在12:02:12，employee表被删除。当删除该表之后，在DATE_LOG表中插入了一个未来的日期记录。在PRACTICE数据库恢复到删除表语句之前的时间点时，找不到一个未来14年的CREATE_DATE记录（最新插入的日期记录）。恢复操作将在删除表语句之前停止应用重做记录。

任务2：不完全还原与恢复数据库

不完全的恢复首先还原在期望恢复的时间点之前进行的备份集。假设这个时间之前的备份是可用的，RMAN将选择最适合的一个备份。

```
run {
  # set time to just before data was lost
  set until time "TO_DATE('01/14/2002 12:02:10','MM/DD/YYYY
HH24:MI:SS')";
  shutdown immediate;
  startup mount;
  allocate channel d1 type disk;
  restore database;
  recover database;
  alter database open resetlogs;
}
```

不完全恢复意味着整个数据库必须被还原到以前的某个时间点，包括控制文件。有效地执行时间点恢复的技巧在于set命令。Set until命令通知RMAN用恢复时间最新的备份来还原数据库。这样恢复才能将数据文件返回到指定的时间点。在本示例中，删除employee表的确切时间是知道的，可以将恢复的最后时间设置到这个时间之前。当然也可以选择指定日志序列号或者SCN来设置恢复的停止点。

在这个删除表的场景中，基于时间的恢复是有意义的。可以用前面讲述过的各种方法来执行不完全恢复。如果必须恢复（但包含丢失或损坏的归档日志文件）时，可以恢复到这个指定的日志文件（即，损坏或丢失的日志文件之前）。为了指定一个日志序列号的停止点，可以修改设置运行的属性如下：

```
set until logseq 1400 thread 1;
```

这个set命令将一直应用归档日志文件，直到序列号1399为止。

比较少见的不完全恢复选项是基于SCN的恢复。可以修改set的运行属性如下，指定一个100000的SCN停止点：

```
set until SCN 100000;
```

如果数据库发生了某些损坏而且告警日志在损坏发生时提供了一个相关的SCN值（或者可以用第10章中的LogMiner来查找特定命令/错误的SCN），那么可以使用这种恢复。一旦设置了恢复的停止，就可以将所有的工作交给RMAN来处理。在查看以前的r_whole_incomplete.rcv脚本时，请注意下列关于RMAN不完全数据库恢复的几个问题：

- 在还原与恢复开始之前，关闭数据库。
- 分配通道的语句必须在数据库实例重新启动之后执行。分配通道命令打开一个数据库连接，该连接在数据库关闭时断开。
- 更改数据库打开的命令可以在运行块之外执行。

RMAN进行的不完全恢复只有以下两个地方与完全恢复不同：

- RMAN在恢复停止点之前从基准备份还原所有的数据文件。这些还原的文件根据停止点，可能是最新的基准备份，也可能不是。
- 恢复一直持续到停止点。RMAN比较智能，能够知道应该还原并应用哪些增量备份。RMAN将会找到并应用归档重做日志文件（而且有可能是当前的重做文件），一直到停止点为止。

不完全恢复的输出结果非常类似于练习14.1中的完全恢复执行的结果。在输出中并没有什么信息说明是时间点恢复。当以重置日志打开数据库时，RMAN将更新目录中的新记录，有效地执行RESET DATABASE命令。RESET DATABASE命令从目标数据库控制文件中获得更新RMAN目录的新记录的信息。

```
RMAN> alter database open resetlogs;
RMAN-03022: compiling command: alter db
RMAN-06400: database opened
RMAN-03023: executing command: alter db
RMAN-08006: database registered in recovery catalog
RMAN-03023: executing command: full resync
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete
```

在验证恢复结果之前，我增加了一个脚注。也可以在这些命令的后面都用一个until语句为还原和恢复增加一个停止点：

```
restore database until time "TO_DATE('01/14/2002 12:02:10','MM/DD/YYYY HH24:MI:SS')";
recover database until time "TO_DATE('01/14/2002 12:02:10','MM/DD/YYYY HH24:MI:SS')";
```

还原与恢复命令中的停止点也可以通过日志序列号或SCN当作时间来定义。同样地，还原与恢复的停止点不能具有同样的时间。根据上面设置的选项，在还原与恢复命令停止点。

任务3：验证表空间的恢复

进行重置日志操作之后，就得到了数据库的一个新的形态。可以使用LIST INCARNATION

命令观察该数据库的新形态。

```
RMAN> list incarnation;
List of Database Incarnations
DB Key   Inc Key DB Name   DB ID                CUR Reset SCN   Reset Time
-----
232      233      PRACTICE 2630335893         NO   83852           03-JAN-02
232      1190     PRACTICE 2630335893         YES 476863          14-JAN-02
```

注意，该状态的日期将是执行重置日志操作的日期。同时也有重置日志文件被重置的数据库的SCN值。尽管重做日志被重置了，而且日志序列号也从1开始记数，但数据文件的头部以及控制文件的SCN并没有被重置，而且继续沿用重置日志操作之后的序列。

当不完全恢复完成之后，请检查是否已经恢复了Scott的employee表。打开一个SQL*Plus会话并从该表中查询。应该看到不完全恢复已经完成了。主要的目标：

```
SQL> SELECT * FROM scott.emp;
```

接下来，检查DATE_LOG表中最大创建日期的值。查看是否在该表中有一个未来14年日期的记录。这条记录不应该出现在表中，因为恢复停止在删除表语句之前，也应该在插入未来14年这个日期记录以前。因此，包含有在DATE_LOG表中插入语句的重做并没有被应用（其他超过恢复停止点的重做都没有被应用）。

```
SQL> SELECT MAX(create_date) FROM tina.date_log;
```

当PRACTICE数据库恢复之后，还需要完成许多重要的事情。因为日志被重置了以确保进行以下操作：

- 执行新的基准备份。通过重置日志操作（使用带有当前日志备份的当前重做日志）并不能轻易地进行数据库恢复。
- 保护或删除任何现存的日志备份。虽然可以使用当前日志备份来恢复数据库，但是要通过重置日志操作来前滚是非常困难的。因此，在重置日志之前至少保留一个基准备份。在一段时间之后，如果没有在重置日志之前恢复数据库的潜在需要，则可以针对以前的数据库状态删除所有备份。使用REPORT OBSOLETE ORPHAN命令可以显示应用到以前数据库形态的备份。
- 在归档存储目的路径中清理归档日志文件。只要序列号增长，新的归档重做日志文件将覆盖任何以前存在的归档日志文件。可以将这些已经存在的归档日志转移到新的路径下或者删除。

技巧 在练习14.3中，用RMAN重新恢复了整个数据库并恢复了已经删除的表。在第17章中，将会看到RMAN如何执行表空间时间点恢复来完成同样的表恢复，而不需要所有的数据库进行不完全恢复。

14.3 恢复的技巧

在前面对RMAN进行还原与恢复的简要介绍中，只是讲解了RMAN可以完成的基本行为。本章中的练习则提供了简单的场景并解释了RMAN如何进行还原和恢复数据库。下面的这些恢复技巧是从其他一些方面进行考虑，或是如何在更复杂的恢复情况下发挥优势作用：

- 还原多个备份集和备份片时，当还原数据库时应增加多个通道。额外的通道将使数据文件还原以并行方式运行：

```
run
{
    allocate channel d1 type disk;
    allocate channel d2 type disk;
    allocate channel d3 type disk;
    restore database;
    recover database;
}
```

- 如果丢失了数据文件并需要从映像拷贝中快速恢复，则可以采用如下方式通知RMAN切换到新的数据文件拷贝上（也许需要这个数据文件的恢复结果）：

```
run
{
    allocate channel d1 type disk;
    sql 'alter database datafile 5 offline';
    switch datafile 5 to datafilecopy
'/oradata/PRACTICE/tools01.dbf';
    recover datafile 5;
    sql 'alter database datafile 5 online';
}
```

- 如果丢失了数据库服务器上的文件系统，则需要从原始的位置向其他位置还原数据文件以恢复数据库。在还原和恢复丢失的数据文件的执行工作中可以使用SET NEWNAME属性，同时也可以使用SWITCH属性通知目标控制文件来使用新位置下已经还原的数据文件。可以按如下方式在新的路径中还原与恢复数据文件：

```
run
{
    allocate channel d1 type disk;
    set newname for datafile 5 to '/u01/PRACTICE/tools01.dbf';
    sql 'alter database datafile 5 offline';
    restore datafile 5;
    switch datafile 5 to datafilecopy
'/u01/PRACTICE/tools01.dbf';
    recover datafile 5;
    sql 'alter database datafile 5 online';
}
```

- 如果丢失了当前控制文件的所有拷贝，但还有数据库当前日期的目录，则使用RESTORE CONTROLFILE命令重新生成控制文件，该控制文件与以前的备份类似。RMAN将自动把控制文件存放在init.ora文件指定的路径下。还原与恢复之后，必须重置日志并打开数据库。可以按如下方式还原控制文件：

```
run
{
    allocate channel d1 type disk;
```

```
restore controlfile;
}
```

- 可以从RMAN备份中还原归档日志文件，并手工在SQL*Plus中进行恢复操作。
- 默认情况下，RMAN将归档日志文件还原到归档转储目的路径下。可以如下所示，通过SET ARCHIVELOG DESTINATION命令在其他路径下还原这些文件：

```
run {
  set archivelog destination to '/tmp';
  allocate channel d1 type disk;
  restore archivelog from logseq 2 until logseq 4;
}
```

- 默认情况下，RMAN使用归档转储目的路径下的归档日志文件来恢复数据文件。如果通知RMAN在运行期间在其他路径下还原归档文件，则RMAN将在恢复期间自动查找还原的归档日志文件并应用。
- 可以像第12章中对备份性能进行跟踪一样，对还原与恢复的速度进行跟踪（V\$SESSION_LONGOPS、V\$PROCESS以及V\$SESSION）。

14.4 疑难解答

如果用RMAN进行还原与恢复时遇到了问题，可以参照下列错误信息及注解：

1) RMAN-06004: ORACLE error from recovery catalog database:

RMAN-20003: target database incarnation not found in recovery catalog

在SQL*Plus中以重置日志选项打开目标数据库之后，目录中并没有包含该数据库新形态的信息。当连接目标数据库与目录数据库时应在RMAN中发布RESET DATABASE命令。用LIST INCARNATION命令检查是否进行了新形态更新工作。

2) ORA-00376: file 3 cannot be read at this time

ORA-01110: data file 3: '/oradata/PRACTICE/users01.dbf'

数据文件被还原与恢复后，在被读取之前必须设置为联机状态。当试图访问存储在users01.dbf文件中的一个表的数据时可能会遇到该错误信息。

3) PLS-00553: character set name is not recognized

RMAN-06031: could not translate database keyword

当恢复数据库时，会话的字符集必须与目标数据库的字符集匹配。为了获得目标数据库的字符集，从V\$NLS_PARAMETERS中查询目标数据库的字符集(SELECT * FROM V\$NLS_PARAMETERS WHERE PARAMETER = 'NLS_CHARACTERSET')。退出RMAN并设置NLS_LANG环境变量为该值。例如，可以在Linux上为美国设置NLS_LANG环境变量为we8iso8859p1：

```
LINUX> export NLS_LANG=american_america.we8iso8859p1
```

4) RMAN-04005: error from target database:

ORA-00604: error occurred at recursive SQL level 1

ORA-02248: invalid option for ALTER SESSION

当在RMAN提示符下进行连接时，类似NLS_LANG和NLS_DATE_FORMAT等环境变量，如

果设置不正确将会引起该错误。例如，如果在Windows系统中设置变量时使用了单引号而不是双引号，则将出现该错误。在启动RMAN之前最好检查环境变量是否设置正确。

5) RMAN-10035: exception raised in RPC:

ORA-19573: cannot obtain exclusive enqueue for datafile 3

如果试图还原一个当前处于联机状态的数据文件，会出现这个错误。在还原该数据文件之前应先将该文件脱机。如果还原系统表空间数据文件，则应关闭数据库，然后再加载数据库。

6) RMAN-11001: Oracle Error: ORA-01113: file 3 needs media recovery

当打开数据库或将数据文件联机时，该文件也许还没有完全恢复。在将数据文件联机之前请确认已经恢复了所有还原的数据文件。

14.5 小结

我希望本章突出的内容是：在典型的环境下，服务器管理的还原与恢复是非常简便的，可以让服务器来完成这项工作。只需要下达“restore”命令，RMAN就知道需要完成什么工作；只要发布“recover”命令，RMAN就完成其余工作。为了获得关于RMAN恢复的更详细的资料，请参看《Oracle 8i Recovery Manager User's Guide and Reference》的第6章。如果需要了解restore、recover以及set选项的语法规则请参看本书第10章的相关内容。

习题

请回答以下这些问题，巩固本章所学的概念和练习。

1. 什么命令能从RMAN备份中重新生成一个文件？

- A. Replace
- B. Recover
- C. Channel
- D. Restore

2. 什么类型的数据文件备份为数据文件的还原生成了一个基准？（多项选择）

- A. 映像拷贝
- B. 增量级别为0的备份
- C. 增量级别为1的备份
- D. 完全备份

3. 当恢复一个数据文件时，RMAN或许会在应用归档日志文件之前将其还原到磁盘上。

- A. True
- B. False

4. 执行下面哪条还原命令时需要将数据库关闭（假定TOOLS表空间只有一个名为tools01.dbf的数据文件，而且不包含回退段）？

- A. Arestore database
- B. Brestore tablespace tools

- C. Crestore datafile 'tools01.dbf'
- D. Drestore archivelog
- 5. 当数据文件还原时, 需要应用增量级别1-4的备份。
 - A. True
 - B. False

答案

1. D。Restore。类似于用户管理的恢复中在操作系统下向目的地拷贝备份文件的操作。

2. A、B、D。映像拷贝、增量级别为0的数据文件以及完全数据文件备份都在一个数据文件中拷贝了所有使用过的数据块。这些备份和拷贝可以被RMAN用于还原数据文件。增量级别1~4的备份可以被应用于基准备份(映像拷贝、完全备份以及增量级别为0备份), 以回退被更改的数据块。

3. True。在恢复期间, RMAN首先在归档存储路径以及SET ARCHIVELOG DESTINATION TO命令指定的路径下查找所需要的归档日志文件。如果都没有找到, 则从备份中将其还原到磁盘上, 以便恢复操作的进行。

4. A。在还原系统表空间时, 数据库必须关闭(但不是加载)。还原与恢复可以在任何非系统脱机数据文件上进行。当数据库打开和所有表空间联机时可以进行归档日志文件的还原。

5. False。在还原期间, 只有基准备份被用于还原数据文件。在恢复期间, 增量级别为1-4的备份才被重做操作用于数据文件。



第15章 RMAN 复制数据库

通过使用数据库备份，DBA可以在同一服务器或其他服务器上建立副本数据库。这个副本数据库可以和主数据库有相同的名称（数据库拷贝）或者与主数据库名称不同（数据库克隆）。Oracle在数据库备份和数据库克隆之间惟一不同的是拷贝的数据库不能更改原名称。使用RMAN的复制数据库特性，可以从RMAN备份创建一个新的数据库，并为这个副本数据库保留已有的数据库名称或者赋予新的名称。在第6章中已经介绍了复制数据库的概念。在第12章和第14章中，讲述了如何用恢复管理器（RMAN）进行备份与恢复。这一章将讲解如何综合这三章的知识，用RMAN创建副本数据库。用RMAN进行数据库复制的功能是从Oracle 8i开始引入的。

RMAN如何从备份创建一个副本数据库？在复制期间，RMAN连接主数据库和副本的实例。复制进程包括在副本服务器上还原并恢复主数据库的备份文件。首先在副本数据库服务器上恢复用RMAN创建的主数据库备份基准。在恢复期间，所需的主数据库的增量备份和归档重做日志都将被应用到副本数据库中。副本数据库不能进行完全恢复，因为联机重做日志不能还原（由于没有用RMAN进行备份）。因此，不完全恢复将执行到一个指定的停止点，或者直到RMAN所知的最新归档日志文件被应用。RMAN将不会应用任何来自联机日志的重做。

副本数据库相比主数据库而言已经过时了。在复制期间，会生成一个新的控制文件和联机重做日志文件。RMAN也将赋予这个数据库一个新的数据库标识（DBID）。

就像其他介绍RMAN的每章一样，我将定义属于本章内容的新的RMAN术语或命令。本章将介绍以下RMAN术语和命令：

- **辅助数据库（Auxiliary database）**：RMAN将目标（主）数据库复制到该数据库实例。要创建该数据库的参数文件、路径和口令文件。必须在RMAN数据库复制之前以非加载（NOMOUNT）模式启动辅助数据库实例。可以将该实例当作副本数据库实例。
- **复制（Duplicate）**：从其他数据库的RMAN备份创建一个新的数据库。要在RMAN执行复制操作的位置配置辅助数据库并启动Oracle的实例。从RMAN的角度来看，目标数据库被复制到副本数据库。
- **设置新名称（Set newname）**：在一个RMAN运行块内为数据文件设置新的名称。提供给该参数的文件名将覆盖任何该数据文件的辅助数据库名（用SET_AUXNAME）或者辅助数据库参数（DB_FILE_NAME_CONVERT）。这个新名称的值只在运行块内有效。
- **设置辅助名称（Set auxname）**：为数据文件设置辅助名称。这个辅助名称将在RMAN会话之间存在。如果不希望以后的RMAN使用这个设置的名称，则必须将该名称设置为null。
- **日志文件（log file）**：可以在duplicate命令中使用这个关键字，以指定副本数据库创建的联机重做日志文件。如果没有特别指定这个关键字，则RMAN将在辅助参数文件中的LOG_FILE_NAME_CONVERT参数决定的路径中生成日志文件。如果没有这个RMAN关键字，而且也没有设置辅助参数，则RMAN将在目标数据库同样的位置创建日志文件（假设已经指定了NOCHECKFILENAME选项）。
- **不检查文件名（Nocheckfilename）**：默认情况下，RMAN将检查在副本主机上被恢复到主

目标数据文件路径下的数据文件，以确保不会被错误地覆盖。使用该选项就可以覆盖这个默认的操作。这样就需要手工确保RMAN没有覆盖任何已经存在的数据文件。应当谨慎使用这个命令，以防止覆盖数据文件。

在duplication命令执行的过程中，RMAN执行了一系列工作。当连接到目标、辅助和可选的目录数据库后，RMAN将进行以下操作：

- 1) 根据最近发生的或者是提供的恢复停止点来决定复制操作将使用哪个基本的备份。
- 2) 根据辅助数据库参数或RMAN设置的命令和选项来决定将数据文件保存在辅助数据库实例的什么位置。
- 3) 为辅助数据库读出备份片或映像拷贝并恢复数据文件。这个RMAN的功能与执行正常的数据库还原是一样的。
- 4) 根据恢复停止点将任何增量备份应用于还原数据文件。这个增量方式的应用与用RMAN发布恢复数据库命令的任务是一样的。
- 5) 根据恢复停止点从磁盘或备份将所有归档日志文件应用于还原数据文件。
- 6) 为辅助数据库创建新的控制文件。
- 7) 当重新设置联机重做日志文件时，打开副本数据库。新的联机重做日志文件将根据RMAN复制数据库命令中指定的或者根据转换的辅助参数文件进行创建。

为什么要使用RMAN来创建副本数据库呢？如果备份只是RMAN备份则必须使用RMAN进行操作。如果既有RMAN备份也有用户管理的备份，则RMAN进行复制会比较容易。RMAN将创建新的控制文件并处理数据文件的还原与恢复。

当用RMAN进行数据库复制时，应该考虑以下注意事项：

- 不是必须要使用RMAN目录。RMAN信息被存储在目标数据库的控制文件中。必须确保要创建副本数据库的备份仍然在目标控制文件中。例如，如果将init.ora文件的CONTROLFILE_RECORD_KEEP_FOR_TIME参数设置为21天，则不能从2个月以前开始还原。
- 如果当数据库打开时进行备份，则应包括随后的日志切换以确保当备份进行时的任何重做的更改都被归档。
- 确保选取的备份包含了必要的归档日志。RMAN必须清楚应用的归档日志。
- 不能在复制期间进行完全恢复，恢复必须有停止点。该停止点可以是一个指定的停止点也可以是归档重做日志所知的最后一个点。可以基于时间、日志序列或者SCN来指定一个停止点。
- 为了在复制期间创建数据库的拷贝，应使用与辅助数据库参数文件中的目标文件一样的数据库名称。
- 如果副本数据库驻留在其他服务器上，而且服务器拥有与主数据库服务器同样的路径结构，则在复制期间不需要重新命名任何文件。默认情况下，DUPLICATE命令将告知RMAN还原的辅助文件的位置与目标数据库文件备份时的主数据库文件位置相同。
- 新的副本数据库的数据库名称可以和主数据库的相同，也可以不同。但数据库的名称必须与任何共享ORACLE_HOME路径的数据库的名称不同。因此，如果主数据库和副本数据库驻留在同一服务器上，而且共享相同的ORACLE_HOME，则副本数据库必须重新命名。

如果虽然都在同一服务器上，但是拥有独立的ORACLE_HOME路径或在不同的服务器上，则它们可以具有相同的数据库名称。

在本章的练习中，将要用RMAN创建一个副本数据库。可以通过辅助初始化参数文件以及RMAN命令更改数据文件的名称和位置。图15-1表示的是本章练习将要执行的RMAN复制操作。

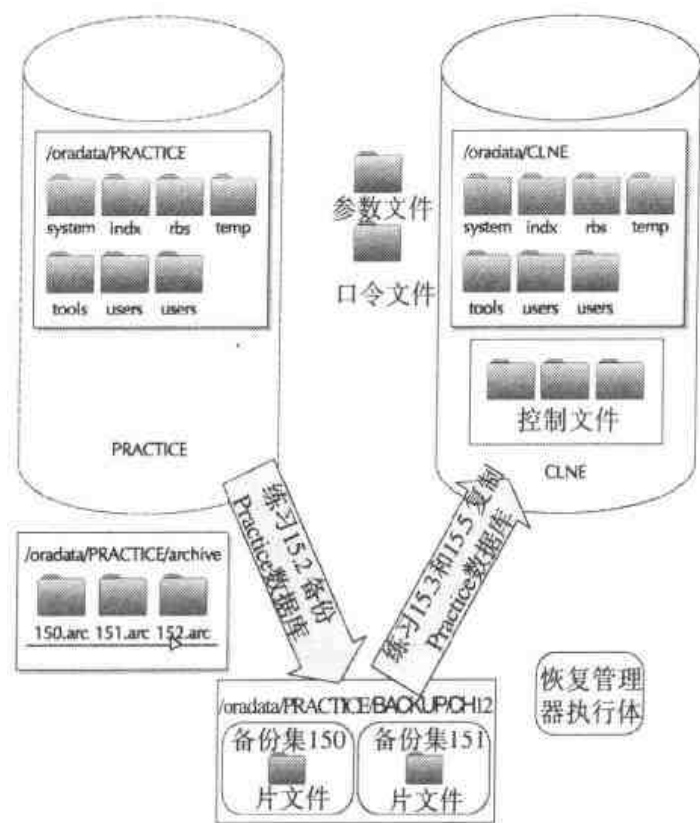


图15-1 用RMAN复制PRACTICE数据库

15.1 恢复管理器复制

创建副本数据库涉及一系列相互协调的步骤。在下列描述的步骤中，主数据库称为目标数据库，而副本数据库称为辅助数据库。

- 1) 使RMAN备份可以被将要存储副本数据库的服务器访问 如果备份生成在磁盘上，可以用ftp的二进制模式传送，或用NFS这样的磁盘共享机制进行访问。
- 2) 准备副本数据库参数文件 在副本数据库被创建以前，RMAN需要连接到一个辅助实例上。这类似于第9章中要执行的表空间时间点恢复时需要连接的辅助实例。RMAN连接到辅助实例后，则从备份开始还原数据文件以生成副本数据库。为了启动辅助实例，必须创建一个包含许多特殊参数的初始化参数文件。当在与主数据库同样的服务器上创建副本数据库时，必须确保控制文件、数据文件和联机重做日志没有覆盖已经存在的文件。为了达到这个目的，可以告知RMAN在执行DUPLICATE命令时在新的位置重新命名这些文件，或者可以指定某些特殊的初始化参数自动完成这个工作。在本章的练习中将会看到这些方法是如何工作的。

- 3) 生成副本数据库口令文件 为了进行DBA验证,副本数据库需要一个口令文件,可以按照第6章练习6.1任务1的步骤生成这个文件。
- 4) 创建Net8连接文件 RMAN需要同时连接到主数据库(目标实例)以及副本数据库(辅助实例)。如果这两个数据库驻留在不同的服务器上,则需要配置副本服务器上的listener监听器,而且需要配置运行RMAN的服务器(通常是主数据库服务器)上的tnsnames文件。
- 5) 启动辅助实例 在RMAN创建副本数据库以前,辅助实例需要以非加载模式被启动。启动时将读取第2步创建的参数文件。
- 6) 加载或打开主数据库 RMAN需要主数据库加载或打开以便创建副本数据库。
- 7) 创建副本数据库 当要求RMAN创建副本数据库时,它会自动还原所需的基准备份,并应用增量备份和归档重做日志文件。恢复完成后,副本数据库会以RESETLOGS选项打开。

本章中的练习将上述一些步骤合并为单一的任务。现在练习RMAN数据库的复制操作。

练习15.1: 用RMAN复制PRACTICE数据库

本章只有一个练习,主要是复制PRACTICE数据库。将要克隆生成的数据库名为CLNE(Clone的简写)。因此,在本练习中,PRACTICE数据库就是主/目标数据库。副本数据库是一个克隆,因为该数据库被赋予了一个新的名称。如果在其他服务器上执行该练习,则创建的数据库可以具有相同的名称(PRACTICE),只需要将本练习中的CLNE用PRACTICE代替就可以了。

任务描述	时间(分钟)
1. 准备克隆数据库	10
2. 备份Practice数据库	10
3. 复制Practice数据库	10
4. 验证克隆数据库	5
5. 再次复制Practice数据库	10
总计时间	45

尽管RMAN将完成数据库复制工作的大部分工作,仍然需要我们准备路径和必要的数据库文件。

任务1: 准备克隆数据库

为CLNE数据库开始创建数据库路径、参数文件、口令文件(及Windows服务)。如果已经执行了第6章的练习,这些步骤就都已经完成了。

• 路径 为将要生成的CLNE数据库创建所有相关路径:

```
export ORACLE_SID=CLNE
export ORACLE_BASE=/app/oracle
export ORACLE_HOME=/app/oracle/product/8.1.7
export ORACLE_DATA=/oradata/$ORACLE_SID
export ORACLE_ADMIN=$ORACLE_BASE/admin/$ORACLE_SID
mkdir $ORACLE_ADMIN
```



```
mkdir $ORACLE_ADMIN/pfile
mkdir $ORACLE_ADMIN/bdump
mkdir $ORACLE_ADMIN/cdump
mkdir $ORACLE_ADMIN/udump
mkdir $ORACLE_ADMIN/create
mkdir $ORACLE_DATA/
mkdir $ORACLE_DATA/archive
```

- **参数文件** 接下来，通过将PRACTICE数据的参数文件拷贝到\$ORACLE_BASE/admin/CLNE/pfile目录下为CLNE数据库创建参数文件。再在\$ORACLE_HOME/dbs目录下增加默认参数文件的Linux符号连接，或者在%ORACLE_HOME%\database目录下增加Windows参数文件。

打开pfile目录下的新的参数文件，并将其中所有的“PRACTICE”修改为“CLNE”。当执行这个全文搜索和替换时，就为参数更改了数据库名、服务名、控制文件的位置以及许多路径。应确认修改了以下参数：DB_NAME, INSTANCE_NAME, SERVICE_NAMES, CONTROL_FILES以及目的路径。在CLNE数据库参数文件中增加两个参数：

```
db_file_name_convert = ("PRACTICE", "CLNE")
log_file_name_convert = ("PRACTICE", "CLNE")
```

在复制期间，这两个参数将把文件名和位置名从旧值转换为新值。在这个示例中，任何数据文件全名称中的PRACTICE都将被改为CLNE。因此，PRACTICE数据库文件/oradata/PRACTICE/system01.dbf将在CLNE中被创建为/oradata/CLNE/system01.dbf。日志文件/oradata/PRACTICE/redo01.log将在CLNE数据库中被创建为/oradata/CLNE/redo01.log。

- **口令文件** 在RMAN复制期间，需要以SYSDBA角色的身份连接到CLNE数据库。创建口令文件。这个口令文件也将用于远程身份认证安全。

```
LINUX> orapwd file=$ORACLE_HOME/dbs/orapwCLNE password=CLNE entries=4
WINNT> orapwd file=%ORACLE_HOME%\database\PWDCLNE.ora password=CLNE
entries=10
```

- **Windows服务** 在Windows中，用oradim工具为CLNE数据库创建一个新的名为OracleServiceCLNE的数据库服务：

```
WINNT> oradim -new -sid CLNE -intpwd CLNE
```

- **Net8连接** 最后，为CLNE数据库配置Net8连接。这个步骤对于第6章中复制数据库是没有必要的，因为在连接到PRACTICE数据库的同时，不需要连接到CLNE实例。应用Net8 Assistant（在Linux上是netasst或者在Windows中的start菜单中的快捷方式）来更新Net8的配置。

如果已经创建了一个在第6章中描述的克隆数据库，则没有必要执行所有这些步骤（除了为CLNE参数文件增加两个转换参数之外）。数据库文件已经存在于/oradata/CLNE目录下。可以保留这些以前创建的数据库文件（数据文件、日志文件和控制文件），也可以删除。如果要删除，就要将/oradata/CLNE目录下的所有文件都删除。

任务2：备份Practice数据库

现在，可能已经有了足够多的PRACTICE数据库的备份可以用来复制到CLNE数据库。为了

确保成功，继续创建两个用于复制的PRACTICE数据库备份。

第一个备份是用第12章的b_whole_0.rcv命令脚本创建的基准备份（增量级为0）。数据文件和控制文件的基准备份也是数据库中所有数据文件和控制文件的完全数据库备份或映像拷贝。没有基准备份的增量级为1-4备份对于数据库复制是不充分的。

第二个备份在基本数据文件和控制文件备份之后进行，是一个归档日志文件备份。可以运行第12章的名为b_archive_2days.rcv的RMAN命令文件来生成这个备份。该文件中包含的命令将强迫进行一次日志切换，并备份所有最近两天以来的含有重做信息的归档日志文件：

```
LINUX> rman
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @b_whole_inc0.rcv
RMAN> @b_archive_2days.rcv
```

标记PRACTICE数据库当前的日志文件。在复制期间，我将使用这个日志文件作为我的停止点。

```
SQL> connect sys/practice@practice
SQL> SELECT sequence#, thread# FROM v$log WHERE status = 'CURRENT';
```

任务3：复制Practice数据库

在这个任务中，RMAN将对几个命令响应并执行一系列操作。所有需要的命令如下所示：

```
connect target sys/practice@practice
connect catalog rman817/rman@rcat
connect auxiliary sys/clone@clne
# Duplicate the PRACTICE database to the CLNE database
run {
  set until logseq 151 thread 1;
  allocate auxiliary channel dl type disk;
  duplicate target database to CLNE;
}
```

下面将解释这里罗列的命令。这个RMAN脚本文件包含连接到目标、目录和辅助数据库的命令。在复制期间，需要目标数据库和辅助数据库的连接。而目录的连接则是可选的。

- **connect target** 要被复制的数据库是目标数据库连接。这个数据库应该被加载或者打开。
- **connect catalog** 如果需要目录，则应该包含所有目标数据库备份的历史。RMAN可以使用合适的备份来还原副本数据库。这个数据库必须被打开。
- **connect auxiliary** 副本数据库是辅助数据库连接。这个数据库必须以非加载模式启动。
- **set until** 因为在目标数据库被打开的情况下复制数据库一定是不完全的，所以可以使用一个比最终归档的重做日志文件的序列号大的日志序列停止点。如果目标数据库上的当前日志文件是151，则恢复到辅助数据库的将包括日志序列150而不是151。必须包括thread的值，但该值并不属于一个Oracle并行服务器的配置。选择的停止点必须在时间点上要大于RMAN能从其中恢复的最早备份时间。
- **allocate auxiliary channel** 在辅助数据库上的操作都是通过辅助通道执行的。目标数据

库的备份片将通过这个通道读取，并还原到CLNE数据库中。CLNE数据库的恢复也是通过这个通道执行的。

- **duplicate target** 目标数据库将被复制到名为CLNE的辅助数据库。复制命令从目标数据库备份还原数据文件到辅助数据库的新位置。还原之后，RMAN将生成新的控制文件，然后恢复数据库。最后，在重新设置日志文件时辅助数据库将被打开。

DUPLICATE命令的输出比较长，所以这里并没有全部显示这些输出。然而，我确实希望能指出一些重要的地方，以便说明RMAN所做的工作：

```
RMAN-08500: channel dl: sid=13 devtype=DISK
```

在复制期间，辅助通道连接到CLNE实例上。当复制过程进行时，可以使用会话标识（sid，不要与系统标识SID混淆）在SQL*Plus中连接到CLNE实例，并查看RMAN连接。

```
RMAN-03027: printing stored script: Memory Script
{
    set until scn 598838;
    set newname for datafile 1 to "/oradata/CLNE/system01.dbf";
    ...
    restore
    check readonly
    clone database;
}
```

复制命令将使RMAN脚本执行。RMAN可以选择使用在内存中动态生成的脚本。因为这些脚本没有在什么地方静态存储，这就是说复制命令有没有目录也可以工作。在第一个内存脚本中，RMAN确定PRACTICE数据库的日志序列151是从SCN598838开始，并将其设置为复制进程的停止点。然后，RMAN还原在这个SCN值之前更适合的基准备份。RMAN将用增量备份和归档重做恢复到这个SCN值。类似于正常的RMAN恢复，RMAN首先在辅助数据库的database_archive_log_dest路径下查找所需要的归档日志文件是否已经存在。如果不存在，则必须被还原并从一个备份中应用。其次，RMAN基于CLNE数据库的init.ora参数文件中的DB_FILE_NAME_CONVERT参数决定数据文件的新的名称。所有遇到的PRACTICE都将被替换为CLNE，当还原数据文件时，有效地为每个数据文件改变路径。最后，向克隆数据库发布restore命令。RMAN生成的脚本中的关键字clone将被写到CLNE辅助数据库。如果已经连接到了一个名为TST1的辅助数据库，则这个命令将仍然是RESTORE CLONE DATABASE。

```
RMAN-08502: set_count=153 set_stamp=449676306 creation_time=15-JAN-02
RMAN-08089: channel dl: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00001 to /oradata/CLNE/system01.dbf
RMAN-08523: restoring datafile 00002 to /oradata/CLNE/rbs01.dbf
...
RMAN-08511: piece handle=
/oradata/PRACTICE/backup/ch12/db_PRACTICE_102_1_44966306 tag=WHOLE_INC0
```

还原命令将发现在任务2中生成的备份，并使用这个备份从目标数据库（PRACTICE）为辅助数据库（CLNE）还原数据文件。数据文件是从备份中还原的，而不是从目标数据库拷贝。因此，将很少影响目标数据库的性能。

```

RMAN-08024: channel dl: restore complete
RMAN-06162: sql statement: CREATE CONTROLFILE REUSE SET DATABASE CLNE ...
LOGFILE
  GROUP 1 ( '/oradata/CLNE/redo01.log' ) SIZE 1048576 REUSE,
  GROUP 2 ( '/oradata/CLNE/redo02.log' ) SIZE 1048576 REUSE,
  GROUP 3 ( '/oradata/CLNE/redo03.log' ) SIZE 1048576 REUSE
DATAFILE ...

```

当还原完成之后，RMAN创建一个新的控制文件。在控制文件的创建命令中，数据名称被设置为CLNE，而不是PRACTICE。因为在任务1中设置了CLNE数据库init.ora的LOG_FILE_NAME_CONVERT参数，所以这个日志文件的位置将从/oradata/PRACTICE更改为/oradata/CLNE。

```
switch clone datafile all;
```

所有克隆数据库（CLNE）中的非系统数据文件必须从目标数据库的位置切换到新的数据库的相应位置。

```

RMAN-03027: printing stored script: Memory Script
{
  set until scn 598838;
  recover
  clone database
  check readonly
;
}
RMAN-03021: executing script: Memory Script

```

数据文件已经被还原到CLNE数据库的位置中而且已经创建了新的CLNE控制文件。接下来，RMAN使用首次增量备份和归档重做日志文件恢复在克隆数据库上的已经还原的数据库文件。因为在本练习中没有增量备份，所以不会看到任何增量备份被还原和应用。应该在RMAN恢复的第四阶段看到使用PRACTICE数据库的归档日志文件进行介质恢复。

```

RMAN-08054: starting media recovery
RMAN-03022: compiling command: recover(4)
...
RMAN-03023: executing command: recover(4)
RMAN-08515: archivelog filename=/oradata/PRACTICE/archive/149.arc
thread=1 sequence=149
RMAN-08515: archivelog filename=/oradata/PRACTICE/archive/150.arc
thread=1 sequence=150
RMAN-08055: media recovery complete

```

在介质恢复后，用resetlogs选项将CLNE数据库打开：

```

Alter clone database open resetlogs;
...
RMAN-06400: database opened

```

如果在整个复制命令脚本的输出中，没有遇到错误信息，而且看到消息RMAN-06400: database opened message，则RMAN数据库复制操作就成功了！

任务4：验证克隆数据库

为了检查CLNE数据库是否已经创建成功，通过SQL*Plus连接，并运行一些查询语句。SYS口令（没有SYSDBA角色）将与PRACTICE数据库上的SYS口令一样，因为CLNE数据库就是PRACTICE数据库的副本。

```
SQL> CONNECT sys/practice@clone
SQL> SELECT sequence#, first_change#, status FROM v$log ORDER BY
status;
SEQUENCE# FIRST_CHANGE# STATUS
-----
1          598839 CURRENT
0              0 UNUSED
0              0 UNUSED
```

当前联机重做日志文件的序列号是1！当CLNE数据库以resetlogs打开时，日志序列号被重置为1。这个日志序列号的首先改变为598839。当数据库被打开并且重置重做日志，则当前重做日志SCN的第一个改变是比恢复结束SCN改变的值大1。在恢复期间，所有数据文件都向前回退到SCN 598838。数据库打开的重置日志操作发生在SCN 598839位置。

```
SQL> SELECT instance_name FROM v$instance;
SQL> SELECT name FROM v$database;
SQL> SELECT name FROM v$datafile;
SQL> SELECT member FROM v$logfile;
SQL> SELECT name FROM v$controlfile;
SQL> SELECT max(create_date) FROM tina.date_log;
```

通过运行这些查询，可以看到数据文件的位置，也可以检查一个数据库表的数据。感觉到可以随意在/oradata/CLNE目录下查看由RMAN复制操作创建的数据文件。因为RMAN自动赋予副本数据库一个新的DBID，所以可以将这个新的数据库在恢复目录中注册并开始备份。即使副本数据库与主数据库拥有同样的名称，但DBID却不同，所以也可以在恢复目录中注册。

任务5：再次复制Practice数据库

最后的这个任务将再次复制数据库，但是却使用RMAN的关键字和命令来控制数据文件和日志文件的名称和位置。（控制文件的位置是由辅助数据库文件参数文件的control_file设置的。）可以如下控制数据文件的位置并设置新名称或设置辅助名称。

```
set auxname for datafile 2 TO '/oradata/CLNE/auxname02.dbf';
set auxname for datafile 4 TO '/oradata/CLNE/auxname04.dbf';
set auxname for datafile 6 TO '/oradata/CLNE/auxname06.dbf';
run {
    allocate auxiliary channel dl type disk;
    set until logseq 152 thread 1;
    set newname for datafile 1 TO '/oradata/CLNE/newname01.dbf';
    set newname for datafile 3 TO '/oradata/CLNE/newname03.dbf';
    set newname for datafile 5 TO '/oradata/CLNE/newname05.dbf';
    set newname for datafile 7 TO '/oradata/CLNE/newname07.dbf';
    duplicate target database to CLNE logfile
        group 1 ('/oradata/CLNE/redo01_1.log',
```

```

        '/oradata/CLNE/redo01_2.log') size 500K reuse,
group 2 ('/oradata/CLNE/redo02_1.log',
        '/oradata/CLNE/redo02_2.log') size 500K reuse,
group 3 ('/oradata/CLNE/redo03_1.log',
        '/oradata/CLNE/redo03_2.log') size 500K reuse;
}
set auxname for datafile 2 TO null;
set auxname for datafile 4 TO null;
set auxname for datafile 6 TO null;

```

以上显示如何在复制期间为CLNE数据库的每个数据文件指定一个新的文件名（或位置）。这样，当将PRACTICE备份还原到CLNE的位置时，每个文件都可以被重新命名。日志文件可以指定为附加到DUPLICATE TARGET DATABASE命令。在以上的示例中，每个数据文件都被重命名，并且日志文件也被重新定义大小、重命名和镜像。为了说明问题，对单数的数据文件重命名而对偶数的数据文件辅助命名。同时，我对不完全恢复增加了一个set until语句。这个复制将使用基准复制，并应用直到日志序列152的重做，但不包括日志序列152。

可以用以上显示的脚本创建备份。确保关闭了CLNE数据库，并在运行这个复制脚本之前以非加载方式打开实例。（在Windows中，在启动CLNE实例之前先关闭Windows服务。如果没有这样做，则控制文件仍然保持加载状态。）尽管在任务1的initCLNE.ora文件后面增加了数据文件和日志文件的转换参数，但通过RMAN set命令的数据文件和日志文件名将覆盖这个init.ora中的参数。

一旦通过运行这些脚本中的一个（或二者都运行）来进行数据库复制工作，请查看任务4中描述的数据文件和日志文件的位置。在/oradata/CLNE下将会看到不同的文件名。这就证实了可以通过RMAN命令和关键字控制副本数据库的文件名和位置。

注意 如果在同样的脚本中指定了auxname和newname，则newname将优先于auxname。

技巧 将每个auxname重置为null以便以后的RMAN命令不会错误地使用同样的名称。例如，在本章中将创建/oradata/CLNE/auxname1.dbf作为数据文件1。而在下一章中，数据文件1将仍然是/oradata/CLNE/auxname1.dbf，除非为这个文件重置辅助名称或者用set newname进行覆盖。

15.2 在不同的服务器上复制数据库

- 很多情况下，复制操作将发生在与主数据库服务器不同的其他服务器上。那么如何处理？
- 如果目标数据库的RMAN备份已经保存在磁盘上，那么，备份片必须被拷贝到复制的服务器上。例如，如果在主机hostA上有一个RMAN备份的备份片文件/oradata/df_100_12345_1，则这个文件必须被拷贝到主机hostB上的相同位置上。这样，RMAN才能够将这个备份的片段用于还原任务。
- 如果目标的RMAN备份已经保存在磁带上，则磁带供应商的软件和磁带的运行库必须对拥有辅助数据库的服务器可用。
- 如果克隆的数据需要与主数据库同样的目录路径结构时，应该增加NOCHECKFILENAME

选项。这样可以阻止RMAN检查还原数据文件的存在，并将覆盖任何已经存在的具有相同名称的文件。

15.3 疑难解答

RMAN期望在数据库复制之前能先完成以下几件事情。目标数据库必须被打开或加载。

- RMAN会话的环境变量必须被设置为与目标数据库相同的NLS语言设置。
- PRACTICE数据库必须进行整体数据库备份，并紧跟着包含一次日志切换的归档日志备份。只有当在复制的数据库归档日志目的路径下没有找到恢复所需要的归档日志时，才需要归档日志备份。如果主数据库的归档日志已经被拷贝到副本的服务器上，则不需要归档备份。
- 辅助的实例（CLNE）必须以非加载模式启动。

如果用RMAN进行创建副本数据库遇到了问题，有可能是以下某些错误：

- 1) RMAN-05001: auxiliary filename /oradata/PRACTICE/tools01.dbf conflicts with a file used by the target database 如果数据文件和日志文件在备份被还原时存在，则RMAN默认的复制特性将阻止覆盖这些文件。如果得到这个错误，应确认已经用辅助的参数设置或RMAN命令重新命名了数据文件。如果期望在不同服务器上的同样的位置重新创建辅助数据库的数据文件，则应在复制命令中使用NOCHECKFILENAME选项。这样，RMAN就不会在覆盖已经存在的文件时报告错误。
- 2) PLS-00553: character set name is not recognized 如果在复制期间RMAN客户端的属性设置与目标数据库的设置不同，则RMAN将显示这个错误。请根据操作系统使用以下操作来修改客户端的属性设置：

```
LINUX > export NLS_LANG=AMERICAN.WE8ISO8859P1
WINNT> set NLS_LANG=AMERICAN.WE8ISO8859P1
```

- 3) RMAN-08060: unable to find archive log

RMAN-06054: media recovery requesting unknown log: thread 1 scn 598778

RMAN复制不能执行完全恢复。使用set until运行选项或者在目标数据库上通过切换联机日志文件使归档日志可用，并且拷贝这些文件到副本数据库或用RMAN备份到副本数据库中，以便为最后归档的日志文件指定停止点。

- 4) ORA-12571: TNS:packet writer failure 为了完成复制任务，必须启动辅助的实例。同时，应确保在RMAN中连接到该实例时没有关闭数据库（Windows中是数据库服务）。
- 5) RMAN-06136: ORACLE error from auxiliary database:

ORA-01503: CREATE CONTROLFILE failed

ORA-01158: database already mounted

RMAN-06097: text of failing SQL statement: CREATE CONTROLFILE

REUSE SET DATABASE CLNE RESETLOGS ARCHIVELOG 如果在复制完成之前，控制文件生成了，但却遇到了不同的失效，则Windows可能会提示错误，认为数据库已经加载了控制文件。关闭Windows服务，重新启动并确认CLNE实例已经启动但没有加载，

然后再次运行复制脚本。

- 6) RMAN-06023: no backup or copy of datafile 1 found to restore 使用了set until值来指定时间点恢复, 但指定的该点却超出了当前的备份或数据库的范围, 这时会出现这个错误。例如, 设置备份的停止点比数据库当前日志序列号的值大。如果自从用于复制的主数据库被备份以来, 在主数据库上没有执行日志切换, 也会遇到这个错误。

15.4 小结

现在可以将本章与第6章进行比较。用户管理的数据库复制比RMAN执行的复制要多做许多工作。用户管理和服务器管理的复制都需要建立数据库路径和支撑文件。虽然两者都需要数据库生成的备份, 但是在服务器管理的备份中, RMAN将完成这个工作。表15-1总结了用户管理的复制和服务器管理的复制之间的差别。

在本章中介绍了如何使用RMAN复制数据库, 讲述了如何在副本数据库中控制数据文件和日志文件的位置。我希望读者成功地创建了PRACTICE数据库的副本数据库。在第16章和第17章中将使用一个辅助数据库。为了更多地了解RMAN复制, 请参看《RMAN Oracle8i Recovery Manager User's Guide and Reference》中第7章的“Creating a Duplicate Database with Recovery Manager”。

表15-1 比较用户管理的复制和服务器管理的复制

用户管理的复制	服务器管理的复制
配置副本数据库(路径、参数文件、口令文件、Windows服务); Net8配置可选	配置副本数据库(路径、参数文件、口令文件、Windows服务); Net8配置可选
采用用户管理备份主数据库打开或关闭备份。备份需要与所有数据文件一样多的磁盘或磁带空间	采用服务器管理备份目标数据库打开、关闭或映像拷贝, 备份需要的磁盘或磁带空间比所有的数据文件的空间少。映像拷贝需要与所有数据文件一样多的磁盘或磁带空间
通过操作系统的命令删除备份文件	只有在副本数据库保存在其他机器上且备份存储在主数据库磁盘或副本服务器上的磁带系统不可用时, 才通过操作系统命令删除文件
用备份控制文件跟踪并编辑该文件, 从而生成控制文件脚本。运行该脚本创建新的控制文件	当连接到目标、目录或者辅助数据库时发布复制数据库命令
必须执行手工不完全恢复并以RESETLOGS打开可以进行完全恢复	RMAN执行不完全恢复并打开重置日志不可能进行完全恢复

习题

请回答以下问题, 巩固本章的概念和练习。

1. 什么RMAN命令复制一个已经通过RMAN备份的数据库?
 - A. clone target database
 - B. duplicate target database

- C. clone auxiliary database
- D. duplicate auxiliary database
- 2. 辅助数据库的备份被复制到目标数据库。
 - A. True
 - B. false
- 3. RMAN的duplicate database命令完成什么工作? (多项选择)
 - A. 从RMAN备份还原数据文件
 - B. 用增量备份和重做恢复数据文件
 - C. 生成新的控制文件
 - D. 生成新的联机重做日志文件
- 4. 在RMAN复制期间, 如何控制本地的数据文件? (多项选择)
 - A. DB_FILE_NAME_CONVERT初始化参数
 - B. SET NEWNAME RMAN关键字
 - C. SET AUXNAME RMAN命令
 - D. DUPLICATE ... LOGFILE命令
- 5. RMAN复制只能在辅助数据库上执行不完全恢复。
 - A. True
 - B. False

答案

1. B。duplicate target database命令是通知RMAN从目标数据库备份向辅助数据库创建一个副本数据库的命令。

2. False。尽管目标数据库是RMAN还原和恢复操作的主要集中点, 但复制命令用目标数据库作为源而辅助数据库作为目的。

3. A、B、C、D。RMAN复制可以完成所有这些工作。

4. A、B、C。因为RMAN复制命名数据文件, 它首先是为数据文件寻找新名称设置。然后, 为数据文件寻找辅助名称设置。如果都没有找到, 则在辅助参数文件中寻找转换匹配的名称。如果没有指定名称更改, 则还原的数据文件将拥有与目标备份相同的名称和路径。

5. True。RMAN复制将不访问主/目标数据库的联机日志。因此, 在副本/辅助数据库上的恢复一定是不完全的。



第16章 RMAN 备用数据库

在第7章中，我们学习了如何应用用户管理技术实现一个备用数据库。我们知道，备用数据库就是主数据库的一个拷贝，这与上一章所讲的副本数据库类似。备用数据库与主数据库一模一样，包括数据库的名称。当在原始数据库生成了归档日志文件时，这些文件也被传送并应用到备用数据库上。如果主数据库失效而且不能很快解决问题，或者要求有立即的失效恢复解决方案，则备用数据库就可以被激活。如果发生这样的情况，则备用数据库就变为主数据库，所有用户都将连接到这个新的主数据库上，同时需要立即创建一个新的备用数据库。如果没有这样做，那么新的主数据库失效时，就没有用来迁移所有用户的失效解决方案。

非常类似于在第15章中用RMAN创建副本数据库的方法，RMAN也可以用于创建备用数据库。实际上，这些步骤几乎都一样，只是几个重要的地方有差别，这些差别在进行本章的练习时就会一目了然。一旦创建了备用数据库，对它的维护（传送以及应用归档日志文件）就和第7章中讲述的完全相同。用RMAN的DUPLICATE创建备用数据库是在Oracle 8i中首次引入的。因此，不可能在低于8i版本的数据库上用RMAN创建备用数据库。

阅读本章的前提条件是先通读第7章，在第7章中有相关概念和维护备用数据库的所有基本信息。在本章中，我将讲述如何使用RMAN创建备用数据库并备份该数据库。类似于数据库复制，主数据库被当作目标数据库，备用数据库作为辅助数据库。

数据库的替补

设想自己是一部本地社区戏《The Wizard of Oz》的导演。经过几个星期的准备、布景、收集道具、彩排以及售票工作，终于到了演出的时候。在第2幕第2场，Dorothy（女主角）刚说到：“Toto，我有一种感觉，我们并没有在堪萨斯”，然后就昏倒了。可能是由于紧张或疲劳，但无论什么原因，这部戏中止了，所有的人都在看着你——导演，等待着你的指挥。幸好，你已经对这种情况早有准备。一眨眼的工夫，你做手势让替补演员上台来代替主角。替角牵着狗Toto，接着说：“Toto，我有一种感觉，我们并没有在堪萨斯”，而此时昏倒的Dorothy已经被舞台助理抬到了一边。在彩排时，替角就准备着在这种紧急情况下代替女主角。备用数据库就类似于戏中的替角：随时准备在紧急情况下登上舞台代替主角。

在本章中，我将介绍以下几个新的RMAN命令。前面几章已经讲解的命令仍然适用。

- **INCLUDE CURRENT CONTROLFILE FOR STANDBY** 在备份命令中包含一个备用控制文件。RMAN为了创建备用数据库必须在其备份中有一个备用控制文件。
- **DUPLICATE TARGET DATABASE FOR STANDBY** 创建一个副本数据库作为目标的备用数据库。这个命令类似于第15章中的数据库克隆操作。
- **DORECOVER** 从目标数据库用归档日志备份恢复备用数据库。一旦备用数据库的创建完成，RMAN将对备用数据库应用主数据库中的所有归档日志，一直到RMAN中注册的最

新的归档日志。

在RMAN为备用操作复制数据库期间，RMAN将自动执行任务的所有要求。当连接到目标、辅助和可选的目录数据库时，RMAN将进行以下任务：

- 1) 决定需要还原哪个基本备份。
- 2) 根据辅助数据库参数或者RMAN设置命令和选项，确定将数据文件放置在辅助实例的什么位置。
- 3) 为辅助数据库还原备用控制文件。
- 4) 加载备用数据库控制文件。
- 5) 从第1)步决定的备份片或者映像拷贝来还原数据文件。
- 6) 如果指定了DORECOVER选项，将根据恢复停止点对还原的数据文件应用所有的增量备份。
- 7) 如果指定了DORECOVER选项，将根据恢复停止点从磁盘或备份对还原的数据文件应用所有归档日志文件。

为什么要用RMAN来创建备用数据库呢？如果所有的备份都是RMAN备份，则必须要应用RMAN。如果既有RMAN备份也有用户管理的备份，则RMAN进行备用数据库的创建将更容易且自动化程度更高。RMAN可以创建新的备用控制文件并处理数据文件的还原和恢复。

当RMAN创建备用数据库时，应注意以下事项：

- 如果在备用数据库还原期间不希望使用目录，则应在RMAN命令行中指定NOCATALOG。
- 在备份备用控制文件时的重做日志必须被归档。（在恢复时，归档日志文件应用需要数据文件头的备份SCN。）
- 必须有包含备用控制文件的备份，否则，当试图创建备用数据库时会得到一个“RMAN-06024: no backup or copy of the control file found to restore”错误。
- 不能在备用创建期间执行完全恢复。如果当前的联机重做日志对数据库不可用，则常常会发生这种情况。应用DORECOVER选项后，RMAN将应用已知的归档日志文件。
- 如果备用数据库驻留在其他的服务器上并拥有与主服务器相同的目录结构，则在备用创建期间不必重新命名任何一个文件。默认情况下，DUPLICATE命令将不能继续执行，而RMAN将会通知还原的辅助文件的位置与被备份的目标数据库文件的原始位置是一致的。这样做可以防止DBA偶然覆盖已存在的数据文件。NOFILENAMECHECK选项将会忽略这个检查。
- 备用数据库的数据库名称必须和主数据库的名称相同。如果主数据库和备用数据库在同一机器上（类似本章中的练习），则对备用数据库使用LOCK_NAME_SPACE初始化参数。这将允许两个具有相同名称的数据库驻留在同一机器上，而不相互干扰。
- RMAN将只创建备用数据库并在创建点上初步恢复该数据库。不能进行备用数据库的后续的维护和恢复（传送、应用归档重做日志文件）。

在本章中，将使用RMAN创建PRACTICE数据库的一个备用数据库（参见图16-1）。当创建完成这个备用数据库后，将使用RMAN来备份这个备用数据库，有效地备份主数据库。

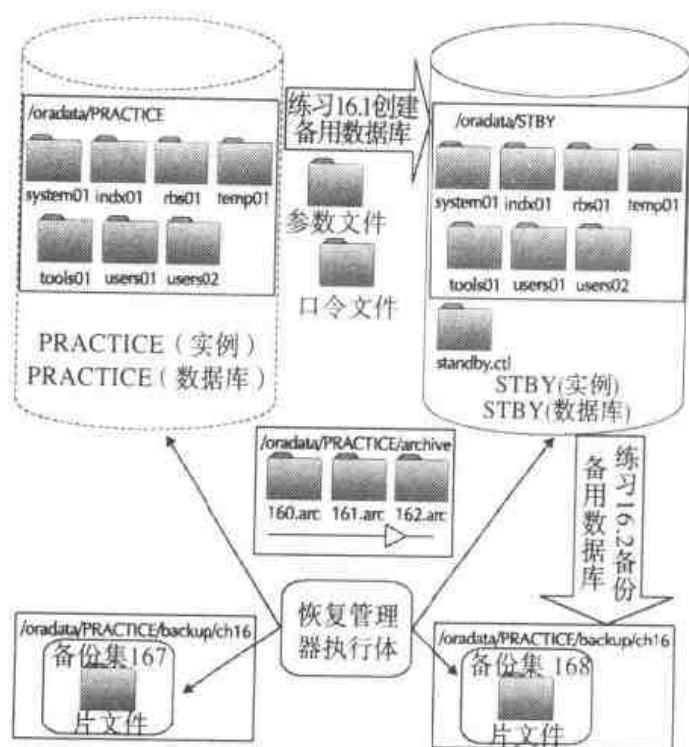


图16-1 用RMAN创建并备份备用数据库

16.1 恢复管理器的备用

当用RMAN创建备用数据库时，需要遵循以下步骤，其中很多步骤与创建副本数据库的步骤相同。

- 1) **准备备用主机** 备用数据库可以驻留在与主数据库相同的机器或者其他机器上。如果都在同一机器上，则数据文件、控制文件和联机重做日志所在的路径必须不同。因此，必须首先创建包含备用数据库的路径结构。这与第6章的练习6.1中的步骤相同。
- 2) **创建主数据库的RMAN备份** 为了用RMAN创建备用数据库，则需要一个合适的用于RMAN向备用服务器上还原的备份。这个备份可以在主数据库打开、关闭时生成，并且可以是映像拷贝、代理拷贝或备份。备用数据库还需要一个特殊的备用控制文件。在8.1.7版本以前，这个控制文件备份必须用SQL命令（ALTER DATABASE CREATE STANDBY CONTROLFILE AS）来创建，但现在RMAN也可以生成这个控制文件。
- 3) **配置备用数据库的参数文件** 必须为备用数据库创建数据库初始化参数文件。备用数据库需要的许多参数都与Oracle进行自动命名数据文件和日志文件有关。这可以在数据文件需要迁移到其他位置时，防止覆盖文件和人工干涉。这个参数文件将与第7章中的作用相同（参见练习7.3中的任务3）。
- 4) **配置备用数据库的口令文件** 为了DBA身份认证，备用数据库需要一个口令文件。可以按照第6章的步骤得到这个文件（练习6.1中的任务1）。
- 5) **配置Net8的配置文** RMAN需要同时连接主数据库和备用数据库。如果这两个数据库驻留在不同的服务器上，可以用Net8 Assistant来配置复制服务器上的监听器以及配置运行RMAN的服务器上的tnsnames。

- 6) **使将要驻留备用数据库的服务器可以访问RMAN备份** 如果该备份创建在磁盘上, 则该备份可以用ftp的二进制传输模式、类似NFS的磁盘共享技术或Windows的共享驱动器映像等技术进行传输。备份必须以和主服务器上相同的位置保存到备用服务器上的路径下, 这样才能使RMAN知道从哪里可以重新找到文件。如果使用的是磁带备份, 则复制服务器必须可以访问主数据库用来备份的介质管理层和磁带库。如果不能访问, 则可以在主服务器(确保为重建的数据文件指定了不同的路径, 以防止没有覆盖生成的数据库)上还原备份, 然后传送到备用服务器上。如果主数据库和备用数据库驻留在同一台机器上, 则该步骤可以省略。
- 7) **启动备用数据库实例** 在RMAN创建备用数据库之前, 备用实例需要以非加载(NOMOUNT)模式启动。此时将读取第3步中建立的参数文件。
- 8) **加载或打开主数据库** RMAN创建备用数据库时, 需要加载或者打开主数据库。
- 9) **创建备用数据库** 一旦RMAN连接到主数据库和备用实例上, 将建立通道并创建备用数据库。一旦数据库被还原, 就可以应用增量备份和重做以向前回退该数据库。当备用创建之后, 数据库将保持加载状态一直等到被设置为管理恢复模式或被手动应用重做日志。

备用数据库通过对归档重做日志文件的应用来保持与主数据库的同步。当备用数据库创建之后, RMAN就可以执行恢复操作。如果通知RMAN在备用数据库创建时就恢复, 则Oracle将决定是在刚才还原的基准备份层上应用增量备份还是应用归档日志文件。RMAN通常将还原基准备份(全部或者增量级别为0), 接着再应用所有的后续级别的增量备份。一旦最后一个增量备份应用之后, 则将应用归档重做日志使数据库向前回退。将DORECOVER关键字附加在DUPLICATE命令后, 会自动在备用数据库创建时进行恢复。如果这样操作了, RMAN将自动还原最新的基准备份, 应用所有后续的增量备份, 然后应用归档重做日志。除非给定了一个恢复停止点(用SET UNTIL), 恢复将继续执行直到RMAN所知道的最后一个归档日志被应用。这是用RMAN创建备用数据库最好的方法, 因为可以减少大量人工操作。

默认情况下, 当你让RMAN创建备用数据库时, 基准备份会被还原但是并没有进行恢复。这意味着, 必须手工应用此备份之后的归档日志文件。如果最新的基准备份是很长一段时间之前进行的, 则在应用所有这些归档日志之前需要进行许多工作, 并且这些文件在磁盘上是可用的。

警告 将备用数据库与主数据库保存在同一机器上并不是一个明智的方法, 因为, 如果机器失效了, 则主数据库和备用数据库都会丢失。我只是为了练习的目的, 才介绍这种单机主/备用数据库的配置方式。

练习16.1: 用RMAN创建备用数据库

在本练习中, 将使用一个包括备用数据库控制文件的RMAN备份创建一个备用数据库。应用该备份, 将创建一个备用数据库、恢复该数据库并以只读方式打开。在本章中, 主/目标数据库是PRACTICE数据库, 备用数据库是STBY。

任务描述	时间 (分钟)
1. 准备备用数据库	5
2. 备份PRACTICE数据库	5
3. 生成备用数据库	5
4. 验证备用数据库的生成	10
总计时间	25

本练习的目的是用RMAN创建一个备用数据库，类似于第7章中采用用户管理技术进行的操作。管理和激活备用数据库不是RMAN的任务，RMAN只是创建备用数据库。因此，除了验证备用数据库是否已经正确地创建，将不再在本练习中重复管理任务。

任务1：准备备用数据库

准备创建的备用数据库需要和第7章中一样的一些支撑文件。现在，做一些练习来创建一个数据库（参看第6、7、9和15章）。实际上，如果已经在第7章中建立了备用数据库，则所有这些步骤都已经完成了。为备用数据库准备以下部件：

- **目录** 创建所有的管理目录（\$ORACLE_BASE/admin/STBY）和数据目录（/oradata/STBY）。
- **参数文件** 拷贝PRACTICE数据库的参数文件并命名为\$ORACLE_BASE/admin/STBY/pfile/initSTBY.ora。这样就创建了一个参数文件。配置备用数据库参数文件：将所有“PRACTICE”修改为“STBY”，除了数据库名（DB_NAME）必须保留，使备用数据库与原始数据库的名称相同。为了在备用数据库创建期间转换文件名，增加两个转换参数。同时，增加LOCK_NAME_SPACE参数以便可以在同一台机器上同时打开两个同名的数据库：

```
db_file_name_convert = ("PRACTICE", "STBY")
log_file_name_convert = ("PRACTICE", "STBY")
lock_name_space = "STBY"
standby_archive_dest = /oradata/STBY/archive
```

- **口令文件** 为备用数据库创建一个口令文件，以便RMAN可以用SYSDBA身份连接到SYSDBA。
- **Windows服务** 在Windows环境中，使用oradim工具为STBY数据库创建一个新的名为OracleServiceSTBY的数据库服务。
- **配置Net8** 最后，为STBY配置Net8连接。这一步对于第7章中的复制数据库是不必要的，因为不需要同时连接到STBY实例和PRACTICE数据库。使用Net8 Assistant，为STBY数据库增加一个数据库服务，并为STBY数据库配置监听器。

如果已经像第7章中描述的那样创建了一个备用数据库，那么上述所有这些步骤都不是必须的（除了为STBY参数文件增加两个转换参数）。数据库文件将已经存在于/oradata/STBY目录中。可以保留这些以前生成的数据库文件（数据文件、日志文件和控制文件），也可以删除这些文件。为了删除这些文件，需要将/oradata/STBY目录下的所有文件都删除。

任务2：备份PRACTICE数据库

创建备用数据库需要一个备用控制文件的备份。备用控制文件备份以后还需要一个日志切

换。在命令行中增加一个日志文件参数，以便该脚本的输出可以输入到一个文件中：

```
LINUX> rman trace=b_standby.log
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> @b_standby.rcv
```

下面显示的备份除了与前几章的RMAN备份有一些差别外，其他的基本上都相同。这个备份显示了数据库的几个不同部分是如何在一个单一备份脚本中被备份的。以前，数据文件和归档日志文件的备份是分别运行不同的脚本产生的。

```
run {
  allocate channel d1 type disk;
  backup
    incremental level 0
    database
    format '/oradata/PRACTICE/backup/ch16/db_%d_%s_%p_%t'
    tag = 'STBY_INCO'
    include current controlfile for standby;
  sql "alter system archive log current";
  backup archivelog
    from time 'SYSDATE - 1/24'
    format '/oradata/PRACTICE/backup/ch16/ar_%d_%s_%p_%t';
}
```

现在读者已经熟悉了大多数的备份脚本。下面是对这个备份的一些说明：

- **control file for standby** 当备份整个数据库时，控制文件将自动包含在系统表空间中。为了备用数据库的生成必须创建一个特殊的控制文件。RMAN可以在一个数据文件备份片中存储这个特殊的文件。如果需要，这个备份控制文件也可以用于还原目标数据库的控制文件。
- **archive log current** 当备用数据库控制文件创建时，备用数据库的恢复从这个当前日志开始。因此，这个日志文件应该在备用数据库被创建之后，为了在备用数据库上恢复而被归档。
- **from time 'SYSDATE - 1/24'** 在归档日志备份集中最新的归档日志文件。这个命令将备份包含最近时间内值得重做的归档日志。

查看在屏幕上的或者RMAN日志文件中的备份输出，将会看到表示备用控制文件已经包含在这个备份中的消息：RMAN-08020: including standby controlfile in backupset

在进入下一个任务之前，请记录PRACTICE数据库当前的日志序列号。自动恢复将从这个日志序列号开始：

```
SQL> connect sys/practice@practice
SQL> SELECT sequence# FROM v$log WHERE status = 'CURRENT';
```

在这个练习中，可以认为从这个查询返回的日志序列号是162。也可以在RMAN输出信息中被备份的sequence#中看到这个结果。

如何知道一个备份集中是否包含一个备用控制文件？可以在目录中查询名为RC_BACKUP_

CONTROLFILE的视图中的该注册数据库，控制文件类型“S”(Standby)。正常的控制文件备份将是一个控制文件类型“B”(Backup)。下面显示的结果是从第一个查询中得到的，列出了刚刚创建的备份。可以与备份输出的跟踪文件b_standby.log文件比较set-count。

```
SQL> connect rman817/rman@rcat
SQL> SELECT set_count, checkpoint_change#, checkpoint_time,
2         status, completion_time, controlfile_type
3     FROM rc_backup_controlfile
4     WHERE db_name = 'PRACTICE' AND controlfile_type = 'S'
```

同时下面的查询语句可以在备份集和备份片中查找包含的备用控制文件。

```
SELECT bs.bs_key, bs.set_count, bs.backup_type, bs.incremental_level,
       bs.completion_time, bs.elapsed_seconds, bp.handle, bp.status
FROM rc_backup_set bs, rc_backup_piece bp
WHERE bs.controlfile_included = 'STANDBY'
      AND bs.db_id = bp.db_id
      AND bs.bs_key = bp.bs_key
      AND bs.db_id IN
      (SELECT dbid FROM rc_database WHERE name = 'PRACTICE');
```

该查询的输出与来自备份执行的日志文件中显示的文本有关。确保日志文件中备份片的名称与该查询语句中的列名相同。备份经历的时间也相同。

任务3：创建备用数据库

只需要几个命令就可以完成备用数据库的创建。简单地启动RMAN并连接到目标、目录和辅助数据库上。下面的示例是Linux对NLS语言环境变量的设置。如果目标数据库与默认的环境的语言属性相同，则该变量没有必要设置。应该注意，如果NLS_LANG环境变量与目标数据库的不同则有可能使备用数据库的创建失败。

```
LINUX> export NLS_LANG=AMERICAN.WE8ISO8859P1
LINUX> rman trace=standby.log
RMAN> @standby.rcv
```

RMAN执行的备用数据库创建工作只需要几个命令。下面列出的命令可以用前述任务中创建的备份生成备用数据库：

```
connect target      sys/practice@practice
connect catalog     rman817/rman@rcat
connect auxiliary   sys/stby@stby
run {
    set command id to 'Standby Creation';
    allocate auxiliary channel dl type disk;
    duplicate target database for standby dorecover;
}
```

这几行命令用数据库的备份创建了PRACTICE数据库的副本。然而，这个副本被配置成为PRACTICE数据库的一个STBY。这个备用数据库将被创建到辅助实例。尽管这个脚本中的命令很少，但实际上RMAN完成了大量的工作：

- **set command id** 这个设置选项使RMAN填充了v\$session视图的client_info列。因此，在

STBY数据库被创建时，可以将client_info的值设置为“Standby Creation”来查询v\$session表，并观察会话的进行。

- **duplicate target database for standby** 这个命令使创建备用数据库需要完成大量的工作。首先，备用数据库的控制文件需要被复制到辅助数据库init.ora文件中的CONTROL_FILES参数值指定的位置。然后，目标数据库备份中的数据文件将被还原到辅助数据库的还原位置上。
- **dorecover** 当目标数据库复制之后，备份数据库将用增量RMAN备份和归档重做日志恢复。一直恢复到停止点的位置。

RMAN复制命令如何知道需要在什么位置创建复制的数据文件呢？因为备用数据库和主数据库在同一服务器上，所以不能使用相同的文件名和路径。复制命令将按照下面的顺序在三个地方查找数据文件存放位置的指示：

- **set newname** 为特定的复制命令指定新的数据文件名。在一个运行块内，该参数定义了一个文件名。对于特殊的复制命令，这个设置是暂时性的。这个新名称的值只在运行块内有效。
- **set auxname** 为所有辅助操作指定一个新的数据文件名，该设定在其他指定以前都有效。这个设定在运行块以外执行并在RMAN会话之间有效。
- **db_file_name_convert** 只为数据库的参数文件中指定一个转换字符串。这个设置对该数据库的所有数据文件都适用。

日志文件可以用以下两种方法之一进行设定：

- 在复制命令中设定
- 在辅助数据库参数文件中的：log_file_name_convert

任务4：验证备用数据库的生成

再次检查备用数据库的文件是否都已经保存在正确的目录（/oradata/STBY）下了。也可以连接到备用数据库并查询v\$database, v\$datafile, v\$logfile和 v\$controlfile以证实是否所有的数据库文件都保存在期望的位置。

既然已经创建了备用数据库，现在要验证这个STBY数据库是否确实是PRACTICE数据库的一个备用数据库。如何才能验证STBY是一个备用数据库呢？可以这样检查备用数据库的功能：修改主数据库的数据，将这个改变传递给备用数据库，以只读方式打开备用数据库，然后检查修改的数据。

- **修改主数据库中的数据** 为了验证STBY执行了备用数据库的功能，修改主数据库中的数据。首先，删除TINA.DATE_LOG表中所有表示未来时间的记录行。然后，在该表中插入一个未来16年的记录，稍后将在备用数据库中可以看到该记录。在确定了这些修改之后，检查当前的重做日志序列号并切换日志文件。刚才对表TINA所做的修改将包含在最新归档的日志文件的重做记录当中。如果修改时的日志序列是162，则日志切换将会在/oradata/PRACTICE/archive目录中生成一个名为162.arc的归档日志文件：

```
SQL> connect sys/practice@practice
SQL> DELETE FROM tina.date_log WHERE create_date > SYSDATE;
SQL> INSERT INTO tina.date_log VALUES (SYSDATE+365*16);
```

```
SQL> SELECT sequence# FROM v$log WHERE status = 'CURRENT';
SQL> COMMIT;
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

- **通过归档日志传递数据** 一些发生在主数据库的数据修改还没有被应用到备用数据库中。这些修改将包含在一个名为/oradata/PRACTICE/archive/162.arc的归档日志文件中。为了将这些修改传递给备用数据库，这个新生成的归档日志文件必须应用于备用数据库。连接到备用数据库并应用该归档日志文件。因为本练习中数据库都在同一机器上，所以没有必要传送这个日志文件，只是通知恢复命令去查询该文件所在的路径。然后，在恢复命令中选择自动恢复选项来恢复备用数据库。

```
SQL> connect sys/stby@stby as sysdba
SQL> set logsource /oradata/PRACTICE/archive
SQL> recover standby database;
```

恢复日志文件包含了对TINA表的修改。

- **从备用数据库读取新的数据** 检查在PRACTICE数据库中生成并被传递给STBY数据库的更改数据。以只读方式打开数据库。从TINA.DATE_LOG表选择行记录。将会看到一个未来16年的记录。然后关闭备用数据库并返回到备用模式。

```
SQL> ALTER DATABASE OPEN READ ONLY;
SQL> SELECT create_date FROM tina.date_log WHERE create_date > SYSDATE;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

如果对于这个任务有任何疑问，请回顾第7章的内容，以便了解如何验证备用数据库更改的详细情况。现在，用RMAN已经创建了一个备用数据库。

16.2 恢复管理器备用数据库的备份

一旦创建了备用数据库，就可以用RMAN来备份这个备用数据库了。备份备用数据库，实际上就是对主数据库的备份。这个工作也可以由用户管理备份完成，但相比较而言，用RMAN更简单且易于操作。既然已经有了一个主数据库的额外拷贝，而且这个拷贝可以保存在与实际的产品级环境不同的机器上，那么就可以从这个机器上进行备份了。当备份是从备用数据库的服务器而不是主数据库的服务器进行时，减少了由于将数据传送给介质管理层之前读数据文件和归档日志文件而引起的主服务器的资源开销。

回顾第11章，那里讲述了用DBID作为每个数据库唯一的标识并将每个数据库注册在RMAN的目录中。这个由Oracle提供给数据库的DBID是数据库第一次生成时获得的，并且在一些情况下会发生改变，其中一种情况就是用RMAN创建了一个复制的数据库。如果希望用RMAN进行新创建的复制数据库的RMAN备份，则必须首先在目录中注册（第11章练习11.1任务3中有操作指导）。RMAN自动强迫复制数据库拥有一个新的DBID，该DBID与该数据库拷贝得到的不同。一个备用数据库，在还没有被激活成为主数据库时，RMAN可以将其当作备用数据库的原始数据库进行备份。因为该备用数据库就是主数据库的一个拷贝，而且DBID也一样。备用数据库通过读取从主数据库获得的归档重做日志而保持与主数据库同步。主数据库和备用数据库数据文

件内容的差别依赖于有多少归档日志文件还没有被传递和应用。为了进行备用数据库的备份，需要停止恢复并关闭，然后以非加载方式启动。使用RMAN连接到备用数据库和恢复目录上，并进行数据文件的冷备份。甚至可以对所有从主数据库传送来的归档日志文件进行备份。这将节约CPU、内存以及磁盘资源，否则会占用主服务器的资源。不能用RMAN对备用控制文件进行备份，但却可以用于还原主数据库，必须从主数据库进行控制文件的备份。备用的控制文件只能用于备用数据库。同时必须确保主服务器和备用服务器都能与MML和磁带库进行通信。从备份数据库写数据，然后向主服务器还原这些数据。用这种备份方法，可以还原从主数据库得到的文件并用于备用数据库，或者可以还原从备用数据库备份得到的文件并在主数据库中还原。

练习16.2：用RMAN备份备用数据库

既然已经创建了一个备用数据库，下面就用RMAN备份该备用数据库。这个备份备用数据库的任务很容易，只需连接到备用数据库（而不是主数据库）即可。

任务描述	时间（分钟）
1. 备份备用数据库	10
总计时间	10

任务1：备份备用数据库

生成一个类似于第12章中用于增量级别0备份的备份脚本。可以使用同样的脚本，但要更改tag和备份路径以便区分以前的备份。下面是一个RMAN命令的脚本：

```
connect target sys/stby@stby
connect catalog rman817/rman@rcat
run {
    allocate channel d1 type disk;
    backup incremental level = 0 cumulative database
        format '/oradata/PRACTICE/backup/ch16/db_%d_%s_%p_%t'
        tag = 'WHOLE_STANDBY_INC0';
}
```

用RMAN备份备用数据库（b_whole_standby_inc0.rcv）。这个脚本将创建一个目标数据库的备份。注意，这个备份的tag将是WHOLE_STANDBY_INC0。同时应注意，备份片文件将保存在ch16目录下。这样容易找到备份集和备份片文件。

运行备份备用数据库并将输出定向到一个跟踪文件中，命令如下：

```
LINUX> rman trace=b_whole_standby_inc0.log
RMAN> @b_whole_standby_inc0.log
```

当进行备用数据库备份时，输出结果类似于以前的输出。然而，注意数据文件备份的路径，都不在PRACTICE数据库的路径下。在日志文件中（以及屏幕上）将看到类似以下的输出：

```
RMAN-08010: channel d1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00001 name=/oradata/STBY/system01.dbf
RMAN-08522: input datafile fno=00002 name=/oradata/STBY/rbs01.dbf
RMAN-08522: input datafile fno=00004 name=/oradata/STBY/temp01.dbf
RMAN-08522: input datafile fno=00005 name=/oradata/STBY/tools01.dbf
RMAN-08522: input datafile fno=00006 name=/oradata/STBY/indx01.dbf
```

```

RMAN-08522: input datafile fno=00003 name=/oradata/STBY/users01.dbf
RMAN-08522: input datafile fno=00007 name=/oradata/STBY/dnew07.dbf
RMAN-08013: channel dl: piece 1 created
RMAN-08503: piece
handle=/oradata/PRACTICE/backup/ch16/db_PRACTICE_11_1_450481477

```

任务2: 列表备份的内容

备份已经完成了, 可以通过tag标识列出其内容。这个列表显示的内容与备份的日志文件(b_whole_standby_inc0.log)中的信息一样。确认备份集中的数量是相同的, 并且备份片名称也是一样的:

```
RMAN> list backup tag 'WHOLE_STANDBY_INC0';
```

有些有趣的事情需要注意: 该备份中列出的数据文件的位置是主PRACTICE数据文件的位置:

List of Datafiles Included

File Name	LV	Type	Ckp	SCN	Ckp Time
1 /oradata/PRACTICE/system01.dbf	0	Full	201752		16-JAN-02
2 /oradata/PRACTICE/rbs01.dbf	0	Full	201752		16-JAN-02
3 /oradata/PRACTICE/users01.dbf	0	Full	201752		16-JAN-02
4 /oradata/PRACTICE/temp01.dbf	0	Full	201752		16-JAN-02
5 /oradata/PRACTICE/tools01.dbf	0	Full	201752		16-JAN-02
6 /oradata/PRACTICE/indx01.dbf	0	Full	201752		16-JAN-02
7 /oradata/PRACTICE/users02.dbf	0	Full	201752		16-JAN-02

尽管备用数据库的备份在/oradata/STBY路径下为数据文件拷贝了数据块, 但目录和控制文件认为这些文件在the /oradata/PRACTICE路径下。因此, 还原将自动把这些数据文件存放到主服务器上的适当位置。

还要做一件事: 再次连接到主数据库上, 并使用同样的list命令查看备份是否存在。关于备用数据库备份的信息包含在备用控制文件和目录中。

```

LINUX> rman
RMAN> connect target sys/practice@practice
RMAN> connect catalog rman817/rman@rcat
RMAN> list backup tag 'WHOLE_STANDBY_INC0';

```

当list命令执行时, 目录和主控制文件会自动同步。RMAN智能化很高, 可以在服务器管理的内容中发现PRACTICE数据库控制文件和目录之间的更改。该目录目前包含关于备份的信息, 而目标控制文件则没有。

```

RMAN-03022: compiling command: list
RMAN-03024: performing implicit full resync of recovery catalog

```

一旦再同步完成, 将看到连接到PRACTICE数据库的备份和将STBY数据库作为目标的备份是一样的。因此, RMAN将STBY备用数据库的备份当作PRACTICE主数据库的备份。

16.3 疑难解答

如果用RMAN在创建备用数据库时遇到了问题, 可以参照以下列出的错误消息及注解:

- 1) RMAN-06024: no backup or copy of the control file found to restore 当创建备用数据库时, RMAN需要一个备用控制文件的备份。如果没有进行备用控制文件的备份, 或者用于备用还原与恢复的停止点发生在备份进行之前, 会出现这个错误信息。
- 2) RMAN-05500: the auxiliary database must not be mounted when issuing a DUPLICATE command 如果备用数据库创建失败, 辅助数据库STBY仍将加载新生成的控制文件。确认STBY数据库以非加载方式启动。
- 3) You created a standby but the datafiles were not named and located where you thought they'd be 在以前的RMAN会话期间设置的auxname将一直有效, 直到数据文件设置为非设置状态。例如, 如果将数据文件1设置为/oradata/STBY/datafile01.dbf, 而在后续的复制命令中, 这个名字会被再次使用。如果采用了set newname选项, 则只在这个运行块内不能使用auxname。可以将set auxname #设置为null命令来重置set auxname。
- 4) RMAN-05507: standby control file checkpoint (180959) is more recent than duplication point in time (180957) 在当前的联机重做日志文件中将执行包含备用控制文件的备份。

16.4 小结

在本章中, 讲述了如何应用比手工操作方法更简单的RMAN创建备用数据库。没有必要浪费时间在从还原的备份处还原并应用归档日志以便将数据库恢复到当前点的位置上。RMAN将还原基准备份并应用增量备份以及归档日志文件。同时本章还讲述了如何进行备用数据库的备份, 这个备份可以用于主数据库还原。因为这两个数据库有相同的名称和DBID, 所以可以这样操作。在备用数据库上进行备份可以减少从主服务器上进行备份的工作量。表16-1列出了采用用户管理和服务器管理技术创建备用数据库的差别。

表16-1 比较用户管理和服务器管理的备用数据库创建

用户管理复制	服务器管理复制
配置备用数据库(路径、参数文件、口令文件Windows服务); Net8配置可选	配置备用数据库(路径、参数文件、口令文件、Windows服务); Net8配置可选
采用用户管理备份主数据库打开或关闭备份, 备份需要与所有数据文件一样大的磁盘或磁带空间	采用服务器管理备份目标数据库打开、关闭或映像拷贝。备份需要的磁盘或磁带空间比所有的数据文件的空间小。映像拷贝需要与所有数据文件一样大的磁盘或磁带空间
在主数据库上创建备用控制文件并拷贝到备用数据库	数据库的备份必须包含一个备用控制文件的拷贝
通过操作系统的命令删除备份文件	只有当副本数据库保存在其他机器上且备份存储在主数据库磁盘或副本服务器上的磁带系统不可用时, 才通过操作系统命令删除文件
用SQL*Plus的命令创建备用控制文件	当连接到目标、目录或者辅助数据库时发布复制数据库命令
要从备份的时间处恢复备用数据库需要重建归档日志文件并应用这些文件	RMAN处理增量备份和归档重做日志文件的应用

在《Oracle 8i Standby Database Concepts and Administration Guide》一书中, 可以查找到要成功地实现Oracle备用数据库的所有细节。在Oracle 8i的文档附录中包含更多用RMAN创建备用

数据库的信息。还可以从Oracle Metalink web站点获得关于用RMAN创建备用数据库的信息。

习题

回答以下有关备用数据库创建的问题，以便巩固本章的主要内容。

1. 什么RMAN命令可以从一个RMAN备份创建一个备用数据库？

- A. duplicate target database for standby
- B. create standby database
- C. duplicate standby database on auxiliary
- D. duplicate standby

2. RMAN在DUPLICATE TARGET DATABASE FOR STANDBY命令中都完成了哪些任务？

(多项选择)

- A. 从RMAN备份还原备用控制文件
- B. 从RMAN备份还原数据文件
- C. 选择性地恢复还原的数据文件
- D. 还原联机重做日志文件

3. RMAN可以从RMAN备份还原备用数据库并执行初始的备用恢复，但不能在备用数据库上执行维护操作。

- A. True
- B. False

4. 默认情况下，创建RMAN备用数据库并不从目标/主RMAN增量备份和归档日志备份来恢复备用数据库。在复制命令中增加什么可选参数就可以完成这个任务？

- A. recover
- B. propagate
- C. dorecover
- D. apply

5. 可以用RMAN备份备用数据库并用这些备份来恢复主数据库。

- A. True
- B. False

答案

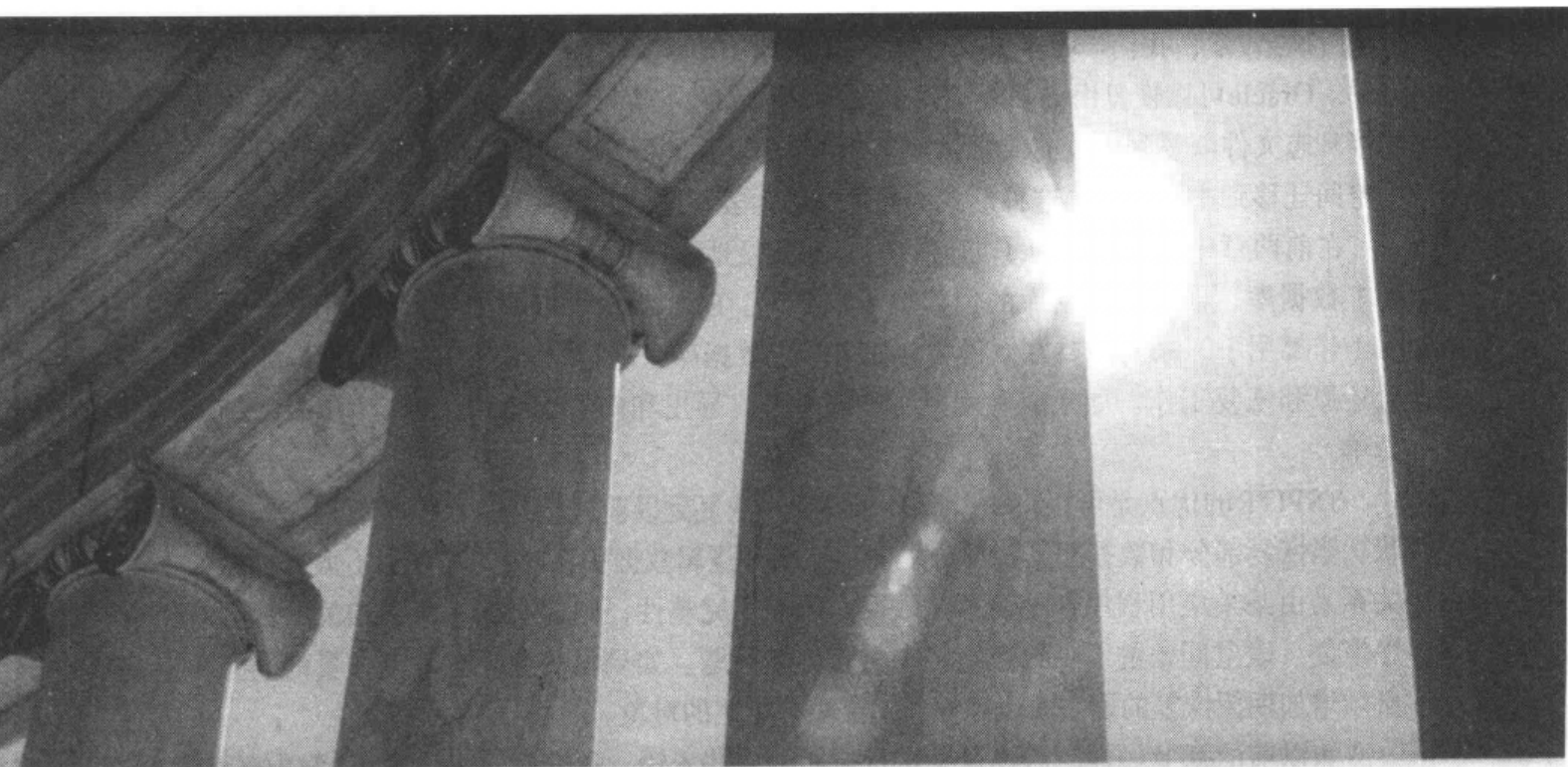
1. A. 复制命令用包含有备用控制文件的备份创建备份数据库。

2. A、B、C。RMAN还原备用控制和所有的数据文件并可选择性地将数据库到恢复最近的归档日志文件。联机重做日志文件没有还原，这些文件在备用数据库被激活以前甚至都不存在。

3. True。RMAN能创建数据文件，但只是在初始阶段创建之后管理从主数据库到备用数据库的归档日志文件的传输。

4. C。dorecover选项通知RMAN从RMAN备份和归档日志文件对备用数据库应用重做操作。这些备份可以是增量1-4级的备份或者是归档日志文件。增量备份首先被应用，然后是归档日志文件。如果归档日志文件在磁盘上存在，则首先应用这些文件。

5. True。一个备用数据库就是一个主数据库的完全拷贝。二者甚至具有相同的DBID。RMAN可以备份一个备用数据库。这些备份可以被用于恢复主数据库。



第17章 RMAN 表空间恢复

回顾第9章，我们学习了如何将数据库中删除的表恢复到错误发生之前，而不必恢复整个数据库。Oracle可以恢复由许多表空间组成的数据库的一部分。表空间数据文件被还原，然后应用归档日志文件恢复到失效前或错误发生处。下来用8i的可迁移表空间功能将包含需要恢复对象的表空间迁移到主数据库。这种形式的恢复被称为表空间时间点恢复（TSPITR）。

在前两章中，我们学习了如何用RMAN方便地进行恢复任务。只需要几个命令就可以创建副本数据库，而备用数据库的创建也非常简单。RMAN也可以用来处理TSPITR，这样就比所有工作都用手工来完成要方便得多。如果对主数据库只进行了RMAN备份，则只能用RMAN执行还原和恢复工作。如果兼有用户管理的备份、导出和RMAN备份，则应用RMAN依然快捷、简单。

TSPITR的优点是可以将一个或多个表空间恢复到以前的状态，而保持数据库其余部分的现状。当恢复部分和数据库其余部分之间存在整体逻辑数据关系时，TSPITR不太方便。如果数据关系是由某个应用程序管理而不是数据库的应用完整性，则需要确保所有相关的数据对象按顺序恢复。表空间中的不一致数据将会引起许多问题。如果相关数据分散在大量的表空间中，则必须增加期望恢复的表空间以确保包括了所有相关的对象。

和以前的章节一样，需要定义本章应用的一些术语。尽管这些术语在第9章中已经讨论过，我认为有必要再次介绍。（在阅读本章前，可以复习第9章内容。）

- **辅助实例：**恢复特定表空间的后台进程和内存结构。该实例将打开辅助数据库，本章中叫做AUXY。
- **辅助数据库：**为了临时表空间恢复而创建的主/目标数据库的子集。本章中，AUXY实例将打开PRACTICE数据库的一个子集。PRACTICE数据库将是一个重建的备份，包括一些数据文件和来自主PRACTICE数据库的控制文件。
- **主数据库：**需要TSPITR的数据库。在本章中是PRACTICE数据库，其USERS表空间将从主数据库的所有其他表空间恢复到不同的时间点。
- **恢复集：**组成表空间的数据文件将被恢复到以前某时间点。SYSTEM表空间数据文件不属于恢复集。在本章中，恢复集是USERS表空间中的两个备份的数据文件。
- **辅助集：**表空间恢复需要的所有数据库文件。在本章中，辅助集是主数据库PRACTICE的备份控制文件、SYSTEM和RBS表空间的备份数据文件、辅助数据库的参数文件和主数据库PRACTICE的归档日志文件。

在TSPITR期间，RMAN将完成一系列的任务。当连接到目标、辅助和可选的目录数据库时，RMAN将进行以下任务：

- 1) 根据最近进行的备份或提供的恢复停止点来确定还原需要的数据库子集需要哪些基本的备份。
- 2) 从RMAN备份为辅助数据库创建一个新的控制文件。
- 3) 根据辅助数据库参数或RMAN设置命令和选项来确定数据文件的存储位置。

- 4) 从第1)步确定的备份片或映像拷贝还原数据文件。这个功能与RMAN通常执行数据库还原是一样的。只有恢复集和辅助集中的文件才被还原与恢复。
- 5) 根据恢复停止点向还原的数据文件应用所有的增量备份。这个工作与在RMAN中发布恢复数据库命令是一样的。
- 6) 基于恢复停止点向还原的数据文件应用所有来自磁盘或备份的归档日志文件。与采用RMAN进行正常恢复一样,归档对象最后被应用到还原的数据库上。
- 7) 当重新设置联机重做日志文件时,打开辅助数据库。这个新的联机重做日志文件将根据从辅助参数文件翻译的内容创建。
- 8) 用可迁移表空间特性导出属于恢复的(不是辅助)表空间的数据字典元数据,并关闭辅助数据库。
- 9) 将恢复的表空间中的数据文件的位置切换到目标控制文件中的新的数据文件的位置。
- 10) 将恢复的表空间的数据字典元数据导入到主/目标数据库中。现在就可以访问恢复的表空间中的对象了。

在TSPITR结束时,恢复的表空间是脱机的。保持脱机状态是为了提醒大家注意,这些表空间应该被备份。一旦表空间参与到TSPITR中,则在TSPITR期间,以前的备份和归档日志文件都不能被用于恢复该表空间。进行新的备份为将来的恢复提供新的基准。

当考虑RMAN TSPITR时,应该注意Oracle文档中提到的几个条件,列举如下,其中有些限制条件与用户管理的TSPITR一样。

- 辅助数据库和主数据库必须驻留在同一机器上(可以在OPS族的其他节点)。如果机器缺少磁盘空间或者其他资源,则这个限制将会导致严重的问题。
- 运行没有目录的TSPITR也有许多限制。类似于其他RMAN恢复情况,可以只还原和恢复存储在目标数据库控制文件中的数据文件。如果只配置了数据库存储7天的RMAN备份信息,则不可能恢复21天前又没有进行更新的备份的表空间。
- 不能恢复已经删除的表空间,不能恢复已经删除但又用同样的表空间名还原的表空间。
- 不能在辅助数据库上进行数据库更改,辅助数据库只具有临时功能。
- 在恢复集中不能包括下面这些对象:副本主表、带有VARRAY列的表、包含嵌套表的表、包含外部文件的表、快照日志、快照表、SYS的对象和回退段。

警告 RMAN TSPITR也有同样的限制,所以在执行TSPITR之前必须进行研究,以确保没有混杂数据的相关性和一致性。要获得详细介绍请参看第9章中的练习9.1。

在本章中,用RMAN执行在第9章中进行的用户管理恢复练习。因为两者之间的差别比较明显,所以这样的操作很有帮助,而且还可以强化前面所学的内容。因此,在进行本章练习之前应复习第9章的内容。

17.1 恢复管理器的TSPITR

如何用RMAN进行TSPITR?需要遵循以下步骤,其中前面几步与创建副本数据库和创建备用数据库的步骤相同:

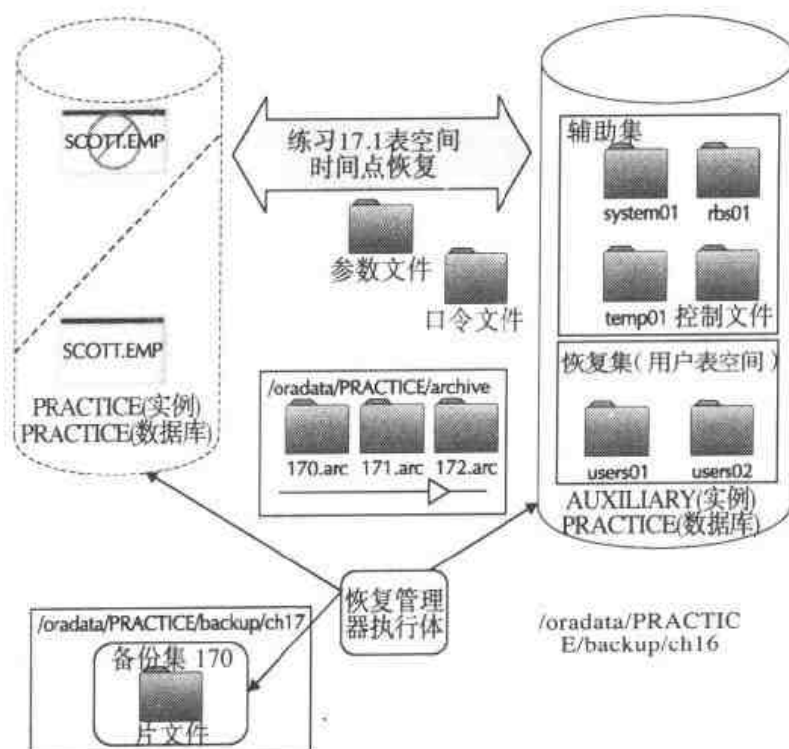


图17-1 用RMAN进行表空间时间点恢复

- 1) **准备主数据库的RMAN备份** 为了RMAN执行TSPITR, RMAN需要一个合适的备份来还原和恢复辅助数据库。备份可以在数据库打开或关闭时进行, 并且可以是映像拷贝、代理拷贝或者备份。因为辅助实例必须驻留在与主数据库相同的机器上, 所以没有必要像复制和创建备用数据库一样在其他主机上使磁带可用。
 - 2) **配置辅助数据库参数文件** 必须为辅助数据库创建一个数据库初始化参数文件。辅助数据库需要的大多数参数都与Oracle自动重命名以及人工干预数据文件和日志文件有关, 这样可以防止数据文件需要移动位置时的覆盖或者人工干预。重要的参数包括lock_name_space (用与同一机器上相同的数据库名为一个实例创建一个惟一的内存空间)、db_file_name_convert、log_file_name_convert (自动重新定位数据库文件和日志文件)。
 - 3) **配置辅助数据库的口令文件** 为了DBA身份认证, 辅助数据库需要一个口令文件。
 - 4) **配置Net8的配置文件** RMAN需要同时连接主数据库和辅助数据库。其中一个连接可以由ORACLE_SID环境变量定义为本地的, 而另一个则必须通过Net8连接。如果还要连接一个目录数据库则需要再配置一个Net8连接。
 - 5) **启动辅助数据库实例** RMAN在辅助数据库上执行TSPITR之前, 辅助实例必须以NOMOUNT模式启动。第2)步中建立的参数文件在此时将被读取。
 - 6) **打开主数据库** RMAN需要在TSPITR之间打开目标数据库。导入的恢复集字典数据需要目标数据库被打开。
 - 7) **连接目标、辅助和可选目录数据库时激活TSPITR** 当连接到辅助实例, 通知RMAN恢复表空间时, RMAN将执行TSPITR。为辅助实例分配磁盘通道并为恢复设置停止点。指定需要恢复的表空间, 然后RMAN执行剩下的任务。
- RMAN选择最合适的基准备份用来还原, 然后再计算需要什么样的增量和归档重做日志。

因为正在进行不完全恢复，所以当完成辅助数据库备份后，RMAN将自动以RESETLOGS模式打开辅助数据库。同时，辅助数据库需要重新生成日志文件。一旦辅助数据库被打开，RMAN将导出包含在恢复的表空间中的所有对象的元数据。同样地，当以手工方式从一个数据库向另一个数据库传送表空间时，数据将以同样的方式被导出。当元数据被导出后，辅助数据库就被关闭，而目标数据库的控制文件就被修改到新的数据文件刚恢复的位置。这些都是用RMAN的切换命令自动完成的。最后，元数据被导入到主数据库，而表空间仍然处于脱机状态。现在对这些表空间进行备份非常重要，因为在TSPITR期间这些以前的表空间备份将不能被用于前滚数据库。一旦进行了备份，将表空间联机，这些表空间可以使用了。

警告 必须确保为辅助数据库使用了不同的重做日志路径，以防止损坏主数据库。

练习17.1：表空间时间点恢复+

本章只有一个练习。这个练习的目标是完成一个简单的TSPITR操作。这样，就可以验证RMAN恢复一个包含了两个数据文件的单个表空间的工作。

任务描述	时间（分钟）
1. 删除表SCOTT.EMP	5
2. 准备辅助数据库	10
3. 执行RMAN TSPITR	10
4. 确认恢复	5
5. TSPITR之后的清理	10
总计时间	40

在这个练习中，目标数据库是PRACTICE，辅助数据库是AUXY。下面的任务是在错误地删除了一个表后，成功地恢复一个表空间。接着要在删除命令之前、之后修改对象，以便显示表空间的恢复如何影响其他对象。一旦恢复完成，就可以证实删除的表已经被恢复了，而其他的表空间数据保持不变。为了使情况更简单些，我选择了一个自含的表空间，我不希望改变第9章中所做的模式。这个练习不必进行对象依赖性检查。

任务1：删除表SCOTT.EMP

删除SCOTT的employee表。这次使用TSPITR恢复该表。在删除表之前标记时间。稍后，将把用户表空间恢复到这个时间，SCOTT的employee表和包含在这个表空间中的其他对象都将被恢复。

```
SQL> CONNECT sys/practice@practice;
SQL> SET TIME ON;
10:57:13 SQL> DELETE FROM tina.date_log WHERE create_date > SYSDATE;
10:57:35 SQL> DROP TABLE SCOTT.EMP;
10:57:46 SQL> SET TIME OFF;
```

可以看到，删除表发生在10:57:35。对于这个例子，该练习发生在2002年1月17日。在对AUXY实例的TPSPITR的任务3中将再次使用该时间。为了确认TSPITR按要求进行了工作，在删除表命令以后为USERS表空间创建一个对象。同时，在USERS表空间中的一个表中插入一行记录。最后，在USERS以外其他表空间中的一个表中插入一行记录：

```

SQL> CREATE TABLE scott.dept_copy TABLESPACE users AS
2 SELECT * FROM scott.dept;
SQL> INSERT INTO scott.dept (deptno, dname, loc)
2 VALUES ('50', 'SUPPORT', 'DENVER');
SQL> INSERT INTO tina.date_log VALUES (SYSDATE+365*17);
SQL> COMMIT;
SQL> ALTER SYSTEM SWITCH LOGFILE;

```

执行完TSPITR之后，主数据库PRACTICE的USERS表空间中将没有SCOTT.DEPT_COPY表。因为TSPITR之后USERS表空间的任何数据更改都不能被恢复，所以department表中没有support部门。最后，TINA.DATE_LOG中的数据在TSPITR之后将保持不变。应该在TSPITR之后看到一个未来17年的日期记录。还将发现没有恢复的表空间没有丢失任何数据或者对象。如果没有恢复的表空间中的对象与恢复的表空间中的对象（例如索引、触发器和约束）有直接的联系，那么它将受到影响。最后，切换日志文件使所有这些对PRACTICE数据库进行的更改都被记录到归档的日志文件中。

任务2：准备辅助数据库

准备名为AUXY的辅助数据库，与上一章中进行的准备一样，不同的是对所有的路径、参数和文件名应用AUXY而不是STBY。因为进行了第9章中的练习，所以这些文件也许已经存在了：

- **路径** 生成所有的管理目录(\$ORACLE_BASE/admin/AUXY)和数据库目录(/oradata/AUXY)。
- **参数文件** 用PRACTICE数据库的拷贝创建一个参数文件，并命名为\$ORACLE_BASE/admin/AUXY/pfile/initAUXY.ora。增加这些参数将所有PRACTICE改为AUXY（除了DB_NAME）。

```

db_file_name_convert = ("PRACTICE", "AUXY")
log_file_name_convert = ("PRACTICE", "AUXY")
lock_name_space = "AUXY"

```

- **口令文件** 为备用数据库创建一个口令文件，使RMAN可以用SYSDBA连接到AUXY数据库。
- **Windows服务** 在Windows系统中，用oradim工具为AUXY数据库创建一个名为OracleServiceAUXY的数据库服务。

任务3：执行RMAN TSPITR

最后，需要将SCOTT的employee表返回。引导RMAN在USERS表空间执行TSPITR。实际的命令清单非常简短，如下所示：

```

connect target sys/practice@practice
connect catalog rman817/rman@rcat
connect auxiliary sys/auxy@auxy
run {
set command id to 'Performing TSPITR';
set until time "11-JAN-2002 10:57:35";
allocate auxiliary channel d1 type disk;
recover tablespace users;
}

```

这个脚本文件打开一个到目标数据库（PRACTICE）、目录数据库（RCAT）和辅助数据库（AUXY）的连接。为辅助实例分配了一个通道并将V\$SESSION.CLIENT_INFO动态视图列更新为“Performing TSPITR”。然后，只需一个恢复表空间的命令，RMAN就可以执行了！

运行该脚本并在跟踪文件中检查输出：

```
LINUX> rman trace=tspitr.log
RMAN> @tspitr.rcv
```

我将指出一些有趣的输出，以便说明RMAN完成了所有的任务。首先应该明确，RMAN向辅助数据库重建与恢复目标数据库的一个子集。一旦辅助数据库恢复了，则恢复集表空间将被回传到目标数据库：

```
RMAN-08021: channel d1: restoring controlfile
RMAN-08505: output filename=/oradata/AUXY/control01.ctl
```

第一个生成的文件就是辅助数据库的控制文件。PRACTICE数据库控制文件的一个备份被在辅助参数文件指定的位置还原。在输出信息中显示控制文件还原所在的备份片。注意RMAN从PRACTICE数据库的备份创建了辅助实例，而不是从数据库本身。

```
# set a destination filename for restore
set newname for datafile 1 to
# set a destination filename for restore
set newname for datafile 2 to
# set a destination filename for restore
set newname for datafile 3 to
# set a destination filename for restore
set newname for datafile 7 to
"/oradata/AUXY/system01.dbf";
"/oradata/AUXY/rbs01.dbf";
"/oradata/PRACTICE/users01.dbf";
"/oradata/PRACTICE/users02.dbf";
# restore the tablespaces in the recovery set plus the auxiliary tablespaces
```

为了恢复USERS表空间，RMAN知道它必须在表空间的恢复集中要还原的数据文件，并必须还原随后的恢复所需的表空间和数据文件。Users01.dbf和users02.dbf属于USERS表空间，必须被还原。在本示例中，这两个数据文件是组成恢复集的两个数据文件。System01.dbf和rbs01.dbf也必须被还原（本例中组成了辅助数据集）。

```
sql clone "alter database datafile 1 online";
#online the datafiles restored or flipped
sql clone "alter database datafile 2 online";
#online the datafiles restored or flipped
sql clone "alter database datafile 3 online";
#online the datafiles restored or flipped
sql clone "alter database datafile 7 online";
# make the controlfile point at the restored datafiles, then recover them
recover clone database tablespace USERS, SYSTEM, RBS;
alter clone database open resetlogs;
```

这些命令将恢复需要回传USERS表空间到目标PRACTICE数据库的辅助数据库中的3个表空

间。这些命令不是恢复命令，它们是由RMAN在内存中为将要进行的恢复而生成的。因此，RMAN将生成要求完成工作的运行脚本。恢复之后，辅助数据库必须打开，这样USERS表空间才能使自己的字典元数据被导出。

```
RMAN-06162: sql statement: alter tablespace USERS offline for recover
...
RMAN-08089: channel dl: specifying datafile(s) to restore from backup set
RMAN-08523: restoring datafile 00001 to /oradata/AUXY/system01.dbf
RMAN-08523: restoring datafile 00002 to /oradata/AUXY/rbs01.dbf
RMAN-08523: restoring datafile 00003 to /oradata/PRACTICE/users01.dbf
RMAN-08523: restoring datafile 00007 to /oradata/PRACTICE/users02.dbf
RMAN-08023: channel dl: restored backup piece 1
...
RMAN-08024: channel dl: restore complete
```

现在这个输出应该比较熟悉了（考虑所有已经进行的练习）。RMAN的这些输出显示实际上还原的基本备份。注意，辅助集数据文件被保存在/oradata/AUXY目录下，但是恢复集数据文件被拷贝到目标数据库文件的位置。目标数据库的USERS表空间在还原数据文件之前是脱机状态。

```
RMAN-08054: starting media recovery
...
RMAN-03023: executing command: recover(4)
RMAN-08515: archivelog filename=/oradata/PRACTICE/archive/170.arc thread=1
sequence=170
...
RMAN-08055: media recovery complete
RMAN-03022: compiling command: alter db
RMAN-06400: database opened
```

还原之后，辅助数据库中的脱机表空间被恢复到停止点。增量备份也被应用到还原的数据文件上（上面显示的没有增量备份）。然后根据归档日志文件对辅助数据库重做。完成恢复之后，数据库打开，准备导出：

```
host 'exp userid =\'sys/auxy@auxy as sysdba\'
point_in_time_recover=y tablespaces= USERS file=tspitr_a.dmp';
...
host 'imp userid =\'sys/practice@practice as sysdba\'
point_in_time_recover=y file=tspitr_a.dmp';
```

RMAN生成export/import命令，从AUXY数据库向PRACTICE数据库传输新的恢复的USERS表空间。

如果该脚本运行成功而没有错误，TSPITR就成功了。在TSPITR之后，辅助数据库（AUXY）被关闭，目标数据库（PRACTICE）被打开，而恢复的表空间（USERS）则脱机。

任务4：确认恢复

为了确认USERS表空间按要求恢复了，可以将USERS表空间联机后检查数据库中表的状态。

- 类似于SCOTT.EMP表被删除之前的操作，检查USERS表空间中的对象和数据库情况。
- 查看SCOTT.EMP表存在而SCOTT.DEPT_COPY不存在。

- 检查是否可以在SCOTT.DEPT表中找到SUPPORT记录。
- 确认没有DEPT_COPY表，或者SCOTT.DEPT表中没有SUPPORT记录。
- 检查users表空间恢复之后其他表空间包含的数据。
- 检查TINA.DATE_LOG表中是否插入了一条未来17年日期的记录。

所有这些检查可以用以下SQL执行：

```
SQL> connect sys/practice@practice;
SQL> ALTER TABLESPACE USERS ONLINE;
SQL> describe scott.emp;
SQL> describe scott.dept_copy;
SQL> SELECT * FROM scott.dept;
SQL> SELECT max(create_date) FROM tina.date_log;
```

看来使用RMAN的TSPITR比手工操作要简便地多！

任务5：TSPITR之后的清理工作

一旦执行了TSPITR，则必须备份主数据库的数据文件。不能使用TSPITR之前的数据文件备份来恢复表空间，因为来自该备份的归档日志与新恢复的表空间并不对应。尽管整体数据库备份可以保护新恢复的USERS表空间，但可以用以下命令只备份一个表空间：

```
RMAN> run {
2 allocate channel d1 type disk;
3 backup tablespace USERS
4 format '/oradata/PRACTICE/backup/ch17/db_%d_%s_%p_%t';
5 }
```

还有最后一件事：不再需要在磁盘上保存辅助数据库文件。这些文件已经完成了使命，所以可以从/oradata/AUXY目录中删除。这样可以删除数据文件、控制文件、重做日志文件和RMAN进行TSPITR时需要的临时文件。也可以删除为辅助实例创建的数据库初始化文件、口令文件和其他路径。

17.2 疑难解答

如果在这个RMAN练习时遇到问题，可以参考下面问题列表以找到解决方法：

1) RMAN-11003: failure during parse/execution of SQL statement:

```
alter system archive log current
```

RMAN-11001: Oracle Error: ORA-01109: database not open TSPITR期间的目标数据库（PRACTICE）必须打开。目标数据库要切换日志并包含元数据。打开PRACTICE数据库并重新开始TSPITR。

2) ORA-27086: skgfglk: unable to lock file - already in use 检查在TSPITR生成的文件是否正确。如果RMAN不能加锁的一个文件在控制文件中，则确认以正确的参数文件启动了AUXY并更改了这个控制文件参数。如果RMAN不能加锁的一个文件是数据文件，则检查在AUXY参数文件中是否设置了正确的DB_NAME_FILE_CONVERT_FILE参数。如果有问题的文件是一个联机的重做日志文件，则再次检查LOG_FILE_NAME_CONVERT参数设置是否正确。

3) RMAN-11001: Oracle Error: ORA-01190: controlfile or data file 3 is from

before the last RESETLOGS

ORA-01110: data file 3: 'd:/oradata/PRACTICE/users01.dbf' 如果停止点设置的不恰当, 则可能遇到这个错误。修改该数值并重新启动TSPITR。被还原的数据文件或被应用的控制文件来自不同的时间, 并且是在最后RESETLOGS打开之前。

4) ORA-01124: cannot recover data file 7 - file is in use or recovery

ORA-01110: data file 7: 'd:/oradata/PRACTICE/users02.dbf' 没有设置停止点或者设置的时间过长。TSPITR一定是不完全恢复。为恢复使用set until选项或者在恢复表空间命令之后增加一个until关键字。

17.3 小结

对比第9章和本章的内容, 用户管理的TSPITR与本章中的服务器管理的TSPITR相比需要更多的工作。尽管用户管理和服务器管理的TSPITR都需要建立数据库路径和相应的支撑文件以及用于数据库生成的一个备份, 但RMAN却完成了大多数数据库任务。表17-1总结了用户管理和服务器管理TSPITR的差别。

表17-1 比较用户管理和服务器管理的表空间恢复

用户管理TSPITR	服务器管理TSPITR
配置副本数据库(路径、参数文件、口令文件、Windows服务); Net8配置可选	配置副本数据库(路径、参数文件、口令文件、Windows服务); 需要Net8配置
采用用户管理的打开的或关闭的备份来备份主数据库的辅助和恢复集数据文件。备份需要与所有数据文件一样大的磁盘或磁带空间	采用服务器管理的打开的或关闭的备份或映像拷贝来备份目标数据库的辅助和恢复集数据文件。备份需要的磁盘或磁带空间比所有的数据文件的空间少。映像拷贝需要与所有数据文件一样大的磁盘或磁带空间
通过操作系统的命令删除备份文件	还原辅助数据库数据文件
在辅助数据库位置上创建主控制文件的拷贝文件的备份	在辅助控制文件的位置上还原控制
执行手工不完全恢复并以RESETLOGS打开Export/import恢复集表空间	执行手工不完全恢复并以RESETLOGS打开处理恢复集表空间的导出/导入
为恢复的数据文件重新命名或在操作系统拷贝文件	切换目标数据库使用恢复的数据文件
在TSPITR之前从主数据库中删除恢复的表空间	切换数据文件而不需要删除表空间
目标和辅助数据库可以保存在不同的服务器上	目标和辅助数据库必须保存在同一服务器上

在本章中, 介绍了如何使用RMAN进行TSPITR操作, 学习了如何用RMAN对一个辅助数据库进行恢复表空间, 然后将这个表空间再反插到主数据库中。为了获得更多的关于RMAN TSPITR的信息, 请参看《RMAN Oracle8i Recovery Manager User's Guide and Reference》一书中的第8章内容。

习题

这是本书最后一组问题, 检查对本章RMAN TSPITR知识的掌握情况。

1. 用RMAN复制命令来执行表空间的时间点恢复。
 - A. True
 - B. False
2. 在RMAN TSPITR期间, 需要在辅助数据库的参数文件中设置什么参数来控制辅助数据库联机重做日志文件的位置?
 - A. db_file_name_convert
 - B. log_file_name_convert
 - C. The auxiliary database doesn't use online log files
 - D. log_archive_dest
3. 本章的练习中, 恢复的表空间是USERS表空间。还有哪些表空间也必须与这个数据表空间一起被恢复? (多项选择)
 - A. SYSTEM
 - B. RBS
 - C. TEMPORARY
 - D. INDEXES
4. 考虑进行RMAN TSPITR时, 必须和用户管理的TSPITR一样, 需要进行处理并解除潜在的数据库和对象的依赖性关系。
 - A. True
 - B. False
5. RMAN TSPITR执行了哪些任务? (多项选择)
 - A. 向辅助数据库还原必需的目标数据文件。
 - B. 在辅助数据库上恢复还原的数据库数据文件。
 - C. 切换目标数据库来应用恢复的数据文件。
 - D. TSPITR完成之后删除辅助数据库文件。

答案

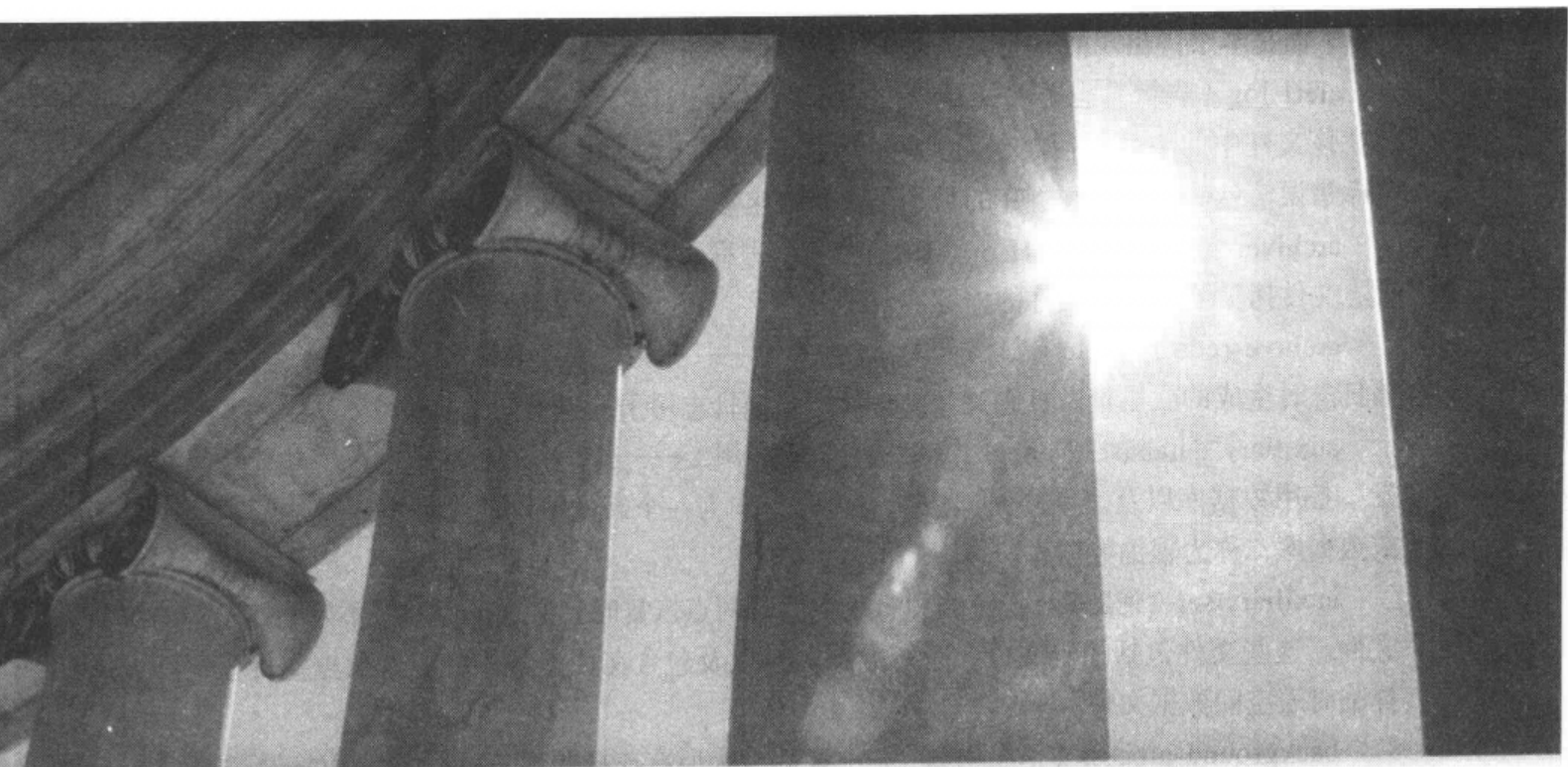
1. False。尽管RMAN执行了一些复制操作, 但用连接到辅助数据库和恢复表空间命令的联合使用来指示TSPITR。

2. B。一旦辅助集和恢复集被还原, 则辅助数据库就要以resetlogs选项被打开。在辅助数据库的恢复期间, 不再需要新生成的联机重做日志(因为resetlogs生成了日志)。当辅助数据库以RESETLOGS打开时用log_file_name_convert在其他路径下生成新的联机重做日志。

3. A、B。在TSPITR期间, system表空间和非系统的rollback段被包含在辅助集中。

4. True。如果在用户管理的TSPITR恢复集中的对象不能通过Export/import传送, 用户管理的TSPITR将失败, 类似情况下RMAN TSPITR也将失败。同样地, 恢复集内外表空间之间的依赖性也将使RMAN TSPITR失败。

5. A、B、C。RMAN并没有删除辅助数据库文件(控制文件、恢复集和辅助集的数据文件、重做日志以及导出的转储文件)。



第四部分 附 录

附录A 术 语 表

下面是本书中出现的有关关键词语的定义。

alert log (告警日志)——一个文本文件，Oracle持续不断地将重要的信息记录在其中。记录在该文件中的事件包括数据库关闭、数据库启动、增加或删除表空间或数据文件以及一些数据库错误信息。这个文件所在的目录是由初始化参数BACKGROUND_DUMP_DEST定义的。

archiver process (ARCH or ARCn) (归档进程)——一种Oracle后台进程，负责向归档重做日志文件拷贝联机重做文件的内容。

archive redo log file (归档重做日志文件)——归档进程(日志切换时发生)或用户要求归档日志时生成的联机重做日志文件的拷贝。归档日志用于从备份恢复数据库。

auxiliary database/instance (辅助数据库/实例)——当应用RMAN中的一些功能(复制数据库、备用数据库以及表空间时间点恢复)时，作为一个新的或临时的数据库生成的第二个实例或数据库，称为辅助数据库。

auxiliary set (辅助集)——当执行表空间时间点恢复时，必须还原一些不属于恢复集的数据文件。这些文件包括控制文件、系统数据文件以及包含在回退段中的所有数据文件，还可以选择临时表空间数据文件。

background process (后台进程)——在后台为所有连接的用户执行任务的Oracle进程。后台进程包括数据库写入器(DBWR或DBWn)、系统监视器(SMON)、进程监视器(PMON)、归档器(ARCH或ARCn)、日志写入器(LGWR)以及检查点进程(CKPT)。

backup (备份)——整体数据库或部分数据库的拷贝。备份包括控制文件、表空间、数据文件、数据库对象或归档重做日志文件。

backup piece (备份片)——当创建备份时由RMAN创建的一种物理文件，其使用的格式只适用于RMAN。这种文件可以存储在磁盘或磁带上。

backup set (备份集)——RMAN备份片的逻辑集合。当RMAN进行备份时，生成由一个或多个备份片组成的一个或多个备份集。备份集包含控制文件和数据文件(或归档日志文件)。备份集不会同时包含数据文件和归档日志文件。

buffer cache (缓冲存储区)——系统全局区域中的一部分区域，用来存储来自数据库的数据文件。数据块被从磁盘上读取并在被浏览或修改之前先存储到缓冲存储区中。

channel (通道)——RMAN和目标数据库之间的通讯通道。通道通常用于查询目标数据库以及在还原和恢复操作期间向被选择的备份介质(磁盘或磁带)传送数据。

change vector (更改矢量)——对数据块进行更改，将其从一个状态转变为另一个状态。更改矢量，有时也称为重做矢量，在恢复期间被直接应用于数据块，而且是在实际数据块在缓冲区中被更改之前创建。

checkpoint ((CKPT), 检查点)——在控制文件和数据文件中存储的一种结构体，说明对于给定SCN的所有更改都被记录在数据文件当中了。在检查点期间，在缓冲区中的已经修改的数据块被写到数据文件之前，日志缓冲中的重做记录被写到当前的联机重做日志中。

checkpoint process (检查点进程)——Oracle后台进程, 向控制文件及其文件首部记录检查点信息。这个进程也通知数据库写入器将缓冲区中的脏数据向磁盘中更新。从Oracle 8i开始, 每三秒向控制文件记录一次检查点。

clone (克隆)——重新命名的数据库拷贝。

closed/cold backup (关闭状态/冷备份)——当数据库关闭时进行的数据文件或控制文件备份。这种备份也被称为一致性备份。

complete recovery (完全恢复)——从备份中还原数据库或数据库子集, 并应用所有的归档日志文件和联机日志文件。所有的数据块更改都将被应用到还原的数据文件中。在RMAN恢复期间, 还需要应用增量备份。

control file (控制文件)——属于数据库的数据库文件, 存储所有关键的结构信息, 包括数据文件、联机日志和归档重做日志文件以及检查点信息。如果使用RMAN进行备份, 则备份信息也将存储在控制文件中。

corruption (讹误)——数据块从正确的状态变到其他状态或Oracle内部结构不正确。为了解决讹误问题, 需要经常进行文件恢复, 或者使用Oracle 9i中的介质恢复功能。

crash recovery (灾难性恢复)——当实例突然崩溃时, 需要在下一次数据库启动时自动地应用所有联机重做日志文件中的重做信息。灾难恢复时不需要归档的重做日志文件。这也叫作实例恢复。

database writer (DBWR or DBWn), (数据库写入器)——将缓冲区中更改过的数据块写到数据文件中的一个Oracle后台进程。数据库写入器在事务提交时并不将缓冲区中的数据写入文件; 大多数情况下是当缓冲区中新的数据块需要空间时, 或者在进行检查点时才写入“脏”的缓冲区。

data block (数据块)——Oracle数据库中最小的单元空间, 这个空间的大小由初始化参数DB_BLOCK_SIZE决定。

data block address (DBA), (数据块地址)——数据库中数据块的地址。其编码表示文件编号和文件中的块编号。

datafile (数据文件)——Oracle生成的物理文件, 由一系列数据块组成。数据文件在一个数据库中只能属于一个表空间。

Data Guard (数据卫士)——在8.1.7版本的一些平台上起用的新改进的备用数据库版本, 在9i中又有了显著提高。对备用数据库的明显增强主要在传送和应用归档日志文件方面。

data definition language (DDL), (数据定义语言)——定义、创建、删除及维护模式对象的SQL语句。CREATE TABLE命令就是一个DDL实例。

data manipulation language (DML), (数据操作语言)——处理存储在数据库中数据的SQL语句。这样的语句有INSERT、UPDATE和DELETE等。

duplicate database (复本数据库)——与原始数据库具有同样或不同名称的数据库拷贝。

export (导出)——Oracle提供的一种工具, 可以将数据库或其子集的内容提取到一个二进制格式的文件中, 该文件可以用来向其他数据库传递对象和数据。

image copy (映像拷贝)——RMAN可以生成数据文件、归档日志文件等的精确映像拷贝。

这类似于操作系统拷贝命令生成的文件拷贝。

import (导入)——Oracle提供的一种工具，可以用来向数据库中插入包含在Oracle二进制导出文件中的对象。

incomplete recovery (不完全恢复)——数据库或其子集从备份还原并应用了并非全部的重做信息，其中一些改变并没有被应用到数据库。因此，存储在重做文件中的一些更改并没有在数据库中得以恢复。为了在不完全恢复之后打开数据库，必须重置联机重做日志。

instance (实例)——当数据库首次启动时创建的Oracle后台进程以及系统全局区域 (SGA) 的组合物。

log buffer (日志缓冲区)——SGA中的一部分内存区域，用来保存数据库改变时的重做记录。当日志缓冲区有1/3部分被填满，或要求提交时，或生成了1MB的重做信息时，或要求日志切换时，日志缓冲区中的信息都要更新到当前的联机重做日志文件中。

logical backup (逻辑备份)——没有物理拷贝数据库文件而通过创建数据库内容产生的备份。Oracle的Export工具可以用于对数据库或数据库的子集进行逻辑备份。

LogMiner——一种Oracle工具，可以从Oracle数据库的重做日志文件中读取信息，重新构建用于生成数据块更改和撤销更改的SQL语句。从Oracle 8i开始引入该工具，但从Oracle 8.0就可以用来浏览日志文件。

log sequence number (日志序列号)——惟一表示一个重做日志文件的编号。当第一次生成数据库时，第一个重做日志的序列号就是1。当日志切换时，则该日志序列号就增加1，变为2。

log switch (日志切换)——日志写入进程停止向一个联机重做日志文件写入并开始写入另外一个联机重做日志文件。当前的联机重做文件被写满时，或用户提交ALTER SYSTEM SWITCH LOGFILE命令时，进行日志切换。

log writer (LGWR), (日志写入器)——Oracle后台进程，负责将日志缓冲区中的重做信息写到当前联机重做日志中。

Media Management Layer (MML), (介质管理层)——第三方软件，负责处理序列介质（比如磁带等）数据的输入与输出操作。

NOLOGGING——当执行直接加载插入操作时进行的日志重做模式。当使用NOLOGGING时，重做信息的数量明显减少，但必须在插入操作后备份受到影响的对象，以便发生失败时的恢复操作。直接加载操作包括CREATE TABLE...AS SELECT, CREATE INDEX, ALTER TABLE MOVE/SPLIT PARTITION和ALTER INDEX...REBUILD/SPLIT PARTITION。

online redo log file (联机重做日志文件)——当刷新日志缓冲区中的内容时，由LGWR连续地执行向两个或更多日志文件集合的写入操作。这些文件以循环方式使用。例如，如果有两组联机重做日志文件，当第二组写完后，将向第一组中写入并覆盖原来的内容。

open/hot backup (打开状态/热备份)——当数据库打开并可用时进行的备份。如果数据库运行在ARCHIVELOG模式，则只能进行这种类型的备份。也叫做不一致备份。

physical backup (物理备份)——数据库文件被物理地从一个地方拷贝到另一个地方。可以使用操作系统的工具或者RMAN的映像拷贝功能来完成这种备份。

primary database (主数据库)——在数据库复制或创建备用数据库时被拷贝的初始或源数据

库。在表空间时间点恢复期间，复制和恢复主数据库部分。

process global area (PGA), (进程全局区域)——当服务进程连接到数据库时，为其分配的私有内存区域。这部分内存区用于排序、会话信息以及LogMiner等。

Process Monitor (PMON), (进程监视器)——处理用户进程恢复失败的后台进程。该进程可以清理并释放失败进程占有的所有资源。

proxy copy (代理拷贝)——一种备份方法，控制管理、备份和还原数据库文件都由介质管理层处理。

recovery (恢复)——应用增量备份(使用RMAN时)和重做信息来重新构建对数据块的更改的过程。

recovery catalog (恢复目录)——供RMAN使用的信息库，其中记录了Oracle数据库以及这些数据库所进行的备份的信息。这是RMAN体系结构中的一个可选部分。其中一些信息也被记录到目标数据库控制文件中。

Recovery Manager (RMAN), (恢复管理器)——用于备份、还原和恢复Oracle数据库的Oracle工具。

recovery set (恢复集)——在表空间时间点恢复期间，执行不完全恢复以还原所需要的数据或数据对象的数据文件集。恢复集不包含任何系统数据文件或回退段的数据文件。

redo record (重做记录)——对数据库进行原子更改的一组重做矢量。重做矢量分组为记录并在数据库恢复期间应用。如果在恢复期间应用了一个重做矢量，则同一记录中的所有矢量也都要应用。

Remote File Server (RFS), (远程文件服务器)——Oracle后台进程，负责在备用管理恢复期间通过Net8接受并创建来自主数据库的备用数据库和归档日志文件。

RESETLOGS——执行不完全恢复后打开数据库的一种模式，将重新创建或重新修改当前的联机重做日志文件、或删除这些文件的所有内容。生成数据库的一个新场景，应该进行备份。当数据库被打开而且数据被更改后，想通过RESETLOGS操作进行恢复几乎是不可能。

Restore (还原)——从备份替换丢失的或损坏的数据库文件。一旦文件被还原，则可以被恢复以重新构建数据块的更改。

resync (再同步)——通过存储在目标数据库控制文件中的信息更新恢复目录的过程。这个过程在RMAN的一些操作(如BACKUP、COPY、RESTORE和RECOVER等)之前自动进行。也可以手工向RMAN发布RESYNC CATALOG命令来执行该过程。

rollback (回退)——回退段用于撤销任何没有提交的数据更改。当数据库使用重做日志前滚时，由Oracle自动执行该过程，或者用户在事务提交之前手工发布ROLLBACK命令。

rollback segment (回退段)——包含的数据块的值与修改前的一样。回退段用于回退未提交的事务，以及准许对数据库进行一致性读取，这样用户看到的数据与其事务开始前是一样的，而不管这些数据在其事务开始时是否被更改了。

ROWID (行标识)——在一个对象中用来指示行的结构。ROWID将说明文件中对象的标识、文件号、数据块的编号，以及数据块中行的位置。

segment (段)——在一个表空间中逻辑上成组的一个或多个域的集合。段分为四种类型：数

据段、索引段、回退段和临时段。当创建一个表时，将要分配一个或多个域来形成一个数据段。

server-managed recovery（服务器管理恢复）——由Oracle的RMAN工具进行管理的备份、还原和恢复。

shared pool（共享池）——在系统全局区域中的一部分内存，被分为库缓冲（包含共享的SQL区域）和字典或行缓冲（包含数据字典）。

standby database（备用数据库）——用于主数据库失效时的一个数据库的精确拷贝。备用数据库通过应用来自主数据库的归档重做日志文件而与主数据库保持同步。当主数据库失效时，备用数据库就被激活并作为新的主数据库。

standby managed recovery（备用数据库管理恢复）——备用数据库自动应用来自主数据库的归档重做日志。

standby manual recovery（备用数据库手工恢复）——必须手工向备用数据库应用来自主数据库的归档重做日志。

system change number (SCN),（系统更改号）——代表了数据库在某一时间点的一致性提交版本的编号，该编号对数据库而言可以被当作时钟。每个提交的事务都要得到一个惟一的SCN。

System Global Area (SGA),（系统全局区）——可以被连接到数据库上的所有用户访问的共享内存。系统全局区由缓冲存储区、共享池和日志缓冲以及Oracle RDBMS使用的许多内存结构组成。

System Monitor (SMON),（系统监视器）——Oracle后台进程，执行许多数据库的工作，包括数据库启动时的灾难恢复、清除不再使用的临时段、为数据字典需要管理的表空间拼接自由域、从数据字典中清理删除的对象。

tablespace（表空间）——相关对象的逻辑存储区域。每个数据库对象都存储在表空间中，表空间由一个或者多个数据文件组成。

tablespace point-in-time recovery (TSPITR),（表空间时间点恢复）——将一个或多个非系统表空间恢复到与数据库其他部分不同的时间点上的过程。这个过程可以使用用户管理或服务器管理技术，通过还原恢复集和辅助集文件来完成。

target database（目标数据库）——进行备份、还原和恢复的主数据库。当RMAN创建副本数据库、备用数据库或执行表空间时间点恢复的辅助数据库时，都要查询该数据库。

transportable tablespace（可迁移表空间）——Oracle的一个功能，允许表空间从一个数据库上迁移或拷贝到另一个具有同样硬件体系结构和操作系统的数据库上。使用导出工具来传送包含在被迁移表空间中的对象的元数据库。带有表空间数据文件的导出文件被传送到另一个数据库中。导入工具用来导入元数据，使新迁移的表空间可以被目的数据库访问。

user-managed recovery（用户管理恢复）——备份、还原和恢复的方法都由DBA自己管理。在这些方法中不使用RMAN，而依靠操作系统工具来备份和还原数据库。DBA必须手工应用重做日志文件并打开数据库。



附录B Oracle 9i 中的 RMAN

本书中，考虑到大多数读者仍然在使用Oracle 8i，因此我选择它作为练习和讨论的版本。但是，许多Oracle用户已经移植到Oracle 9i或正计划移植。本书中讨论的大部分内容也同样适合于Oracle 9i。随着9i版本的发行，RMAN有了明显的改善。在有关RMAN的章节中，我讨论了一些9i中的新特性。另外，本附录与第11、12、13和14章对照讲述怎样使用9i的新特性。虽然本附录并不全面，但提供了一些对Oracle 9i RMAN的改进的说明。

B.1 RMAN的配置

用RMAN配置目录在两个版本中非常相似。在Oracle 8i RMAN中，你必须在命令行提供nocalog选项或者在脚本中连接到目录。在9i中，nocalog被设定为默认值（不管你是否在命令行提供这一参数）。

9i中可以利用的一个有用配置特性是为备份操作配置永久通道。在第12章的备份脚本中，每一个备份必须分配一个通道并格式化该通道。格式化的结果是指定磁盘上的目录和文件名或磁带上一个文件名。在9i中，可以定义磁盘或磁带默认通道的特性，这样备份、还原和恢复操作的执行就可以使用预先配置的通道。这些默认的定义通道可以被备份脚本中外部分配的通道所取代。使用第12章的例子来说明它。

在第12章中，每一个备份脚本在/oradata/PRACTICE/backup/ch12目录下写入备份信息，必须分配磁盘类型的通道并定义备份的格式。在9i中，可以使用RMAN的配置命令告诉RMAN备份的默认设备是磁盘类型以及备份位置是指定目录，下列命令可以完成这个任务：

```
LINUX> rman
RMAN> connect target sys/practice@practice9i
RMAN> connect catalog rman901/rman@rcat
RMAN> configure default device type to disk;
RMAN> configure channel device type disk format =
2      '/oradata/PRACTICE/backup/ch12/db_%d_%s_%p_%t';
```

要使用配置参数，必须连接到目标数据库或恢复目录。因为配置信息存放在目标数据库的控制文件中，这些信息也存放在恢复目录中。

可以定义其他通道参数，如备份块的最大容量。RMAN可以以并行方式备份数据库的各个部分。假定你为备份留出两个文件系统，这些系统中的最大文件为2GB。除非另有指定，否则RMAN将创建备份至/u01和/u02并且不会让备份块大于2GB。一旦连接到目标数据库或恢复目录，就发布下列命令：

```
RMAN> configure device type disk parallelism 2;
RMAN> configure channel 1 device type disk
2>   format = '/u01/oracle/backup/disk1/db_%d_%s_%p_%t'
3>   maxpiecesize 2G;
RMAN> configure channel 2 device type disk
2>   format = '/u02/oracle/backup/disk1/db_%d_%s_%p_%t'
3>   maxpiecesize 2G;
```

在备份中，RMAN使用两个定义好的通道并调用它们：ORA_DISK1和ORA_DISK2，它们分别写入/u01和/u02。可以按大致相同的方式完成磁带配置。

可以配置RMAN，在每次备份命令发布时创建控制文件的备份。创建的控制文件备份并不包含在备份包中，而是一个二进制文件拷贝，以备在恢复目录和当前目标控制文件丢失时的还原和恢复中使用。通过下列命令可以完成创建控制文件的备份（假定已经连接上了目标数据库或恢复目录）：

```
RMAN> configure controlfile autobackup on;
RMAN> configure controlfile autobackup format for device type
2> disk to '/u01/oracle/backup/disk1/cf_%f';
```

在备份操作中，也可以使用configure命令再定位RMAN在备份操作期间将要使用的快照控制文件，方法如下：

```
RMAN> configure snapshot controlfile name
2> to '/u01/oracle/backup/disk1/snap.f';
```

可以通过使用RMAN，或者通过连接恢复日志并发布SQL命令来浏览每一数据库的RMAN配置信息。下面是用RMAN方法实现的例子：

```
LINUX> rman
RMAN> connect target sys/practice@practice9i
RMAN> connect catalog rman901/rman@rcat
RMAN> show all;
```

下面是使用SQL方法的例子：

```
LINUX> sqlplus /nolog
SQL> connect rman901/rman@rcat
SQL> SELECT dbid, dbinc_key, d.name, c.name, c.value
2 FROM rc_database d, rc_rman_configuration c
3 WHERE d.db_key = c.db_key
4 ORDER BY d.name, d.dbinc_key, c.conf#;
```

使用RMAN或SQL，你可以查看当前RMAN的配置设置。

B.2 RMAN backup命令

在9i中，因为提供了新的配置特性，所以备份脚本变得简单了。同时，backup命令（和其他命令）也不必包含在运行块中。例如，第12章中，学习了如何完成完整数据库的增量级别为零的基准备份。下面是8i中脚本命令的内容。运行备份前你必须分配通道。既然你希望将备份块存储在RMAN默认位置（\$ORACLE_HOME/dbs）以外的其他目录，那么必须定义通道的格式或备份命令的格式。

```
run {
  allocate channel d1 type disk;
  backup
    incremental level = 0
    database
    format '/oradata/PRACTICE/backup/ch12/db_%d_%s_%p_%t'
```

```

tag = 'WHOLE_INC0';
}

```

在Oracle9i中, 如果默认通道已经按照刚才概括的步骤预先配置好, 同样的备份脚本可以更加小巧和简便。运行这些命令前, 你仍然必须连接目标数据库或恢复目录。

```

RMAN> backup incremental level 0 database tag = 'WHOLE_INC0';

```

这里backup命令将使用预配置通道定位备份块。级别1的增量备份可以用下列命令完成:

```

RMAN> backup incremental level 1 database tag = 'WHOLE_INC1';

```

Oracle建议只要增加了数据文件, 就应备份关键数据库部件。这些部件包括控制文件和新加入的数据文件。在9i中, backup命令中的NOT BACKED UP关键字提供了一个方便的方法来完成这一任务, 命令如下:

```

RMAN> backup incremental level 0 database NOT BACKED UP;

```

运行这个命令将告诉RMAN备份所有自创建以来没有备份过的数据文件。如果你配置了自动控制文件备份, 那么也将备份控制文件。

如果长时间的备份在没有完成的情况下中断了怎么办? 在Oracle8i中, 你不得不重新开始备份。在Oracle9i中, 你可以告诉RMAN重新开始备份但仅仅备份那些第一次备份中未完成备份的文件(假定你完成一个日常备份), 命令如下:

```

RMAN> backup incremental level 1 cumulative database
2> NOT BACKED UP since 'SYSDATE - 1';

```

在Oracle8i中, 你必须在备份完数据文件后运行单独的backup命令来备份归档日志文件。在9i中可以使用backup命令中PLUS ARCHIVELOG选项来简化这一过程。RMAN看到这一选项时, 会在发布ALTER SYSTEM SWITCH LOGFILE命令确保联机日志文件归档后自动发布BACKUP ARCHIVELOG ALL命令。归档日志在各自的备份包中备份, 在数据文件备份前与已经备份的数据文件分离开。数据文件备份后, RMAN将发布另一个ALTER SYSTEM SWITCH LOGFILE命令来备份所有在数据文件备份中产生的归档日志文件。下面是创建一个所有归档日志的增量备份的示例:

```

RMAN> backup database incremental level 1 cumulative plus archivelog;

```

RMAN也加入了备份归档日志的其他功能, 列举如下:

- 一旦建立了归档日志备份, 自动从所有归档日志目标位置删除磁盘上的归档日志文件。在9i发布以前, 你必须手工操作, 确保所有的归档日志备份并从目标位置删除。
- 备份归档日志之前自动归档当前联机重做日志文件。这样就确保了每次备份都包含了数据库创建的最新的日志文件。
- 当一个归档日志文件丢失或损坏时, 自动搜索所有其他归档日志目标位置查找文件的当前版本。
- V\$ARCHIVED_LOG视图中的BACKUP_COUNT列跟踪RMAN备份归档日志文件的次数。这使监视当前归档日志文件的备份状态变得极其容易。

B.3 RMAN 目录维护

在9i中, Oracle改善了列表选项。在8i中, 你必须逐项读完RMAN所有的显示内容。在这么

多输出结果中寻找一小部分你希望确定的备份会令人筋疲力尽。Oracle9i中一个主要的改善是你可以看到备份以高亮度显示。运行下例看看输出结果如何变得更醒目：

```
RMAN> list backup summary;
RMAN> list backup by file;
RMAN> list backup of datafile 1 summary;
```

许多Oracle用户在磁盘上保留部分备份，在磁带上保留部分备份。（例如你希望将上星期的备份放在磁盘上，而把以前的备份放在磁带上。）在9i中，你可以创建当前RMAN磁盘备份内容的磁带备份。当你使用RMAN执行还原和恢复操作时，它将试图还原磁盘备份。如果没有磁盘备份，将还原磁带备份。下面的命令完成磁带备份：

```
RMAN> backup device type sbt backupset
2> created before 'sysdate-7' delete input;
```

Oracle8i中要注意从MML和恢复日志中删除旧的备份。在Oracle9i中，你可以指示RMAN使用保留策略，同典型的磁带管理软件的保留策略一样。RMAN备份时，它应用保留策略确保你已经使用configure命令指定了不再使用的备份或拷贝。不再需要的备份自动在恢复目录中标记为“过时的”。然后你可以报告“备份已经过时”并删除它们。RMAN不会自动删除过时的备份，你必须发布命令才能完成这项工作。定义保留策略基于你希望恢复的天数或你拥有的冗余备份数。这一时间周期被称为恢复视窗（Recovery Window）。例如，如果你希望RMAN自动删除最近21天以前恢复数据库不再所需要的备份，可以发布下列命令：

```
RMAN> configure retention policy to recovery window of 21 days;
报告过时备份然后从恢复目录和介质中删除它们，可以发布下列命令：
RMAN> report obsolete; # will use the configured retention policy
RMAN> allocate channel for maintenance device type disk;
RMAN> delete obsolete;
RMAN> release channel;
```

B.4 RMAN 还原与恢复

RMAN也改变了备份被还原的方式，可以使用更优的方法完成它。9i发布前，RMAN自动还原恢复全部或部分数据库所需要的数据文件。如果数据文件自上次还原以来已经存在于磁盘上，则RMAN将用还原的文件覆盖以前的文件。在9i中，RMAN很智能，直接使用已经存在的恢复所需的文件而不从备份中还原。

在Oracle9i中，你可以完成块介质恢复。RMAN在数据块层次完成数据文件的操作。RMAN读取每个数据块，判断它是否应该备份（按照增量备份策略），并将其写入备份片。在9i发布前，RMAN还原并恢复的最小单位是单个的数据文件。随着9i的发布，现在可以还原并恢复单个的数据块。这种操作是有必要的，例如，一个2GB数据文件中的一小部分数据块被损坏，使用归档和联机重做日志文件仅仅把受影响的数据块恢复到以前的状态肯定比还原整个2GB数据文件并应用重做日志要快得多。

RMAN要求被恢复的数据块的完整备份存放在合适的位置，并且执行完全恢复。就像数据库打开时恢复单个数据文件一样，所有重做日志（包括当前重做信息）必须被应用到恢复数据

块中。执行块介质的恢复，使用下面命令：

```
RMAN> BLOCKRECOVER DATAFILE 4 BLOCK 7 DATAFILE 4 BLOCK 15;
```

小结

以上是对Oracle9i中RMAN新改善特性的初步体会。我希望这可以成为你从Oracle8i RMAN配置升级到9i的初步指导。如果需要了解9i中更多关于RMAN的内容，可以查阅Oracle 9i的手册：《Oracle9i Recovery Manager User's Guide Release 1(9.0.1)》和《Oracle9i Recovery Manager Reference Release 1(9.0.1)》。最好从《Oracle9i Recovery Manager User's Guide》开始部分的“*What's New in Recovery Manager?*”小节开始阅读。

块中。执行块介质的恢复，使用下面命令：

```
RMAN> BLOCKRECOVER DATAFILE 4 BLOCK 7 DATAFILE 4 BLOCK 15;
```

小结

以上是对Oracle9i中RMAN新改善特性的初步体会。我希望这可以成为你从Oracle8i RMAN配置升级到9i的初步指导。如果需要了解9i中更多关于RMAN的内容，可以查阅Oracle 9i的手册：《Oracle9i Recovery Manager User's Guide Release 1(9.0.1)》和《Oracle9i Recovery Manager Reference Release 1(9.0.1)》。最好从《Oracle9i Recovery Manager User's Guide》开始部分的“*What's New in Recovery Manager?*”小节开始阅读。

块中。执行块介质的恢复，使用下面命令：

```
RMAN> BLOCKRECOVER DATAFILE 4 BLOCK 7 DATAFILE 4 BLOCK 15;
```

小结

以上是对Oracle9i中RMAN新改善特性的初步体会。我希望这可以成为你从Oracle8i RMAN配置升级到9i的初步指导。如果需要了解9i中更多关于RMAN的内容，可以查阅Oracle 9i的手册：《Oracle9i Recovery Manager User's Guide Release 1(9.0.1)》和《Oracle9i Recovery Manager Reference Release 1(9.0.1)》。最好从《Oracle9i Recovery Manager User's Guide》开始部分的“*What's New in Recovery Manager?*”小节开始阅读。