

第1章 从零开始——Oracle 9i 基础

本章目的在于帮助读者从零开始快速掌握 Oracle 9i 的基础知识。

1.1 关系型数据库系统简介

1.1.1 什么是关系型数据

关系型数据是以关系数学模型来表示的数据。关系数学模型中以二维表的形式来描述数据，如表 1.1 和表 1.2 所示。

表 1.1

研究生信息二维表

学号	姓名	专业	导师编号
2002080520	王海	计算机安全	200201
2002080521	李东	软件工程	200209

字段（列）

记录（行）

表 1.2

导师信息二维表

编号	姓名	职称	职务
200201	刘阳	博导	室主任
200209	海涛	硕导	系主任

1.1.2 什么是关系型数据库

1. 什么是主码（主键）

能够唯一表示数据表中的每个记录的【字段】或者【字段】的组合就称为主码。

2. 什么是外码（外键）

表 1.2 的【编号】字段和表 1.1 的【导师编号】字段是对应的。表 1.2 中的【编号】字段是表 1.2 的主码。表 1.2 中的【编号】字段又可以称为是表 1.1 的外码。

1.1.3 什么是关系型数据库系统

一个完整的关系型数据库系统包含 5 层结构，如图 1.1 所示。

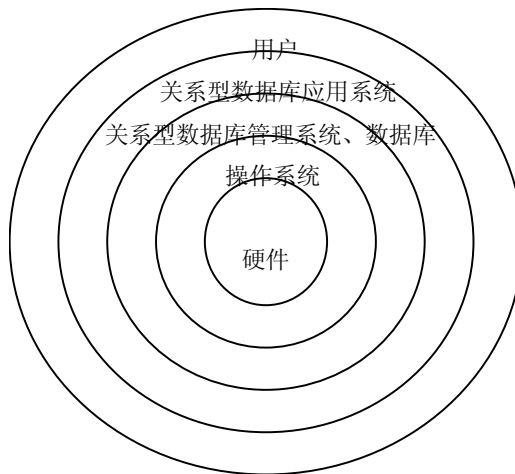


图 1.1 关系型数据库系统的层次结构

1. 硬件

硬件指安装数据库系统的计算机，包括两种。

- ☐ 服务器
- ☐ 客户机

2. 操作系统

操作系统指安装数据库系统的计算机采用的操作系统。

3. 关系型数据库管理系统、数据库

关系型数据库是存储在计算机上的、可共享的、有组织的关系型数据的集合。关系型数据库管理系统是位于操作系统和关系型数据库应用系统之间的数据库管理软件。

4. 关系型数据库应用系统

关系型数据库应用系统指为满足用户需求，采用各种应用开发工具（如 VB、PB 和 Delphi 等）和开发技术开发的数据库应用软件。

5. 用户

用户指与数据库系统打交道的人员，包括如下 3 类人员。

- ☐ 最终用户
- ☐ 数据库应用系统开发员
- ☐ 数据库管理员

1.1.4 什么是关系型数据库管理系统

1. 数据定义语言及翻译程序 DDL
2. 数据操纵语言及编译（解释）程序 DML
3. 数据库管理程序

1.2 目前主流的两类关系型数据库系统

1.2.1 桌面关系型数据库系统

1.2.2 网络关系型数据库系统

在网络关系型数据库系统中，有 3 个特别重要的概念。

1. 数据库服务器

逻辑上的服务器指的是安装在计算机上提供一些基于网络环境的应用的软件。

2. 管理客户机

逻辑上的管理客户机是指对数据库进行管理的软件。

3. 端口

为了区分这些不同的逻辑服务器，使用了称为端口的概念。

1.3 网络关系型数据库的代表 Oracle 9i

1.3.1 Oracle 9i 数据库

1. 企业版（Enterprise Edition）
2. 标准版（Standard Edition）
3. 个人版（Personal Edition）

1.3.2 Oracle 9i 应用服务器

Oracle 9i 应用服务器有两种版本。

1. 企业版（Enterprise Edition）

企业版主要用于构建互联网应用，面向企业级应用。

2. 标准版（Standard Edition）

标准版用于建立面向部门级的 Web 应用。

1.3.3 Oracle 9i 开发工具套件

Oracle 9i 开发工具套件是一整套的 Oracle 9i 应用程序开发工具。

1.4 Oracle 9i 的特点

Oracle 9i 在集群技术、高可用性、商业智能、安全性、系统管理等方面都实现了新的突破，其特点主要包括如下内容。

1.4.1 集群技术

集群的原理如图 1.2 所示。

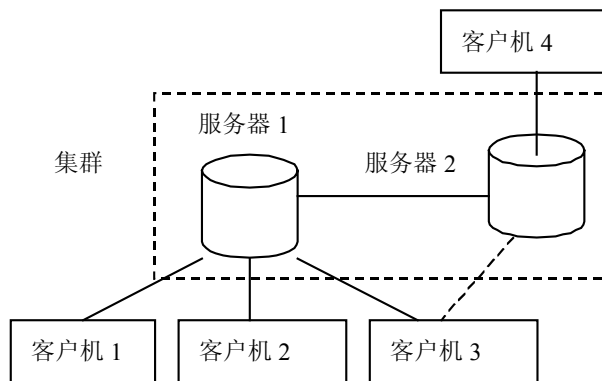


图 1.2 集群的原理

1.4.2 联机分析处理、数据挖掘和分析技术

1. 什么是联机分析处理

2. 什么是数据仓库

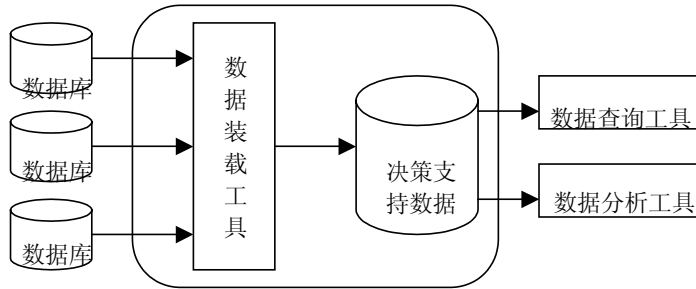


图 1.3 数据仓库的体系结构

3. 什么是数据挖掘和分析

数据分析是从大量的数据中获取所需要的决策数据的技术。数据挖掘是从数据仓库中由数据分析工具主动发现并提取隐藏在数据仓库内部的信息的新技术。

1.4.3 智能管理

1.4.4 分布式



图 1.4 分布式网络数据库

1.5 Oracle 9i 的两种工作模式

1.5.1 客户机/服务器模式

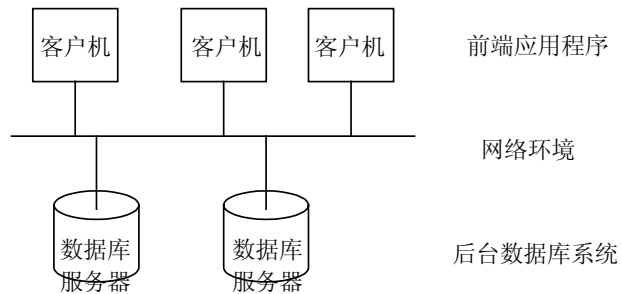


图 1.5 C/S 模式的网络数据库

1.5.2 浏览器/服务器模式

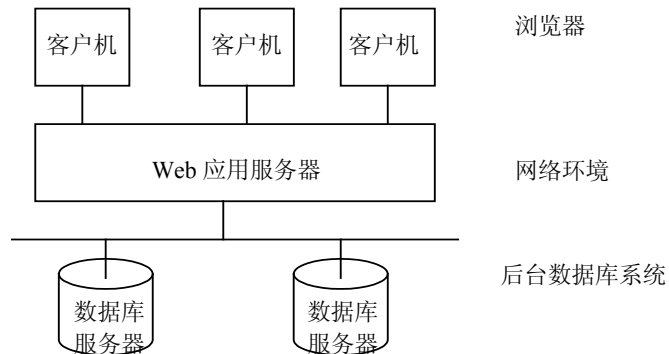


图 1.6 B/S 模式的网络数据库

1.6 习题

- (1) 关系型数据库是如何描述客观世界的信息的？
- (2) 关系型数据库系统和关系型数据库管理系统是什么关系？
- (3) 桌面和网络关系型数据库系统各有什么特点？
- (4) Oracle 9i 包含那些组成部分？各有什么用途？
- (5) 什么是主码和外码？
- (6) 什么是群集技术？
- (7) 什么是分布式网络数据库？
- (8) 什么是联机分析处理？
- (9) Oracle 9i 的两种工作模式的原理是什么？
- (10) Oracle 9i 数据库服务器和客户机有什么区别？
- (11) 端口有什么含义？

第2章 构建环境——安装 Oracle 9i

本章介绍 Oracle 9i 网络中两种主要的成分——数据库服务器和管理客户机的安装和配置过程。本章目的在于帮助读者掌握如何安装 Oracle 9i 以及如何配置 Oracle 9i 网络环境。

2.1 安装数据库服务器

本书采用 Windows 2000 Server 作为安装的网络操作系统平台，数据库服务器采用 Oracle 9i Database for Windows 2000 的企业版。

2.1.1 安装的硬件环境需求

下列从 5 个主要的方面阐述 Oracle 9i 对硬件环境的要求。

1. 对 CPU 的要求

CPU 最低配置到 Pentium 166 就可以。

2. 对内存的要求

内存容量最低为 64MB，最好在 256MB 以上。

3. 对硬盘的要求

建议配置 8GB 容量以上硬盘。

4. 对光驱的要求

建议选用快速光驱，16 倍速以上。

5. 对网卡的要求

一般可以选用 10/100MB 自适应网卡。

2.1.2 安装的软件环境需求

下列从两个主要的方面阐述 Oracle 9i 对软件环境的要求。

1. 对操作系统的要求

建议在全新安装的 Windows 2000 Server 上安装数据库服务器，在 Windows 2000 Server 或 Windows 98 上安装管理客户机。

2. 对虚拟内存的要求

建议可以将虚拟内存适当进行调整以加快安装速度。

2.1.3 安装的网络环境需求

安装 Oracle 9i 数据库服务器，至少需要有两台计算机，通过交换机或集线器构成局域网。

2.1.4 安装环境实例

安装环境实例如图 2.1 所示。

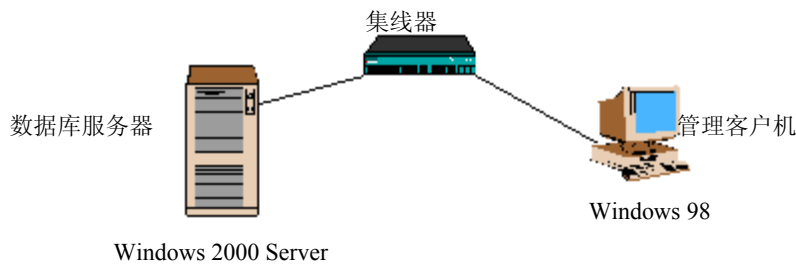


图 2.1 安装环境实例

- 1. 数据库服务器
- 2. 管理客户机

表 2.1 安装环境里计算机的配置情况

配置项	服务器	客户机
CPU	Pentium II 400	赛扬 466
内存	320MB	64MB
硬盘	10GB	8GB
光驱	16 倍速 CD-ROM	16 倍速 CD-ROM
网卡	Intel Pro + 10/100Mbit/s 自适应网卡	Intel Pro + 10/100Mbit/s 自适应网卡

2.1.5 用【Ping】命令测试网络是否连通

- (1) 输入“ping 192.168.100.5 -t”命令行，单击 按钮。
- (2) 出现如图 2.3 所示界面。则表明网络已经连通。

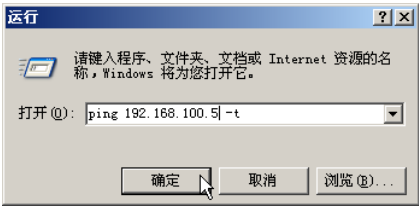


图 2.2 【运行】界面

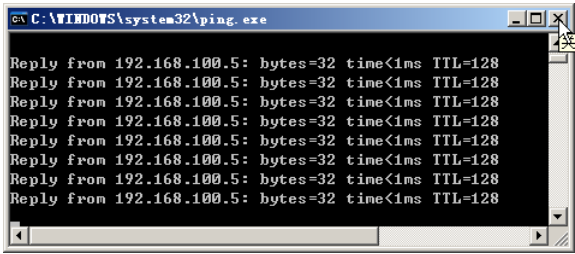


图 2.3 【网络连通】界面

2.1.6 安装步骤

- (1) 出现如图 2.4 所示的【安装】界面。单击【开始安装】按钮。
- (2) 出现如图 2.5 所示的【欢迎】界面，单击 按钮。

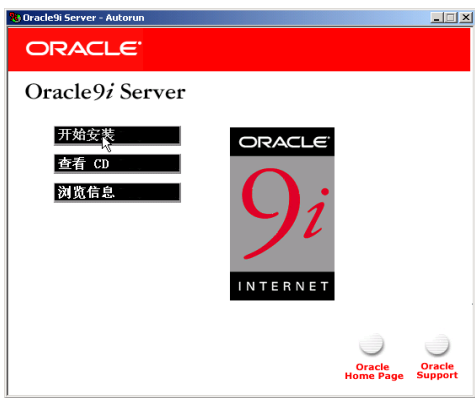


图 2.4 【安装】界面

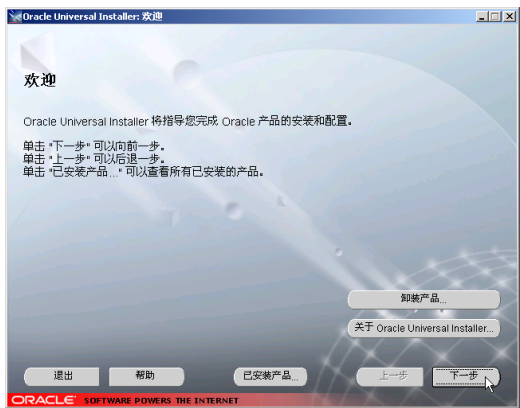


图 2.5 【欢迎】界面

(3) 出现如图 2.6 所示的【文件定位】界面。

(4) 出现如图 2.7 所示的【可用产品】界面。



图 2.6 【文件定位】界面



图 2.7 【可用产品】界面

表 2.2 Oracle 9i 的安装选项

安装选项	安装类型
Oracle 9i Database 9.0.1.1.1	企业版
	标准版
	个人版
	自定义（定制安装）
Oracle 9i Client 9.0.1.1.1	Administrator（安装管理控制台、管理工具、网络服务、实用程序和基本客户软件）
	Runtime（运行时，安装应用开发程序、网络服务和基本客户软件）
	自定义（定制安装）
Oracle 9i Management and Integration 9.0.1.0.1	Oracle Management Server（安装管理服务器）
	Oracle Internet Directory（安装 Oracle Internet Directory、客户机工具集、Oracle Directory Manager 和客户端开发工具包）

	Oracle Integration Server（安装配置高级队列、Oracle Java 虚拟机和工作流的数据库）
	自定义（定制安装）

(5) 出现如图 2.8 所示的【安装类型】界面。

(6) 出现如图 2.9 所示的【数据库配置】界面。

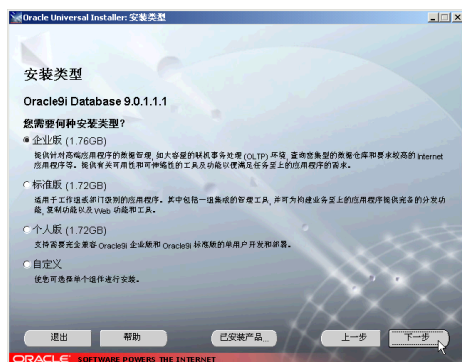


图 2.8 【安装类型】界面



图 2.9 【数据库配置】界面


(7) 出现如图 2.10 所示的【数据库标识】界面。在【全局数据库名】文本框里输入名称 “myoracle.mynet” 后，在【SID】文本框里自动生成 “myoracle”，单击  按钮。



图 2.10 【数据库标识】界面

(8) 出现如图 2.11 所示的【数据库文件位置】界面。

(9) 出现如图 2.12 所示的数据库字符集界面。



图 2.11 【数据库文件位置】界面

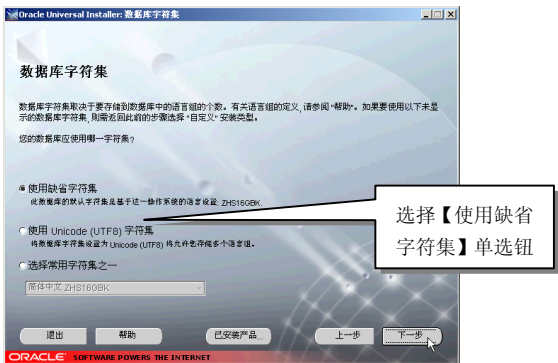


图 2.12 【数据库字符集】界面

(10) 出现如图 2.13 所示的【摘要】界面。

(11) 安装过程开始复制文件，在进行到整个过程的 44%时，出现如图 2.14 所示的【磁盘位置】界面。



图 2.13 【摘要】界面

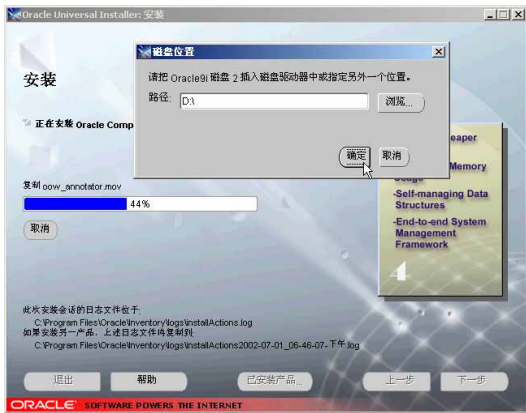


图 2.14 【磁盘位置】界面

(12) 在安装进程进行到 87%时，出现磁盘位置界面。

(13) 出现如图 2.15 所示的【配置工具】界面，安装程序将自动完成 4 项任务。

(14) 调用的【数据库配置助手】界面如图 2.16 所示。



图 2.15 【配置工具】界面

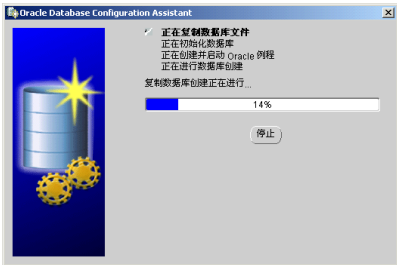


图 2.16 【数据库配置助手】界面

- (15) 【Oracle 数据库配置助手】配置成功后出现如图 2.17 所示的界面。
- (16) 出现如图 2.18 所示的【安装结束】界面。

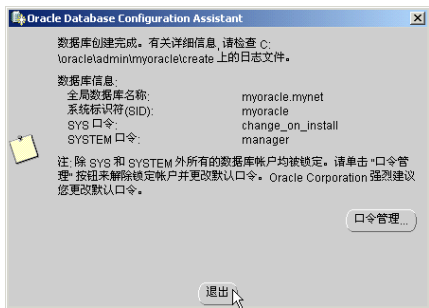


图 2.17 【数据库配置助手】配置成功界面



图 2.18 【安装结束】界面

2.1.7 安装结果

逻辑上来看实际上包括 3 个部分。

- ❑ 一个管理客户机。
- ❑ 一个数据库服务器。
- ❑ 一个数据库: 【全局数据库名】为 myoracle.mynet。

2.2 安装管理客户机

2.2.1 安装步骤

- (1) 出现如图 2.19 所示的【可用产品】界面。



图 2.19 【可用产品】界面

- (2) 出现如图 2.20 所示的【安装类型】界面。

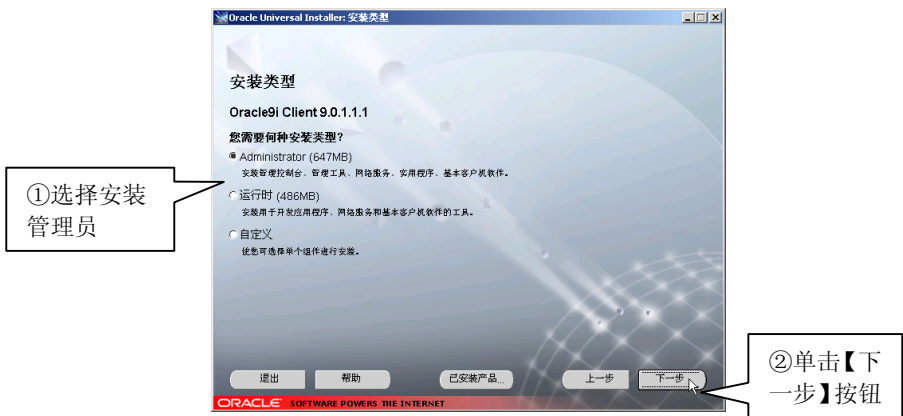


图 2.20 【安装类型】界面

- (3) 出现同服务器安装类似的【摘要】界面。
- (4) 在安装进程进行到 99%和 100%时出现同服务器安装类似的【磁盘位置】界面。
- (5) 出现如图 2.21 所示的 Oracle Net Configuration Assistant (Oracle 网络配置助手) 的【欢迎使用】界面。



图 2.21 网络配置助手【欢迎使用】界面

- (6) 出现如图 2.22 所示的【数据库版本】界面。

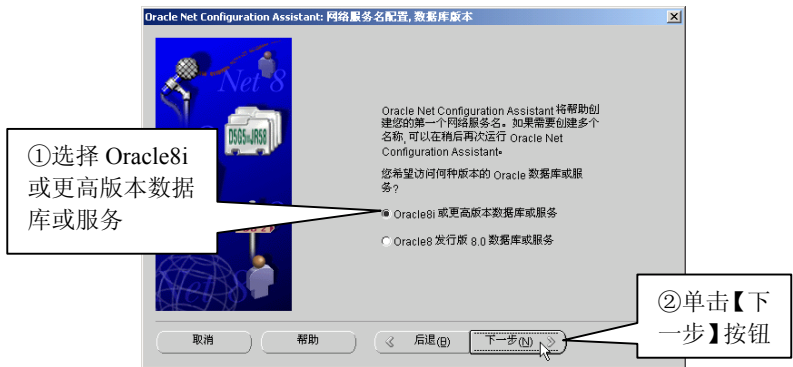


图 2.22 【数据库版本】界面

- (7) 出现如图 2.23 所示的【服务名】界面。

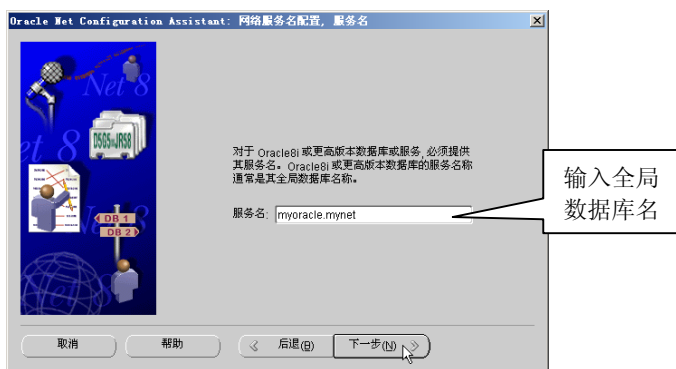


图 2.23 【服务名】界面

(8) 出现如图 2.24 所示的【请选择协议】界面。



图 2.24 【请选择协议】界面

(9) 出现如图 2.25 所示的【TCP/IP 协议】界面。

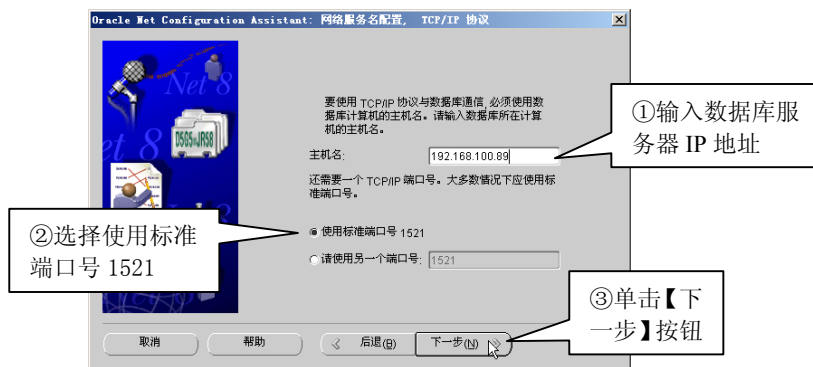


图 2.25 【TCP/IP 协议】界面

(10) 出现如图 2.26 所示的【测试】界面。

(11) 出现如图 2.27 所示的【正在连接】界面。



图 2.26 【测试】界面

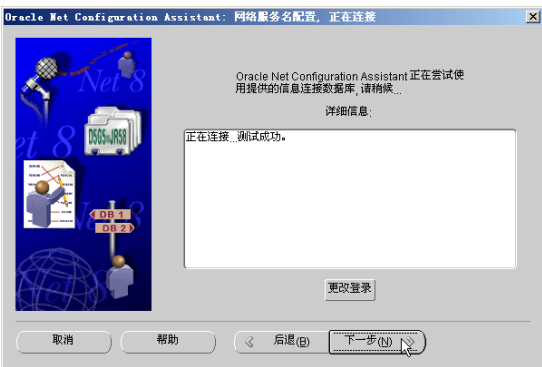


图 2.27 【正在连接】界面

- (12) 出现如图 2.28 所示的【网络服务名】界面。
- (13) 出现如图 2.29 所示的【是否配置另一个网络服务名】界面。



图 2.28 【网络服务名】界面



图 2.29 【是否配置另一个网络服务名】界面

- (14) 出现如图 2.30 所示的【网络服务名配置完毕】界面。
- (15) 出现如图 2.31 所示的【安装结束】界面。



图 2.30 【网络服务名配置完毕】界面



图 2.31 【安装结束】界面

2.2.2 安装结果

客户机的程序组，如图 2.32 所示。

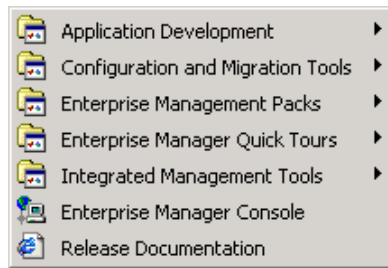


图 2.32 Oracle 9i 管理客户机的程序组

表 2.3 Oracle 9i 管理客户机的程序组

程序组名称	程序组含义
Application Development	【应用开发】程序组
Configuration and Migration Tools	【配置和迁移工具】程序组
Enterprise Management Packs	【企业管理包】程序组
Enterprise Manager Quick Tours	【企业管理者快速巡游】程序组
Integrated Management Tools	【集成管理工具】程序组
Enterprise Manager Console	【企业管理控制台】程序组
Release Documentation	【发行文档】程序组

2.3 服务器和客户机是怎样连接的

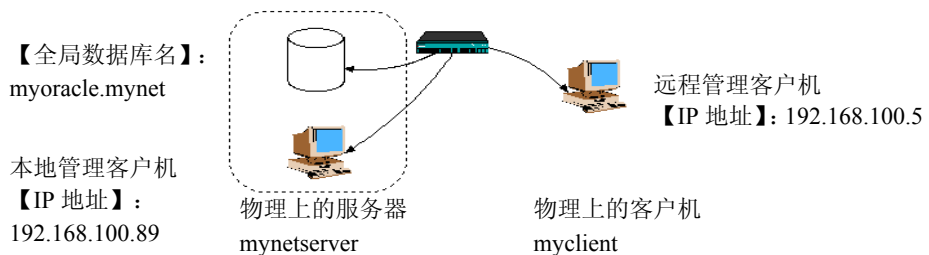


图 2.33 Oracle 9i 网络环境

2.3.1 服务器如何连接客户机

- (1) 出现如图 2.34 所示的【欢迎使用】界面。
- (2) 出现如图 2.35 所示的【监听程序配置，监听程序】界面。



图 2.34 【欢迎使用】界面

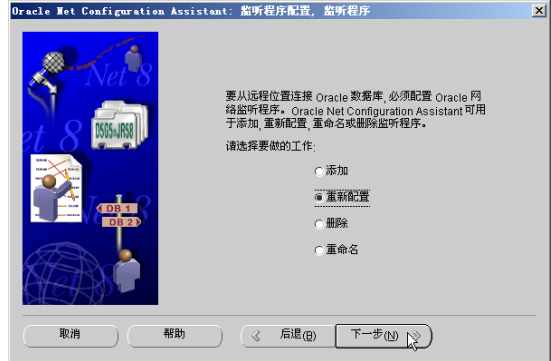


图 2.35 【监听程序配置，监听程序】界面

(3) 出现如图 2.36 所示的【监听程序配置，选择监听程序】界面。

(4) 出现如图 2.37 所示的【监听程序配置，选择协议】界面。



图 2.36 【监听程序配置，选择监听程序】界面



图 2.37 【监听程序配置，选择协议】界面

(5) 出现如图 2.38 所示的【监听程序配置，TCP/IP 协议】界面。

(6) 出现如图 2.39 所示的【监听程序配置，更多的监听程序】界面。

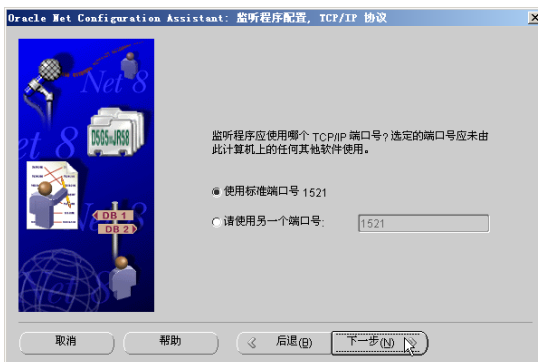


图 2.38 【监听程序配置，TCP/IP 协议】界面



图 2.39 【监听程序配置，更多的监听程序】界面

2.3.2 客户机怎样连接服务器

管理客户机的工作原理如图 2.40 所示。

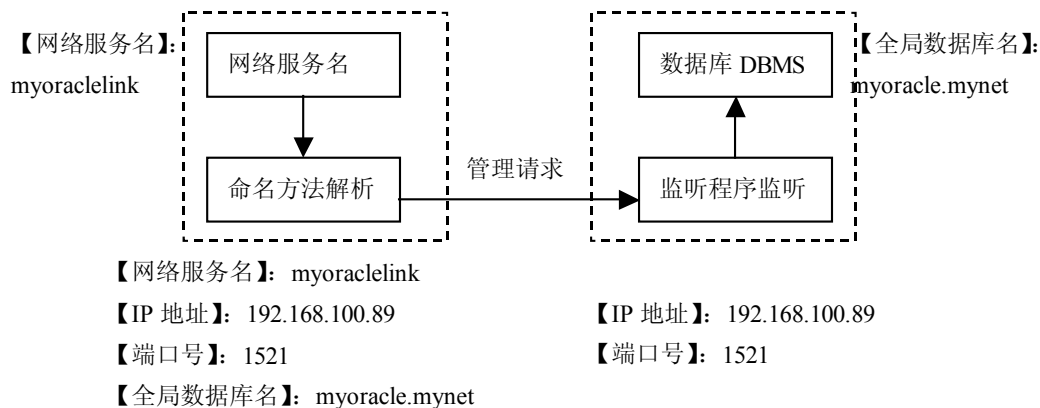


图 2.40 管理客户机的工作原理

1. 【命名方法】的配置

- (1) 出现如图 2.41 所示的【欢迎使用】界面。
- (2) 出现如图 2.42 所示的【命名方法配置，请选择命名方法】界面。



图 2.41 【欢迎使用】界面

图 2.42 【命名方法配置，请选择命名方法】界面

Oracle 9i 支持如表 2.4 所示的 5 种命名方法。

表 2.4 Oracle 9i 的 5 种命名方法

名称	主要特点
本地命名	适合具有少量不经常更改服务的简单分布式网络
目录命名	通过目录服务器进行解析，适合大型网络
Oracle 名称(Oracle Names)	通过 Oracle 名字服务器进行解析，适合大型网络
主机命名	通过【计算机名称】进行解析，适合局域网
外部命名	通过非 Oracle 命名服务进行解析，很少用

- (3) 出现如图 2.43 所示的【命名方法配置，主机名】界面。。
- (4) 出现如图 2.44 所示的【命名方法配置完成】界面。



图 2.43 【命名方法配置，主机名】界面



图 2.44 【命名方法配置完成】界面

2. 【本地网络服务名】的配置

- (1) 出现如图 2.45 所示的【欢迎使用】界面。
- (2) 出现如图 2.46 所示的【网络服务名配置】界面。



图 2.45 【欢迎使用】界面



图 2.46 【网络服务名配置】界面

2.4 数据库服务器的体系结构

2.4.1 进程结构

Oracle 9i 网络环境里共有两大类进程。

1. 用户进程

用户进程是在客户机内存上运行的程序，如客户机上运行的【SQL Plus】、【企业管理器】等。用户进程向服务器进程提出操作请求。

2. 服务器进程

主要的服务器进程如表 2.5 所示。

表 2.5

Oracle 9i 的主要后台支持进程

名称	主要作用
系统监控进程 (SMON)	数据库系统启动时执行恢复性工作, 对有故障数据库进行恢复
进程监控进程 (PMON)	用于恢复失败的用户进程
数据库写入进程 (DBWR)	将修改后的数据块内容写回数据库
日志写入进程 (LGWR)	将内存中的日志内容写入日志文件
归档进程 (ARCH)	当数据库服务器以归档方式运行时调用该进程完成日志归档
检查点进程 (CKPT)	标识检查点, 用于减少数据库恢复所需要的时间
恢复进程 (RECO)	用于分布式数据库中的失败处理
锁进程 (LCKn)	在并行服务器模式下确保数据的一致性
快照进程 (SNPn)	进行快照刷新
调度进程 (Dnnn)	负责把用户进程路由到可用的服务器进程进行处理

2.4.2 内存结构

1. 系统全局区 (SGA)

SGA 如图 2.47 所示。

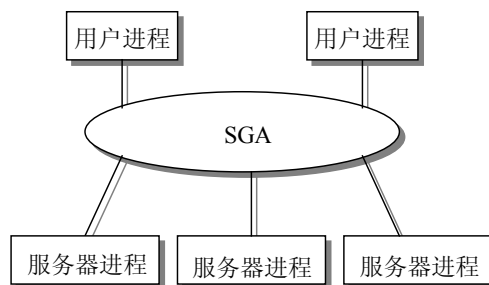


图 2.47 SGA 的作用

2. 程序全局区 (PGA)

PGA 是数据库服务器内存中为单个用户进程分配的专用的内存区域, 是用户进程私有的, 不能共享。

2.4.3 数据库的逻辑结构

Oracle 9i 数据库的逻辑结构主要指从数据库使用者的角度来考查的数据库的组成, 如图 2.48 所示。自下向上, 数据库的逻辑结构共有 6 层。

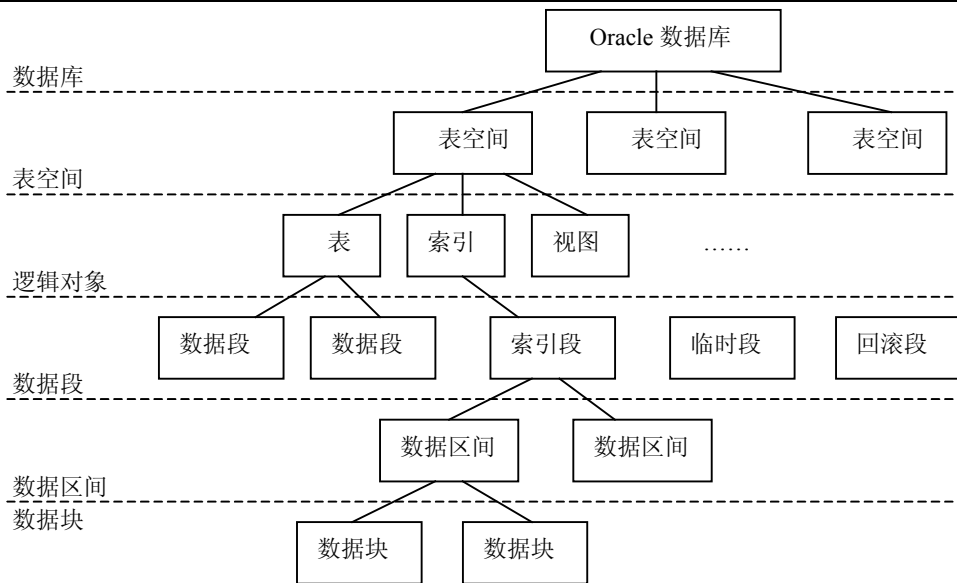


图 2.48 数据库的逻辑结构

1. 数据块 (Data Block)
2. 数据区间 (Data Extent)
3. 数据段 (Data Segment)
4. 逻辑对象 (Logic Object)
5. 表空间 (Tablespace)

Oracle 9i 安装完毕后自动建立 9 个默认表空间，如表 2.6 所示。

表 2.6 Oracle 9i 数据库的默认表空间

名称	主要作用
CWMLITE	用于联机分析处理 (OLAP)
DRSYS	用于存放与工作空间设置有关的信息
EXAMPLE	实例表空间，存放实例信息
INDEX	索引表空间，存放数据库索引信息
SYSTEM	系统表空间，存放表空间名称、所含数据文件等管理信息
TEMP	临时表空间，存储临时表
TOOLS	工具表空间，存放数据库工具软件所需的数据库对象
UNDOTBS	回滚表空间，存放数据库恢复信息
USERS	用户表空间，存放用户私有信息

6. 数据库（Database）

2.4.4 数据库的存储结构

数据库的存储结构指逻辑结构在物理上是如何实现的，共有 3 层，如图 2.49 所示。

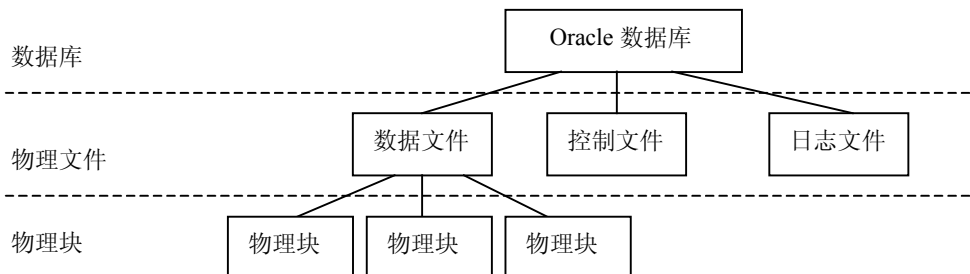


图 2.49 数据库的存储结构

1. 物理块

2. 物理文件

每个物理文件由若干个物理块组成，主要包括数据文件、控制文件和日志文件 3 类。

- ❑ 数据文件：用于存放所有的数据，以 DBF 为扩展名。
- ❑ 日志文件：记录了对数据库进行的所有操作，以 LOG 为扩展名。
- ❑ 控制文件：记录了数据库所有文件的控制信息，以 CTL 为扩展名。

以笔者的安装环境为例，【全局数据库名】为“myoracle.mynet”的数据库的主要物理文件存放在 c:\oracle\oradata\myoracle 下，如图 2.50 所示。

名称	大小	类型	创建日期	属性
CWMILT01.DBF	20,484 KB	DBF 文件	2003-09-06 18:22	A
DRSYS01.DBF	20,484 KB	DBF 文件	2003-09-06 18:22	A
EXAMPL01.DBF	156,164 KB	DBF 文件	2003-09-06 18:22	A
INDEX01.DBF	25,804 KB	DBF 文件	2003-09-06 18:22	A
SYSTEM01.DBF	332,804 KB	DBF 文件	2003-09-06 18:22	A
TEMP01.DBF	40,964 KB	DBF 文件	2003-09-03 17:51	A
TOOLS01.DBF	10,244 KB	DBF 文件	2003-09-06 18:22	A
UNDOTBS01.DBF	204,804 KB	DBF 文件	2003-09-06 18:22	A
USERS01.DBF	25,804 KB	DBF 文件	2003-09-06 18:22	A
CONTROL01.CTL	1,804 KB	Visual Basic Us...	2003-09-06 18:22	A
CONTROL02.CTL	1,804 KB	Visual Basic Us...	2003-09-06 18:22	A
CONTROL03.CTL	1,804 KB	Visual Basic Us...	2003-09-06 18:22	A
REDO01.LOG	102,401 KB	文本文档	2003-09-06 18:19	A
REDO02.LOG	102,401 KB	文本文档	2003-09-06 18:19	A
REDO03.LOG	102,401 KB	文本文档	2003-09-06 18:19	A

图 2.50 Oracle 9i 数据库的物理文件

2.4.5 数据库服务器的总体结构

Oracle 9i 数据库服务器的总体结构如图 2.51 所示。

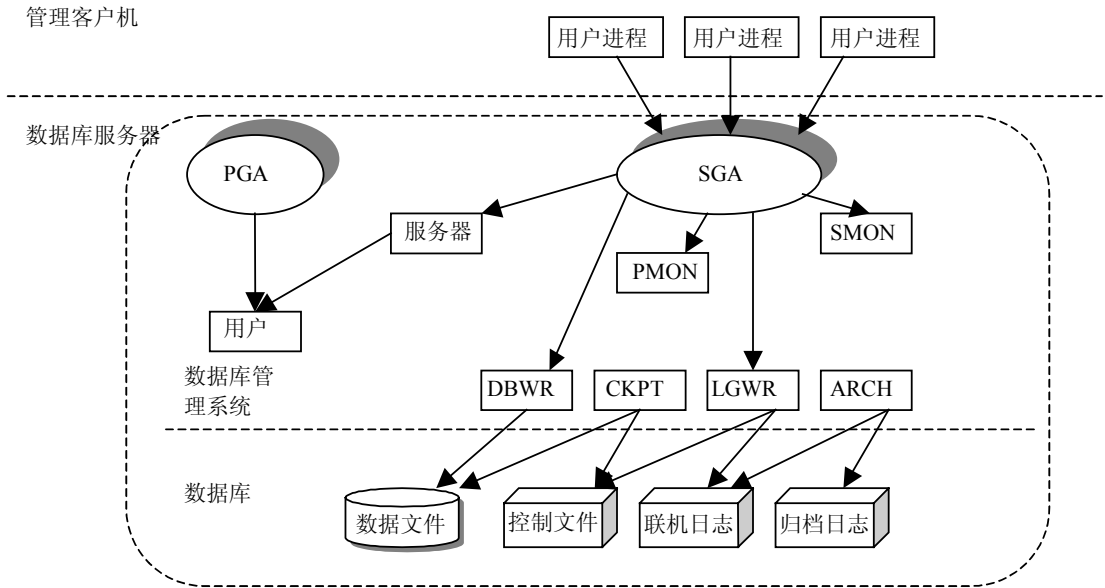


图 2.51 Oracle 9i 数据库服务器的总体结构

2.5 习题

- (1) 安装完数据库服务器后，还需要在同一台计算机上单独安装管理客户机吗？为什么？
- (2) 监听程序有什么作用？它是数据库服务器上的程序还是管理客户机上的程序？
- (3) 在管理客户机上为什么要进行【命名方法】的配置？
- (4) 【本地网络服务名】是如何配置的？
- (5) 试描述采用【本地网络服务名】时，数据库服务器和客户机的工作原理。
- (6) SGA 和 PGA 有什么主要区别？
- (7) 数据库的逻辑结构是什么？
- (8) 数据库主要的 3 种物理文件是什么，各有什么作用？

第 3 章 管理入门——数据库服务器的管理

本章介绍了 Oracle 9i 数据库服务器的基本管理操作。

3.1 管理的模式

3.1.1 直接管理模式

【企业管理器】直接管理模式的原理如图 3.1 所示，共有 3 层结构。

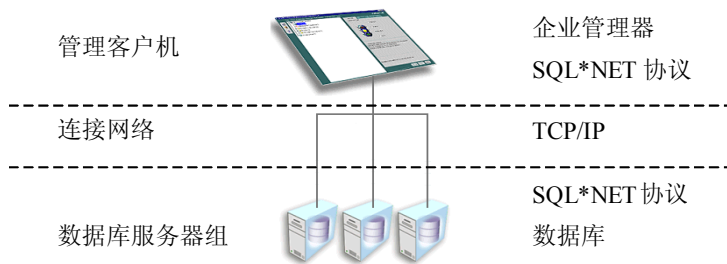


图 3.1 【企业管理器】直接管理模式

1. 管理客户机

在管理客户机上安装【企业管理器】，通过【企业管理器】调用其他的集成管理工具，利用 SQL *NET 协议建立与数据库服务器的连接。

2. 连接网络

常用 TCP/IP 协议构建连接网络。

3. 数据库服务器组

分布式的数据库服务器组，服务器运行 SQL *NET 协议响应客户机的连接，在每个服务器上可能有若干个数据库。

3.1.2 【管理服务器】集中管理模式

集中管理模式原理如图 3.2 所示，除了连接网络层外还有 3 层结构。

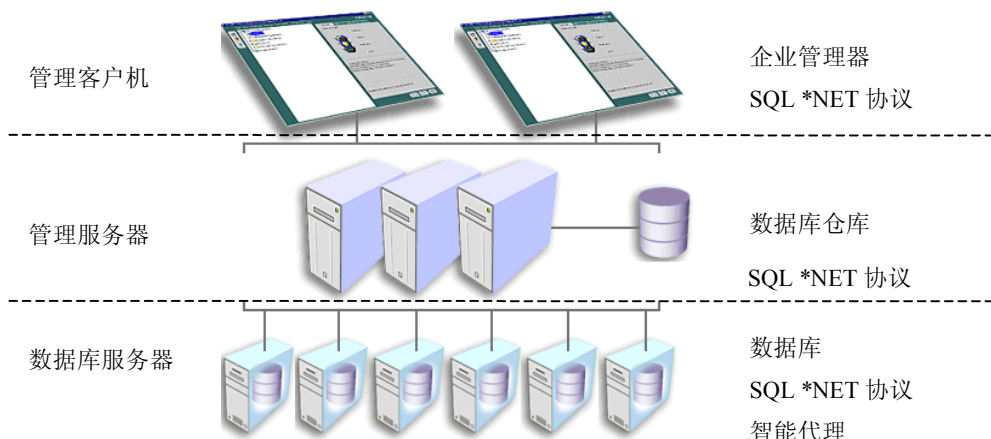


图 3.2 【管理服务器】的集中管理原理

1. 【管理客户机】

□ 在【管理客户机】上安装【企业管理器】，从【企业管理器】登录【管理服务器】，然后调用其他集成管理工具。

□ 【企业管理器】不直接连接数据库，而是通过【管理服务器】登录数据库。

2. 【管理服务器】

□ 【管理服务器】是连接数据库服务器和管理客户机的中间层。

□ 在【管理服务器】上建立了数据库仓库用于存储来自管理客户机的管理信息，然后将管理任务下达给数据库服务器的【智能代理】执行。

□ 要使用事件、作业、组、电子邮件、调度等高级管理功能必须建立管理服务器。

3. 【数据库服务器】

【数据库服务器】由管理目标（包括节点、数据库、Web 服务器、应用服务器等）和【智能代理】构成。通过【智能代理】完成【管理服务器】下发的管理任务。

【智能代理】是专门完成【管理服务器】下达的任务的程序，在选择【典型】安装 Oracle 9i 数据库服务器时已经自动安装上了【智能代理】。

3.2 配置【管理服务器】

【管理服务器】是 Oracle 9i 安装光盘中一个可以单独进行安装的组件。

3.2.1 配置环境

下面介绍在笔者的安装环境下如何配置【管理服务器】，配置的环境如下。

□ 【计算机名称】：mynetserver。

□ 【IP 地址】：192.168.100.89。

□ 【全局数据库名】：myoracle.mynet。

3.2.2 配置步骤

(1) 如图 3.3 所示的【欢迎使用】界面。



图 3.3 【欢迎使用】界面

(2) 出现如图 3.4 所示的【配置操作】界面。

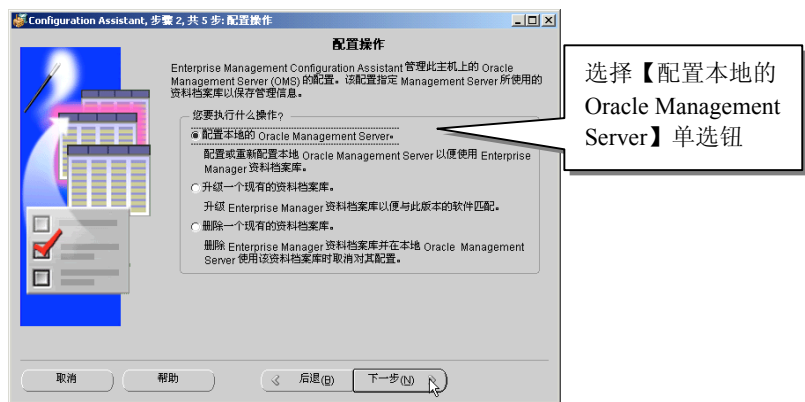


图 3.4 【配置操作】界面

(3) 出现如图 3.5 所示的【配置 Oracle Management Server】界面。



图 3.5 【配置 Oracle Management Server】界面

(4) 出现如图 3.6 所示的【创建新资料档案库选项】界面。



图 3.6 【创建新资料档案库选项】界面

(5) 出现如图 3.7 所示的【创建资料档案库概要】界面。



图 3.7 【创建新资料档案库概要】界面

(6) 系统将调用【数据库配置助手】创建数据库。

(7) 成功配置【管理服务器】后出现如图 3.9 所示界面，单击  按钮。

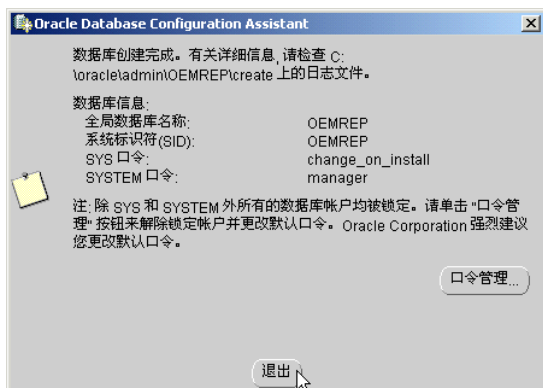


图 3.8 【成功创建 OEMREP 数据库】界面



图 3.9 【成功创建管理服务器】界面

3.2.3 【管理服务器】的启动和关闭

(1) 有两个后台服务与管理服务器有关，如图 3.10 所示。

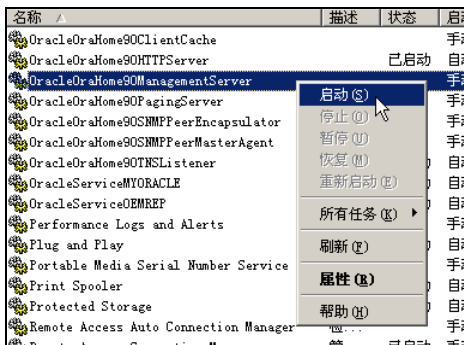


图 3.10 【管理服务器】的后台服务

(2) 若要启动【管理服务器】，在【OracleOraHome90ManagementSever】服务上用鼠标右键单击，在出现的快捷菜单中选择【启动】选项，系统将自动启动【管理服务器】。

(3) 在已启动的【OracleOraHome90ManagementSever】服务上用鼠标右键单击，在出现的快捷菜单中选择【停止】选项，系统将自动关闭【管理服务器】，如图 3.11 所示。

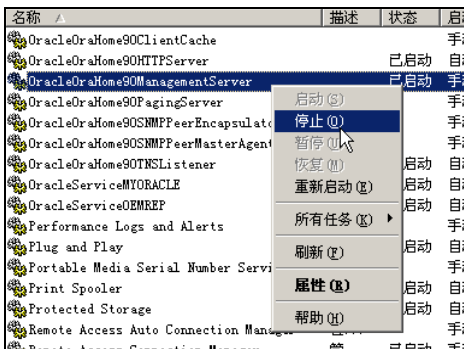


图 3.11 关闭【管理服务器】

3.3 管理的用户

3.3.1 【system】用户

1. 用户密码

【system】用户默认的密码为“manager”。

2. 用户权限

【system】用户具有“SYSDBA”权限，即数据库管理员权限，包括。

- ☐ 打开数据库服务器 关闭数据库服务器
- ☐ 备份数据库 恢复数据库

- ☐ 日志归档
- ☐ 管理功能
- 会话限制
- 创建数据库

3.3.2 【sys】用户

1. 用户密码

【sys】用户默认的密码为“change_on_install”。

2. 用户权限

【sys】用户具有“SYSDBA”或者“SYSOPER”权限。“SYSOPER”即数据库操作员权限，包括。

- ☐ 打开数据库服务器
- ☐ 备份数据库
- ☐ 日志归档
- 关闭数据库服务器
- 恢复数据库
- 会话限制

3.3.3 【scott】用户

1. 用户密码

【scott】用户默认的密码为“tiger”。

2. 用户权限

【scott】用户具有“Normal”，即普通用户权限，可以用来查询某些数据表的数据。

3.3.4 【sys】和【system】用户的比较

【sys】和【system】用户是安装时创建的两个系统管理员用户，但具体使用时是有区别的，如表 3.1 所示。

表 3.1 【sys】和【system】用户比较

比较内容		【sys】用户	【system】用户
默认密码		change_on_install	manager
直接管理模式	Normal	x	√
	SYSOPER	√	x
	SYSDBA	√	√
【管理服务器】 集中管理模式	Normal	x	√
	SYSOPER	√	x
	SYSDBA	√	x

3.4 管理工具的使用

3.4.1 【SQL Plus】的使用

(1) 如图 3.12 所示的【注册】界面。

(2) 成功连接数据库后出现如图 3.13 所示的【SQL Plus】界面。



图 3.12 【注册】界面



图 3.13 【SQL Plus】界面

(3) 输入查询语句“select * from scott.emp;”（查询 scott 用户下的 emp 数据表的所有记录），执行结果如图 3.14 所示。

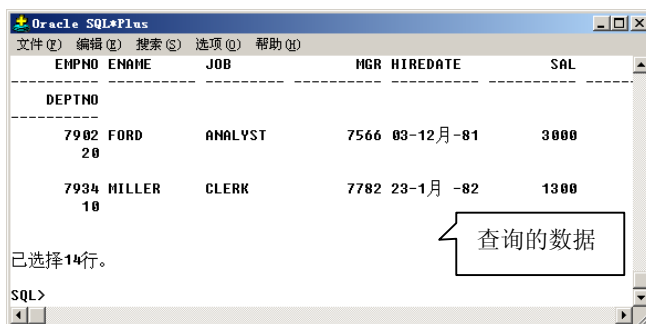


图 3.14 【SQL Plus】执行结果界面

(4) 执行完毕后，输入 quit 或 exit 将返回到 Windows 状态。

(5) 按照图 3.12 登录的【system】用户实际上只具有“Normal”权限。如果要使用户按照“sysdba”或者“sysoper”权限登录，必须显式指明登录身份。如图 3.15 所示。



图 3.15 指明登录身份的【注册】界面

3.4.2 【SQLPlus Worksheet】的使用

(1) 如图 3.16 所示的【企业管理器登录】界面。

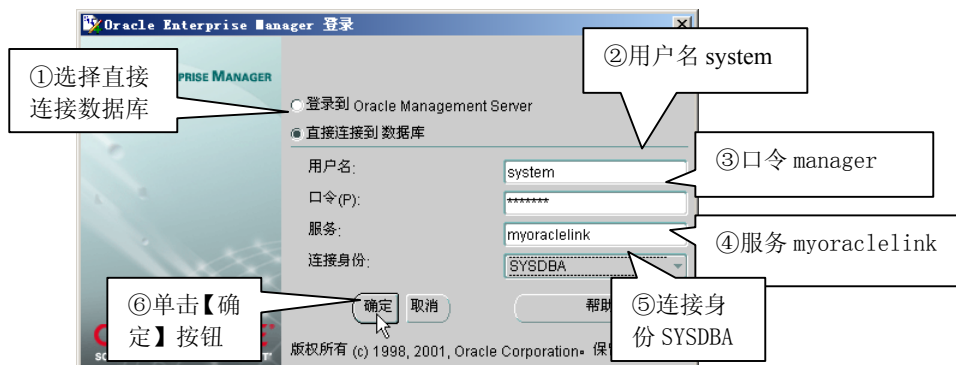


图 3.16 【企业管理器登录】界面

(2) 出现如图 3.17 所示的【SQLPlus Worksheet】界面，主要包括 4 大部分。

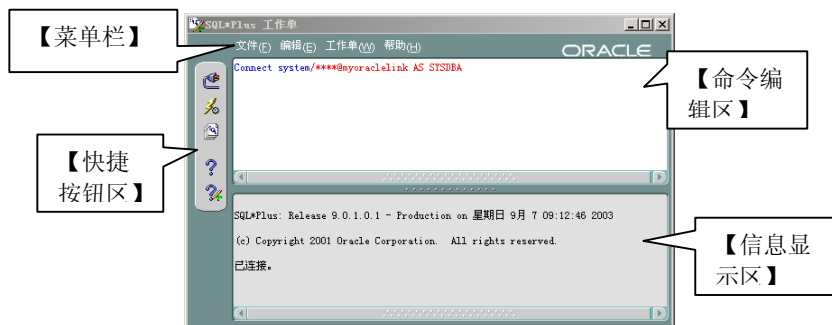


图 3.17 【SQLPlus Worksheet】界面

(3) 在【命令编辑区】输入“select * from acott.emp;”，然后在【快捷按钮区】单击【执行】按钮，结果如图 3.18 所示。

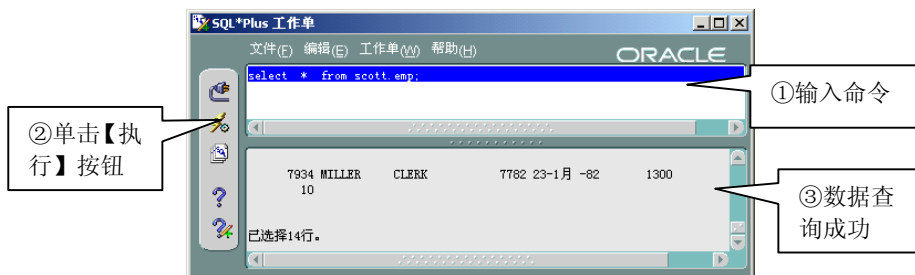


图 3.18 执行查询语句

(4) 在【菜单栏】选择【文件】/【将输入另存为】命令，如图 3.19 所示。

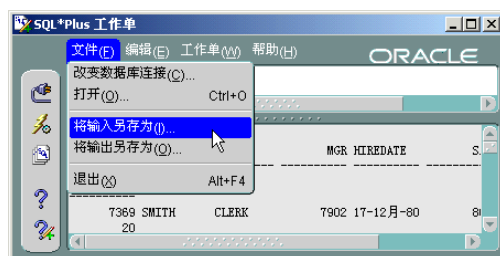


图 3.19 【选择存储 SQL 语句】界面

(5) 出现如图 3.20 所示的【将工作单另存为】界面。

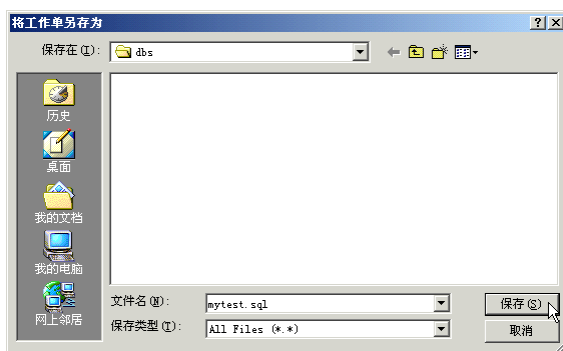


图 3.20 【将工作单另存为】界面

3.4.3 【企业管理器】的使用

1. 独立登录

(1) 出现如图 3.21 所示的【Oracle 企业管理器控制台】界面。

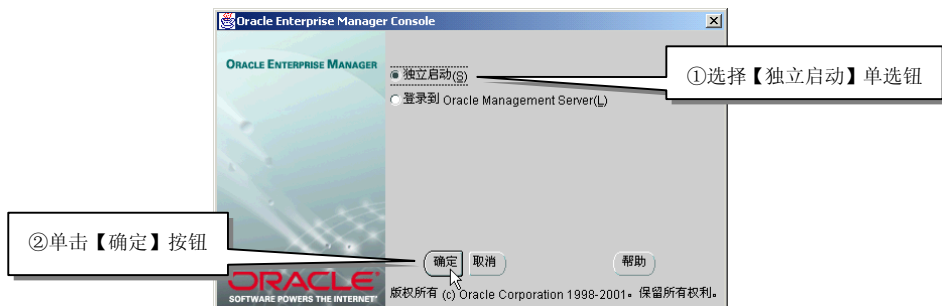


图 3.21 【Oracle 企业管理器控制台】界面

(2) 出现如图 3.22 所示的【独立启动后的企业管理器】界面。

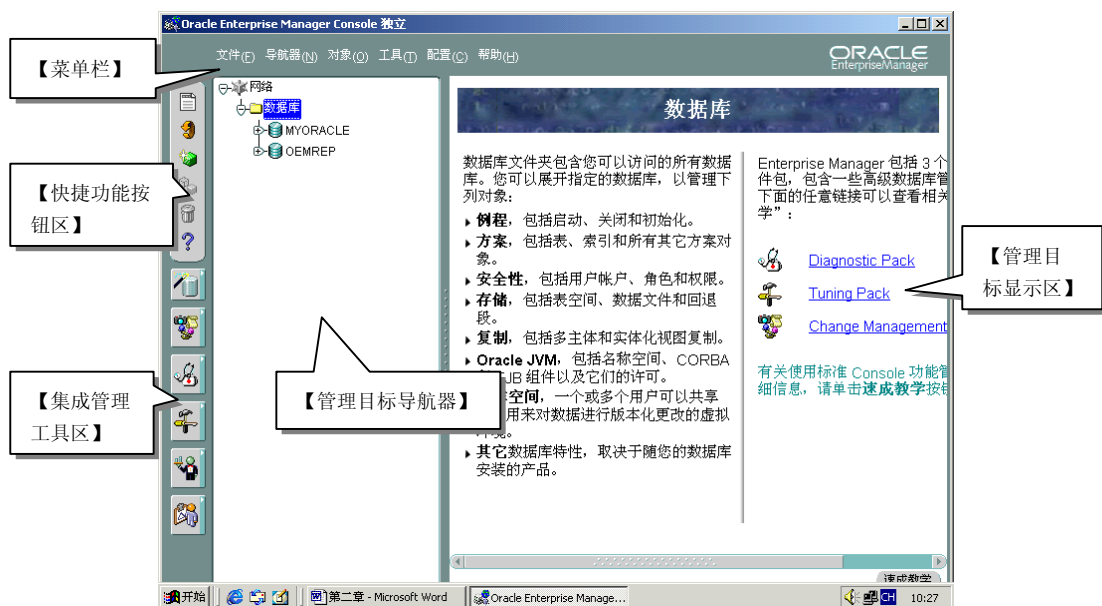


图 3.22 【独立启动后的企业管理器】界面

(2) 【菜单栏】有 6 项内容。

(3) 【快捷功能按钮区】共有 6 个快捷按钮，如图 3.23 所示。

(4) 【集成管理工具区】共有 6 大类集成管理工具，如图 3.24 所示。

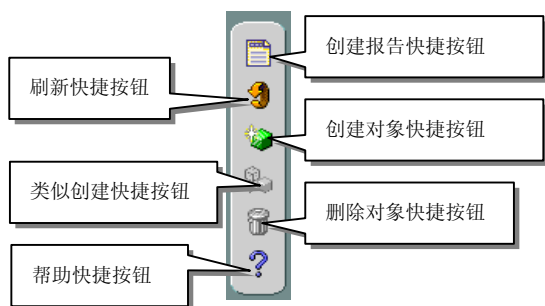


图 3.23 【快捷功能按钮区】

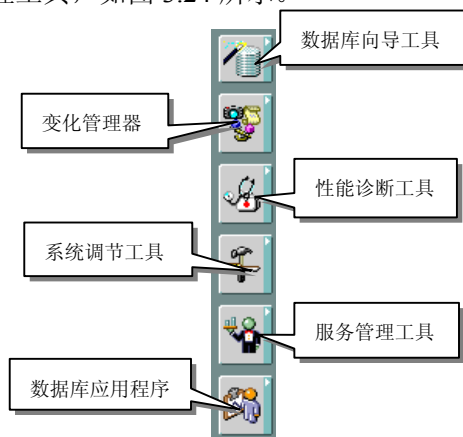


图 3.24 【集成管理工具区】

(5) 【管理目标导航器】集中管理数据库对象，必须先连接数据库后才能显示。【管理目标显示区】将显示该类别下的所有对象。

(6) 要对数据库实行管理首先必须登录数据库，在【企业管理器】里也叫连接。在【管理目标导航器】里双击【网络】/【数据库】/【MYORACLE】选项，出现如图 3.25 所示的【数据库连接信息】界面。

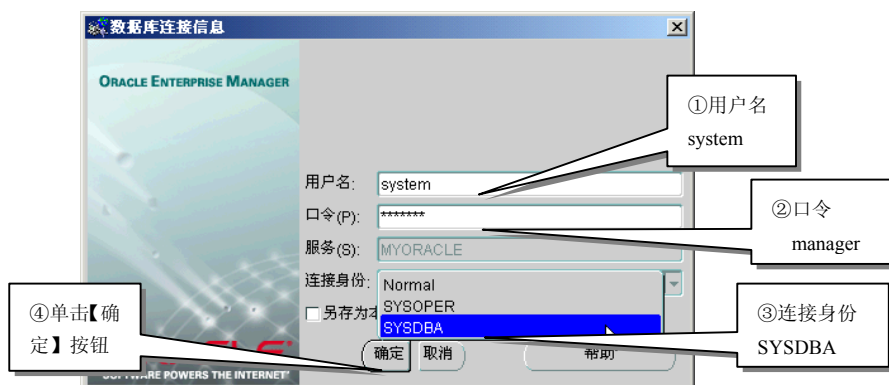


图 3.25 【数据库连接信息】界面

(7) 出现如图 3.26 所示的登录数据库后的【企业管理器】界面，就可以对数据库进行详细的管理了。

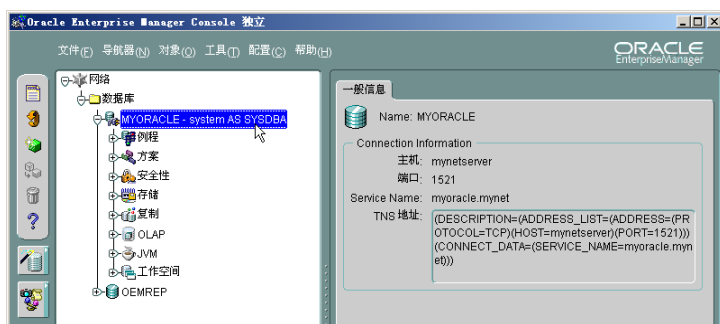


图 3.26 登录数据库后的【企业管理器】界面

2. 登录【管理服务器】

(1) 如图 3.27 所示的【Oracle 企业管理控制台】界面。

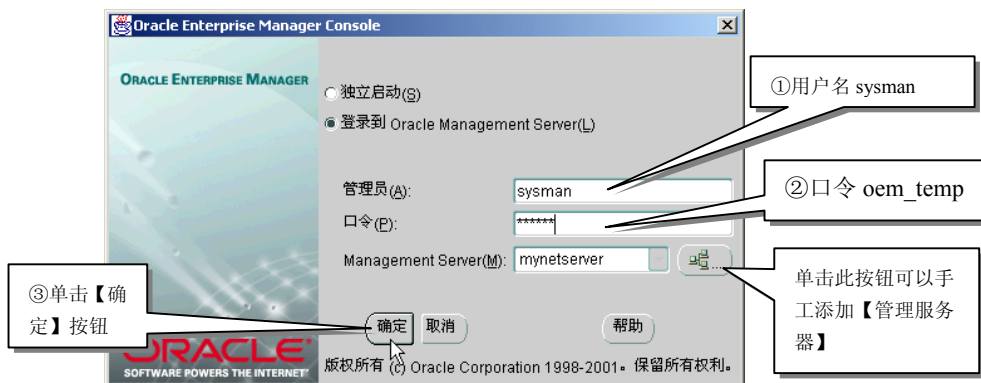


图 3.27 【Oracle 企业管理控制台】界面

(4) 出现如图 3.28 所示的【安全警告】界面。

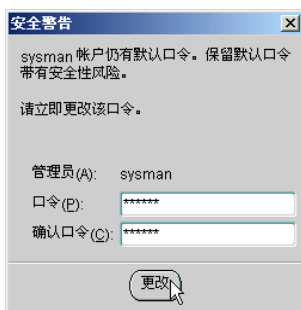


图 3.28 【安全警告】界面

(5) 成功登录管理服务器后的企业管理器如图 3.29 所示。

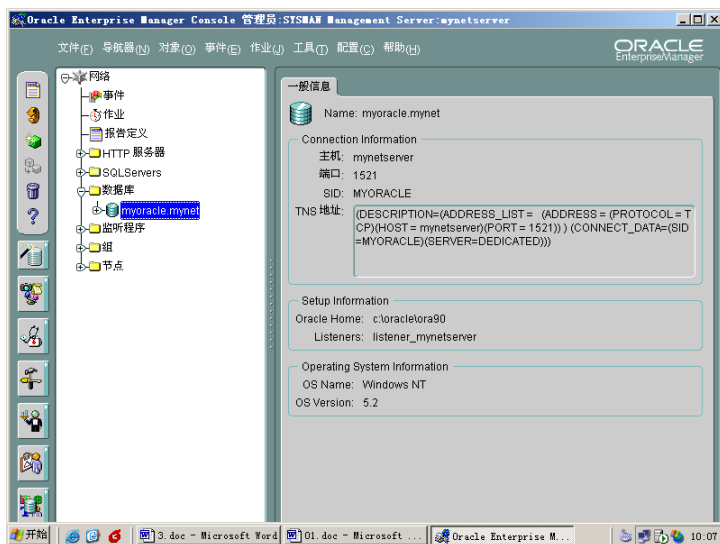


图 3.29 登录【管理服务器】后的【企业管理器】界面

3.5 服务器的关闭

3.5.1 服务器关闭的过程

关闭数据库例程时系统将完成以下步骤。

1. 第一阶段

- ❑ Oracle 将重执行缓冲区里的内容写入重执行日志文件。
- ❑ 将数据库缓冲区内被更改的数据写入数据文件。
- ❑ 关闭数据文件和重执行日志文件。
- ❑ 控制文件仍然打开，数据库不能进行一般性的访问操作。

2. 第二阶段

- ☐ 关闭数据库。
- ☐ 例程开始卸载数据库。
- ☐ 关闭控制文件，但 SGA 内存和后台支持进程仍然在运行。

3. 第三阶段

- ☐ 例程关闭。
- ☐ 释放 SGA 内存。
- ☐ 结束所有后台进程。

3.5.2 服务器关闭的 4 种方式

Oracle 9i 共有 4 种服务器关闭的方式，如表 3.2 所示。

表 3.2 服务器关闭的 4 种方式

方式	特点
正常关闭	数据库正常关闭。应用关闭选项后，不允许有新的连接。所有与数据库连接的用户必须在关闭数据库之前断开与该数据库的连接
立即关闭	数据库立即关闭。当前的客户机 SQL 语句立即终止
中止关闭	中止正在被 Oracle 数据库服务器处理的当前客户机 SQL 语句
事务处理关闭	提供一个指定的时间长度。关闭数据库前在该时间内完成事务处理。直到最后一个数据库事务处理完成后，才关闭数据库

3.5.3 正常关闭

(1) 如图 3.30 所示。

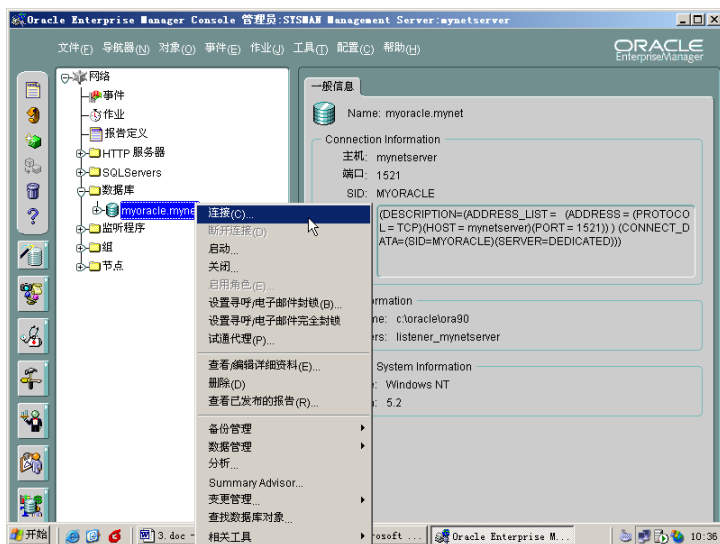


图 3.30 【更改数据库连接身份】界面

(2) 出现如图 3.31 所示的【数据库连接信息】界面。



图 3.31 【数据库连接信息】界面

(3) 出现如图 3.32 所示界面。



图 3.32 【选择关闭例程】界面

(4) 出现如图 3.33 所示的【关闭选项】界面。

(5) 出现如图 3.34 所示的【正常关闭例程的提示信息】界面。

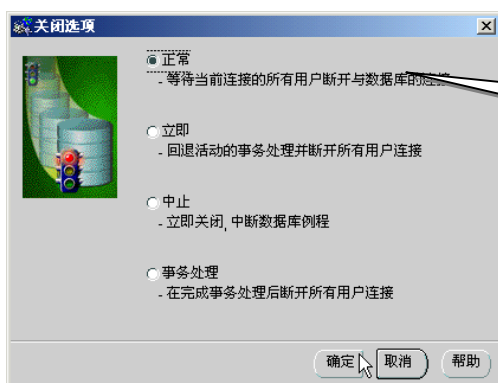


图 3.33 【关闭选项】界面

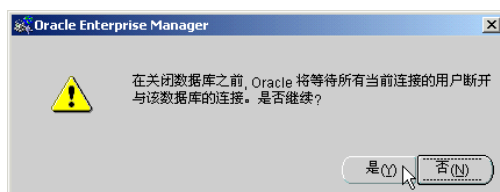


图 3.34 【正常关闭例程的提示信息】界面

(6) 成功关闭例程后出现如图 3.35 所示界面。

(7) 出现数据库配置的【一般信息】选项卡，如图 3.36 所示。



图 3.35 【成功关闭例程】界面

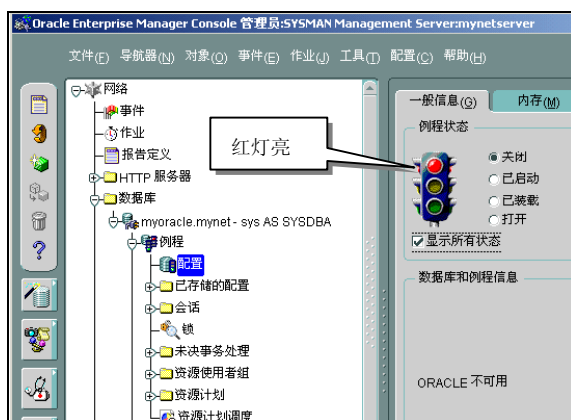


图 3.36 正常关闭例程后的状态

3.5.4 立即关闭

(1) 在图 3.37 所示的【关闭选项】界面里选择【立即】单选钮。

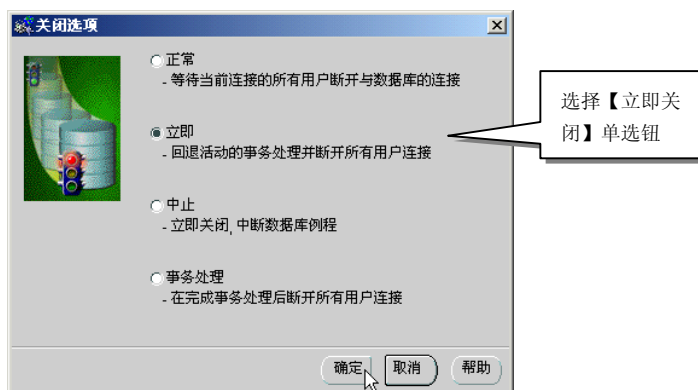


图 3.37 【关闭选项】界面

(2) 关闭完毕，出现如图 3.35 所示界面。

3.5.5 中止关闭

(1) 图 3.38 所示【关闭选项】界面。

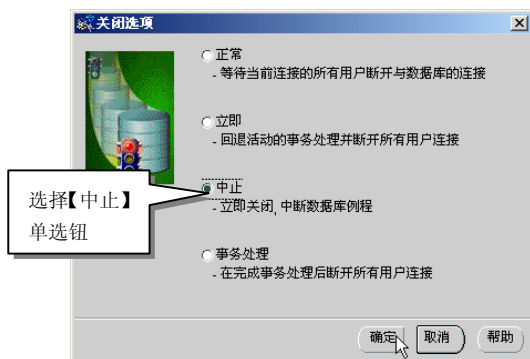


图 3.38 【关闭选项】界面

(2) 关闭完毕，出现如图 3.35 所示界面。

3.5.6 事务处理关闭

(1) 在图 3.39 所示的【关闭选项】界面里选择【事务处理】单选钮。

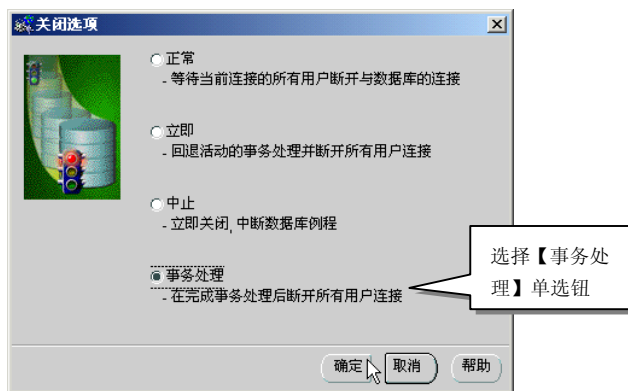


图 3.39 【关闭选项】界面

(2) 关闭完毕，出现如图 3.35 所示界面。

3.6 服务器的启动

3.6.1 服务器启动的 6 种方式

Oracle 9i 的数据库服务器有 6 种启动的方式，如表 3.3 所示。

表 3.3

服务器启动的 6 种方式

方式	特点
正常启动	启动装载和打开数据库，用户可以进行正常访问
不装载启动	在未装载数据库的情况下启动服务器，由于未打开数据库，所以不允许用户访问
装载启动	启动服务器并装载数据库，但不打开数据库。不允许用户访问
强制启动	强制启动未能正常关闭或正常启动的服务器
限制会话启动	只有具备“限制会话”（Restricted Session）系统权限的用户才能连接到数据库
只读启动	只读方式只允许用户查询打开的数据库，因此不可能进行联机数据库修改

3.6.2 正常启动

(1) 如图 3.40 所示。

(2) 出现如图 3.41 所示的【启动选项】界面。单击 **确定** 按钮。

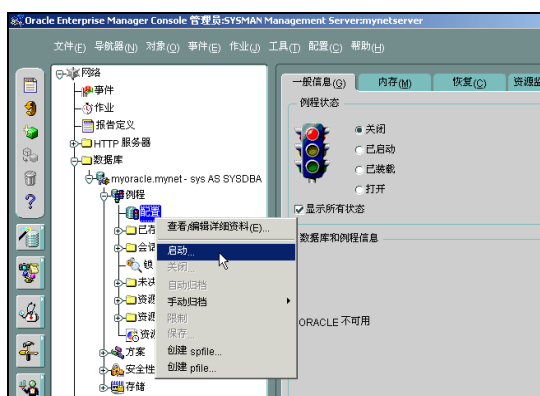


图 3.40 【启动数据库服务器】界面

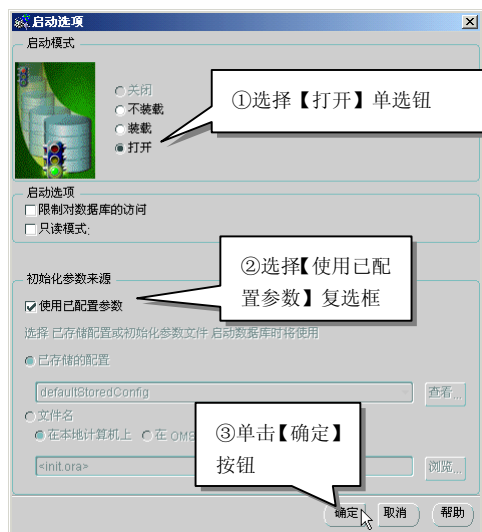


图 3.41 【启动选项】界面

(3) 服务器启动完毕，出现如图 3.42 所示的界面。

(4) 正常启动例程后，例程状态绿灯亮，表明数据库已经正常打开可以进行连接访问，如图 3.43 所示。



图 3.42 【正常启动服务器】界面

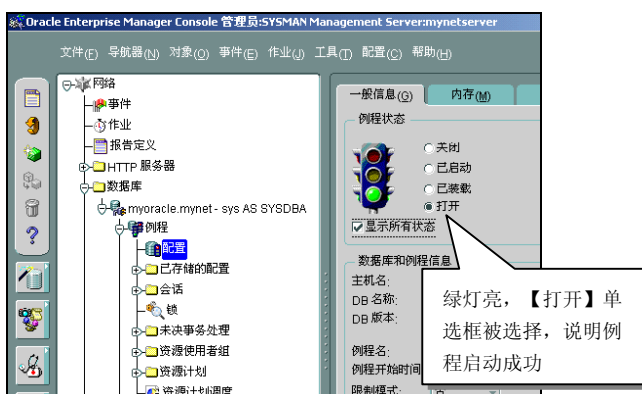


图 3.43 成功启动服务器的状态

3.6.3 不装载启动

(1) 如图 3.44 所示为【启动选项】界面。单击 **确定** 按钮。

(2) 出现如图 3.45 所示界面，单击 **关闭** 按钮。

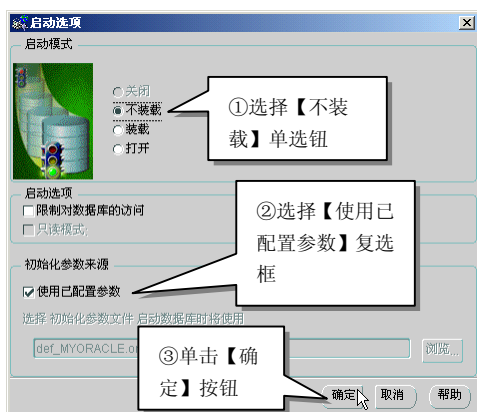


图 3.44 【启动选项】界面



图 3.45 【不装载启动服务器】界面

(3) 如图 3.46 所示。

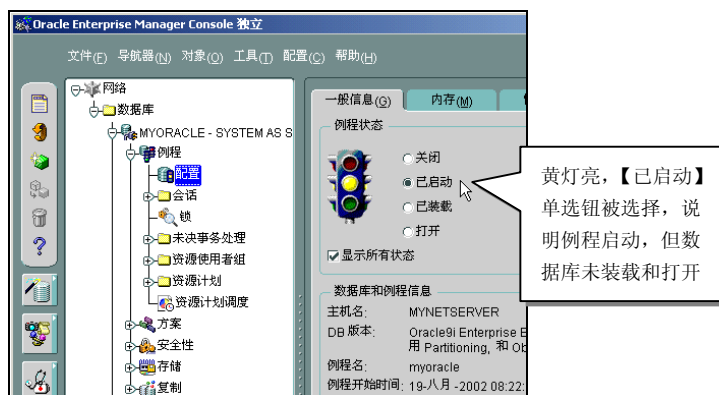


图 3.46 【不装载启动服务器】后的状态

3.6.4 装载启动

- (1) 如图 3.47 所示为【启动选项】界面。
- (2) 出现如图 3.48 所示界面。

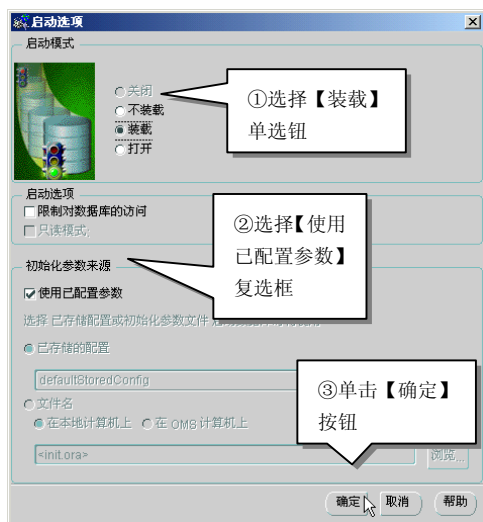


图 3.47 【启动选项】界面



图 3.48 【装载启动服务器】界面

- (3) 例程状态黄灯亮，【启动】单选按钮被选中，如图 3.49 所示。

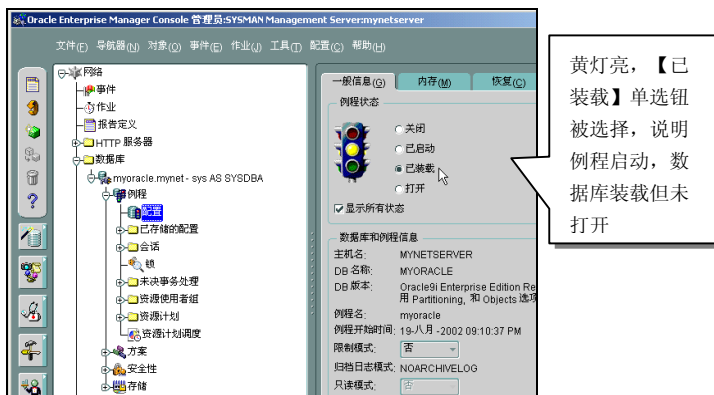


图 3.49 【装载启动服务器】后的状态

3.6.5 强制启动

- (1) 以“SYSDBA”身份登录【SQL Plus】。
- (2) 在【SQL Plus】中执行“startup force;”语句，出现如图 3.50 所示界面，说明例程已经被启动，数据库被装载和打开。

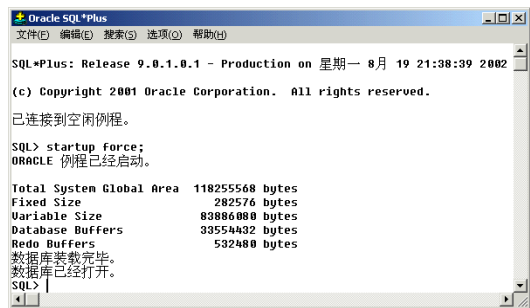


图 3.50 在【SQL Plus】中强制启动服务器

3.6.6 限制启动

(1) 如图 3.51 所示为【启动选项】界面。

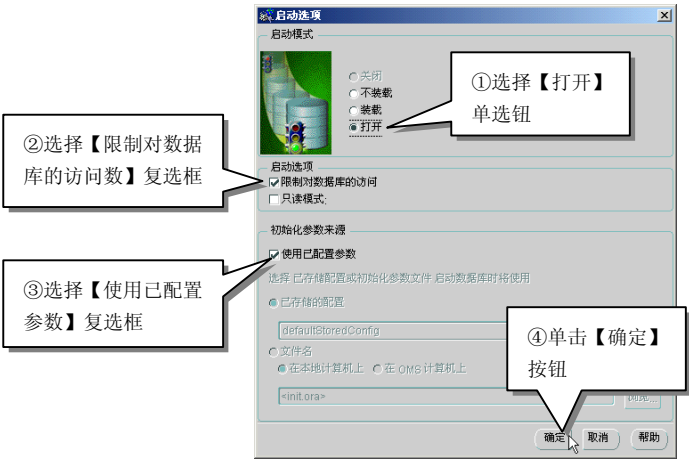


图 3.51 【启动选项】界面

(2) 出现如图 3.42 所示界面。

3.6.7 只读启动

(1) 如图 3.52 所示为【启动选项】界面。

(2) 出现如图 3.42 所示界面。

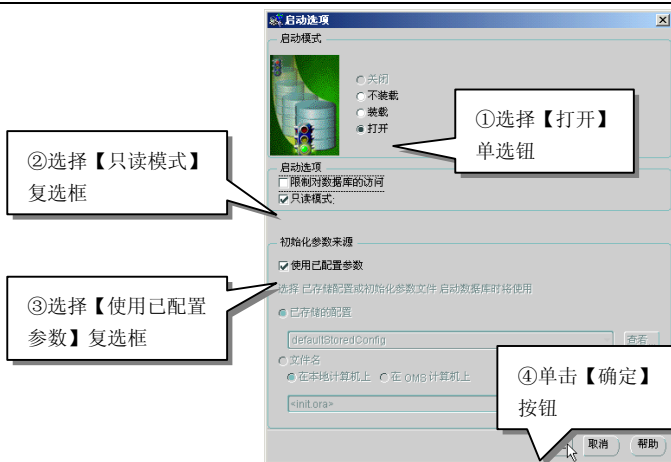


图 3.52 【启动选项】界面

3.7 习题

- (1) 试阐述直接管理模式的组成部分及原理。
- (2) 试阐述【管理服务器】管理模式的组成部分及原理。
- (3) 如何配置【管理服务器】？
- (4) 【管理服务器】的登录账号和 OEMREP 数据库的账号有什么区别？
- (5) 如何启动和关闭【管理服务器】？
- (6) 【sys】和【system】用户在使用上有什么需要注意的地方？
- (7) 4 种数据库服务器的关闭方式各有什么特点？
- (8) 6 种数据库服务器的启动方式各有什么特点？

第4章 数据利器——SQL

本章通过实例的方法引导读者快速掌握 SQL 的使用，从而能够利用【SQL Plus Worksheet】等工具，使用标准 SQL 语言完成对数据库数据的日常管理工作。

4.1 节介绍 SQL 的概念、主要特点、使用 SQL 的工具、SQL 如何访问数据表以及本书实例所使用的两个数据表。

4.2 节介绍对单个数据表进行的查询操作。

4.3 节介绍如何对多个数据表同时进行查询操作。

4.4 节介绍稍复杂的嵌套查询。

4.5 节介绍了最常用的查询函数。

4.6 节介绍如何使用 SQL 录入数据。

4.7 节介绍了如何使用 SQL 删除数据。

4.8 节介绍如何使用 SQL 更新数据。

4.1 SQL 概述

4.1.1 SQL 是什么

SQL (Structured Query Language, 译为结构化查询语言) 在关系型数据库中的地位就犹如英语在世界上的地位。它是数据库系统的通用语言，利用它，用户可以用几乎同样的语句在不同的数据库系统上执行同样的操作。比如“select * from 数据表名”代表要从某个数据表中取出全部数据，在 Oracle 9i、SQL Server 2000、Foxpro 等关系型数据库中都可以使用这条语句。SQL 已经被 ANSI (美国国家标准化组织) 确定为数据库系统的工业标准。

SQL 语言按照功能可以分为 4 大类。

- ❑ 数据查询语言 DQL: 查询数据。
- ❑ 数据定义语言 DDL: 建立、删除和修改数据对象。
- ❑ 数据操纵语言 DML: 完成数据操作的命令，包括查询。
- ❑ 数据控制语言 DCL: 控制对数据库的访问，服务器的关闭、启动等。

4.1.2 SQL 的主要特点

SQL 语言简单易学、风格统一，利用简单的几个英语单词的组合就可以完成所有的功能。在 SQLPlus Worksheet 环境下可以单独使用的 SQL 语句，几乎可以不加修改地嵌入到如 VB、PB 这样的前端开发平台上，利用前端工具的计算能力和 SQL 的数据库操纵能力，可以快速建立数据库应用程序。

4.1.3 Oracle 9i 使用 SQL 的工具

在 Oracle 9i 中为使用 SQL 语言提供了两个主要的工具。

❑ 【SQL Plus】

❑ 【SQLPlus Worksheet】

两种工具在使用上功能都相同，但在可操作性上，【SQLPlus Worksheet】更适合初学者。因此，本书重点介绍后者的使用。

4.1.4 SQL 中访问数据表的方法

在 SQL 语言中访问数据表是通过“用户名.数据表”的形式来进行的。

比如在 Oracle 9i 数据库服务器安装过程中，默认建立有 scott 用户，该用户对 dept 数据表和 emp 数据表有数据查询的权限，因此访问数据表的语句为 `select * from scott.emp`。当然，如果用户是用 scott 用户本身登录的，则访问数据表的语句可以简化为 `select * from emp`，实质是一样的。

在本章的实例中，我们以数据库系统管理员 system、口令 manager 登录数据库，访问数据必须采用 `select * from scott.emp` 的形式。即使是用户本身登录后访问属于自己的数据表，我们也推荐使用“用户名.数据表”的形式来访问数据表以清楚地反映数据表的有权用户信息。

4.1.5 两个范例数据表

在读者没有学习如何创建数据表，如何创建用户，如何将用户赋予对数据表的访问权限之前，我们以数据库已经建立的两个范例数据表为例来介绍。

(1) 启动【SQLPlus Worksheet】

(2) 在【命令编辑区】输入语句“`desc scott.emp`”，然后单击【执行】按钮，出现如图 4.1 所示的 emp 数据表结构。

【参见光盘文件】：\第 4 章\4.1\415-1.sql。



图 4.1 scott.emp 数据表结构



desc, describe 命令的简化形式，作用是显示数据表的结构。使用形式：“desc 数据表名”。

(3) 在【命令编辑区】输入“desc scott.dept”，然后单击【执行】按钮，出现如图 4.2 所示的 scott.dept 数据表结构。

【参见光盘文件】：\第 4 章\4.1\415-2.sql。

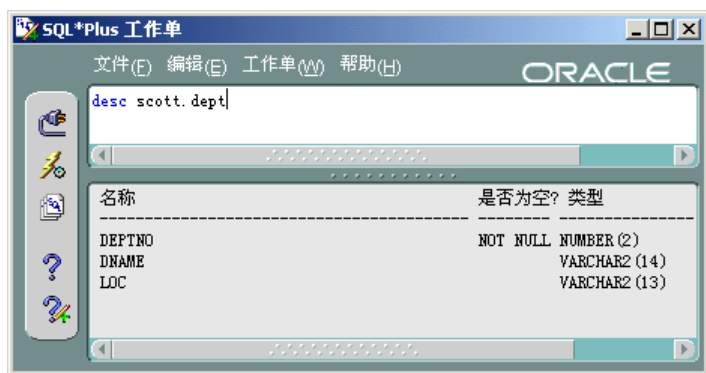


图 4.2 scott.dept 数据表结构

接下来我们以实际查询例子来介绍数据查询的语法，读者可以参照配套光盘的实例跟随本书执行同样的操作。

4.2 用 SQL 进行单表查询

单表查询是相对多表查询而言的，指从一个数据表中查询数据。

4.2.1 查询所有的记录

在【命令编辑区】执行输入“select * from scott.emp”，然后单击【执行】按钮，出现如图 4.3 所示的 emp 数据表所有记录。

【参见光盘文件】：\第 4 章\4.2\421.sql。

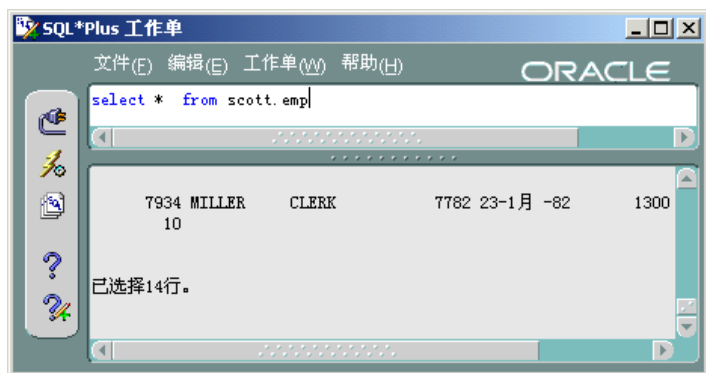


图 4.3 查询 emp 数据表所有记录



select * from 数据表，这里的“*”代表数据表中所有的字段。

4.2.2 查询所有记录的某些字段

在【命令编辑区】输入“select empno,ename,job from scott.emp”，然后单击【执行】按钮，将显示 emp 数据表的 empno、ename 和 job 字段，如图 4.4 所示。

【参见光盘文件】：\第 4 章\4.2\422.sql。



The screenshot shows the SQL*Plus interface with the query 'select empno, ename, job from scott.emp' entered in the command editor. The results are displayed in a table with three columns: EMPNO, ENAME, and JOB. The data is as follows:

EMPNO	ENAME	JOB
7369	SMITH	CLERK
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN
7698	BLAKE	MANAGER
7782	CLARK	MANAGER
7788	SCOTT	ANALYST
7839	KING	PRESIDENT

图 4.4 查询 emp 数据表的 empno、ename 和 job 字段

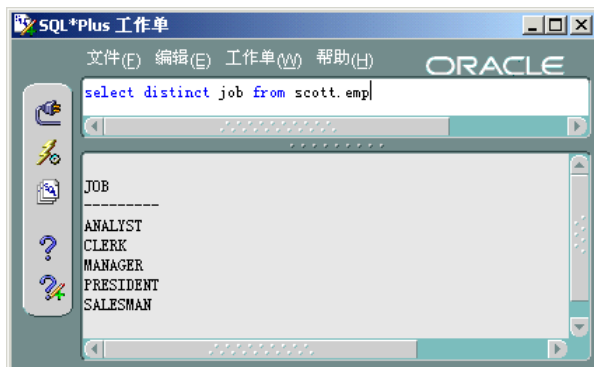


select 字段名 1, 字段名 2,..... from 数据表，将显示某些特定的字段，注意这里的字段名之间的逗号是英文状态下的逗号。

4.2.3 查询某些字段不同记录

在图 4.4 所示的 job 字段中，可以发现相同的数据，为了查询有多少种不同的 job，在【命令编辑区】输入“select distinct job from scott.emp”，然后单击【执行】按钮，出现如图 4.5 所示的结果。

【参见光盘文件】：\第 4 章\4.2\423.sql。



The screenshot shows the SQL*Plus interface with the query 'select distinct job from scott.emp' entered in the command editor. The results are displayed in a table with one column: JOB. The data is as follows:

JOB
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

图 4.5 查询不同的 job 字段



select distinct 字段名 from 数据表，这里的“distinct”保留字指在显示时去除相同的记录，与之对应的是“all”将保留相同的记录，默认为“all”。

4.2.4 单条件的查询

(1) 在【命令编辑区】输入“select empno,ename,job from scott.emp where job='MANAGER'”，然后单击【执行】按钮，出现如图 4.6 所示的字符型字段条件查询的结果，查询的是 job 为 MANAGER 的记录。

【参见光盘文件】：\第 4 章\4.2\424-1.sql。



图 4.6 查询满足字符型字段条件的记录

(2) 在【命令编辑区】输入“select empno,ename,sal from scott.emp where sal<=2500”，然后单击【执行】按钮，出现如图 4.7 所示的数字型字段条件查询的结果，查询的是满足 sal 小于等于 2500 的记录。

【参见光盘文件】：\第 4 章\4.2\424-2.sql。

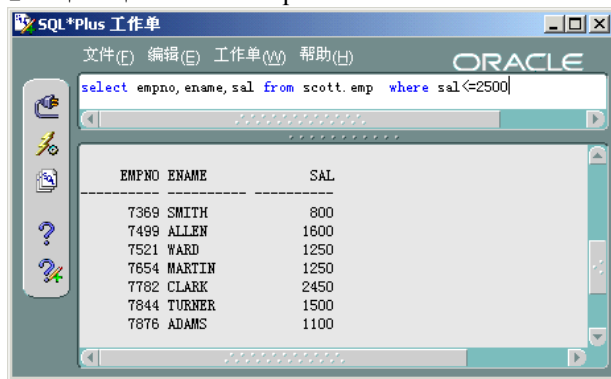


图 4.7 查询满足数字型字段条件的记录



where 可以指定查询条件，如果是指定字符型字段查询条件，形式为字段名 运算符 '字符串'；如果是指定数字型字段查询条件，形式为字段名 运算符 '字符串'。单条件查询使用的比较运算符如表 4.1 所示。

【参见光盘文件】：\第 4 章\4.2\table41.sql。

表 4.1

比较运算符

名称	实例
=(等于)	select * from scott.emp where job='MANAGER';
	select * from scott.emp where sal=1100;

续表

名称	实例
!= (不等于)	select * from scott.emp where job!='MANAGER';
	select * from scott.emp where sal!=1100;
^=(不等于)	select * from scott.emp where job^='MANAGER';
	select * from scott.emp where sal^=1100;
◇(不等于)	select * from scott.emp where job◇'MANAGER';
	select * from scott.emp where sal◇1100;
<(小于)	select * from scott.emp where sal<2000;
	select * from scott.emp where job<'MANAGER';
>(大于)	select * from scott.emp where sal>2000;
	select * from scott.emp where job>'MANAGER';
<=(小于等于)	select * from scott.emp where sal<=2000;
	select * from scott.emp where job<='MANAGER';
>=(大于等于)	select * from scott.emp where sal>=2000;
	select * from scott.emp where job>='MANAGER';
in(列表)	select * from scott.emp where sal in (2000,1000,3000);
	select * from scott.emp where job in ('MANAGER','CLERK');
not in(不在列表)	select * from scott.emp where sal not in (2000,1000,3000);
	select * from scott.emp where job not in ('MANAGER','CLERK');
between(介于之间)	select * from scott.emp where sal between 2000 and 3000;
	select * from scott.emp where job between 'MANAGER' and 'CLERK';
not between (不介于之间)	select * from scott.emp where sal not between 2000 and 3000;
	select * from scott.emp where job not between 'MANAGER' and 'CLERK';
like(模式匹配)	select * from scott.emp where job like 'M%';
	select * from scott.emp where job like 'M__';
not like (模式不匹配)	select * from scott.emp where job not like 'M%';
	select * from scott.emp where job not like 'M__';
Is null (是否为空)	select * from scott.emp where sal is null;
	select * from scott.emp where job is null;
is not null(是否不为空)	select * from scott.emp where sal is not null;
	select * from scott.emp where job is not null;



like 和 not like 适合字符型字段的查询，%代表任意长度的字符串，_下划线代表一个任意的字符。like ‘m%’ 代表 m 开头的任意长度的字符串，like ‘m__’ 代表 m 开头的长度为 3 的字符串。

4.2.5 组合条件的查询

(1) 在【命令编辑区】输入 “select empno,ename,job from scott.emp where job>=‘CLERK’ and sal<=2000”，然后单击【执行】按钮，出现如图 4.8 所示的逻辑与组合查询的结果。

【参见光盘文件】：\第 4 章\4.2\425-1.sql。

SQL*Plus 工作单

文件(E) 编辑(E) 工作单(W) 帮助(H) ORACLE

```
select empno,ename,job,sal from scott.emp
where job>= 'CLERK' and sal<=2000
```

7499	ALLEN	SALESMAN	1800
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300

已选择8行。

图 4.8 逻辑与组合查询

(2) 在【命令编辑区】输入 “select empno,ename,job from scott.emp where job>=‘CLERK’ or sal<=2000”，然后单击【执行】按钮，出现如图 4.9 所示的逻辑或组合查询的结果。

【参见光盘文件】：\第 4 章\4.2\425-2.sql。

SQL*Plus 工作单

文件(E) 编辑(E) 工作单(W) 帮助(H) ORACLE

```
select empno,ename,job,sal from scott.emp
where job>= 'CLERK' or sal<=2000
```

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7499	ALLEN	SALESMAN	1800
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850

图 4.9 逻辑或组合查询

(3) 在【命令编辑区】输入 “select empno,ename,job from scott.emp where not job=‘CLERK’”，然后单击【执行】按钮，出现如图 4.10 所示的逻辑非组合查询的结果。

【参见光盘文件】：\第 4 章\4.2\425-3.sql。

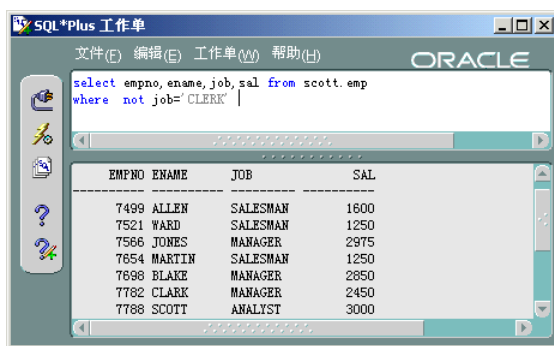


图 4.10 逻辑非组合查询



“not job='CLERK'” 等价于 “job<>'CLERK'”。

组合条件中使用的逻辑比较符如表 4.2 所示。

【参见光盘文件】: \第 4 章\4.2\table42.sql。

表 4.2 逻辑比较符

名称	实例
and(与)	select * from scott.emp where job='MANAGER' and sal<>2000;
or(或)	select * from scott.emp where job!='MANAGER' or sal<>2000;
not(非)	select * from scott.emp where not job>='MANAGER';

4.2.6 排序查询

在【命令编辑区】输入 “select empno,ename,job from scott.emp where job<='CLERK' order by job asc,sal desc”，然后单击【执行】按钮，出现如图 4.11 所示的排序查询的结果。

【参见光盘文件】: \第 4 章\4.2\426.sql。

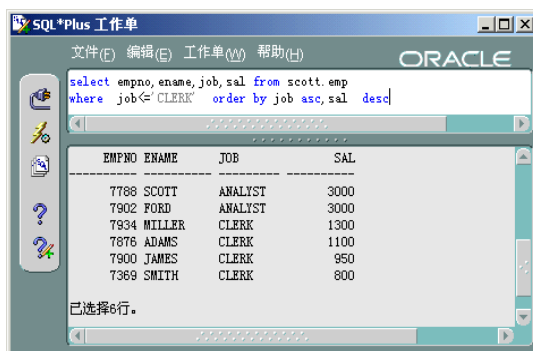


图 4.11 排序查询



order by 可以指定查询结果如何排序，形式为字段名 排序关键词；asc 代表升序排列，desc 代表降序排列，多个排序字段之间通过逗号分割。若有 where 查询条件，order

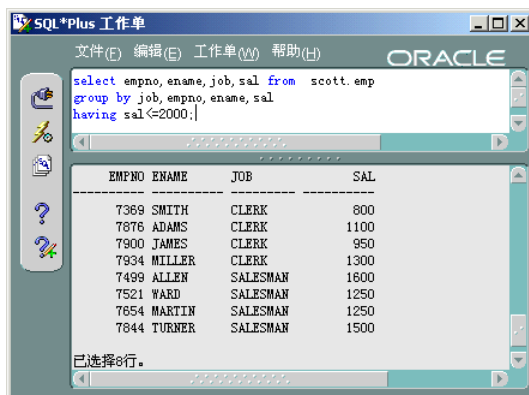
by 要放在 where 语句后面。

4.2.7 分组查询

分组查询是指将查询结果按照字段分组。

(1) 在【命令编辑区】输入“select empno,ename,job,sal from scott.emp group by job,empno,ename,sal having sal<=2000”，然后单击【执行】按钮，出现如图 4.12 所示的分组查询的结果。

【参见光盘文件】：\第4章\4.2\427-1.sql。



SQL*Plus 工作单

文件(F) 编辑(E) 工作单(W) 帮助(H) ORACLE

```
select empno,ename,job,sal from scott.emp
group by job,empno,ename,sal
having sal<=2000.
```

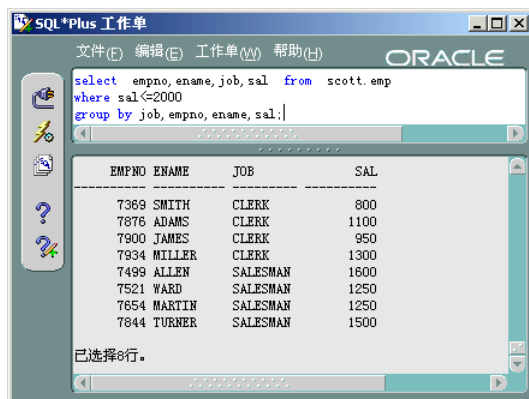
EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500

已选择8行。

图 4.12 使用 having 子句的分组查询

(2) 在【命令编辑区】输入“select empno,ename,job,sal from scott.emp where sal<=2000 group by job,empno,ename,sal”，然后单击【执行】按钮，出现如图 4.13 所示的分组查询的结果。

【参见光盘文件】：\第4章\4.2\427-2.sql。



SQL*Plus 工作单

文件(F) 编辑(E) 工作单(W) 帮助(H) ORACLE

```
select empno,ename,job,sal from scott.emp
where sal<=2000
group by job,empno,ename,sal.
```

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7934	MILLER	CLERK	1300
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500

已选择8行。

图 4.13 使用 where 子句的分组查询



where 检查每条记录是否符合条件，having 是检查分组后的各组是否满足条件。having 语句只能配合 group by 语句使用，没有 group by 时不能使用 having，但可以使用 where。

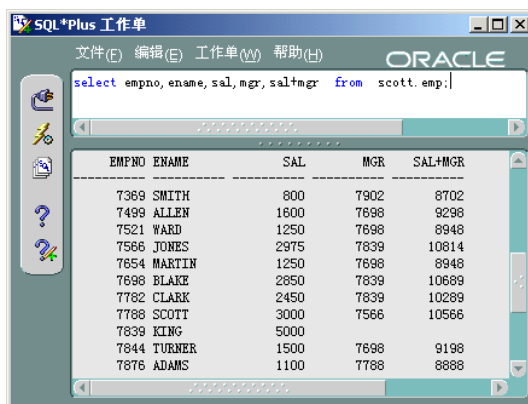
4.2.8 字段运算查询

可以利用几种基本的算术运算符来查询数据。

常见的+（加）、-（减）、*（乘）、/（除）4种算术运算都可以用来查询数据。

在【命令编辑区】输入“select empno,ename,sal,mgr,sal+mgr from scott.emp”，然后单击【执行】按钮，出现如图 4.14 所示的结果。

【参见光盘文件】：\第 4 章\4.2\428.sql。



SQL*Plus 工作单

文件(F) 编辑(E) 工作单(W) 帮助(H) ORACLE

select empno,ename,sal,mgr,sal+mgr from scott.emp;

EMPNO	ENAME	SAL	MGR	SAL+MGR
7369	SMITH	800	7902	8702
7499	ALLEN	1600	7698	9298
7521	WARD	1250	7698	8948
7566	JONES	2975	7839	10814
7654	MARTIN	1250	7698	8948
7698	BLAKE	2850	7839	10689
7782	CLARK	2450	7839	10289
7788	SCOTT	3000	7566	10566
7839	KING	5000		
7844	TURNER	1500	7698	9198
7876	ADAMS	1100	7788	8888

图 4.14 字段运算查询结果



利用算术运算符仅仅适合多个数值型字段或字段与数字之间的运算。

4.2.9 变换查询显示

在【命令编辑区】输入“select empno 编号,ename 姓名,job 工作,sal 薪水 from scott.emp”，然后单击【执行】按钮，出现如图 4.15 所示的结果，可以将默认的字段名以设定的名称显示。

【参见光盘文件】：\第 4 章\4.2\429.sql。



SQL*Plus 工作单

文件(F) 编辑(E) 工作单(W) 帮助(H) ORACLE

select empno 编号,ename 姓名,job 工作,sal 薪水
from scott.emp;

编号	姓名	工作	薪水
7369	SMITH	CLERK	800
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000

图 4.15 变换查询显示

以上我们学习了对单个数据表的查询语句。将上面这些基本的实例经过组合，就可以完

成基本的日常数据查询任务，接下来进一步学习多表查询。

4.3 用 SQL 进行多表查询

所谓多表查询是相对单表而言的，指从多个数据表中查询数据，这里我们主要学习从两个数据表中如何查询数据的方法。

4.3.1 无条件多表查询

无条件多表查询是将各表的记录以“笛卡尔”积的方式组合起来。

如 scott.dept 表共有 4 条记录，scott.emp 表共有 14 条记录，其“笛卡尔”积将有 $4 \times 14 = 56$ 条记录。

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.deptno, dept.dname, dept.loc
from scott.emp, scott.dept;
```

单击【执行】按钮，出现如图 4.16 所示的结果。

【参见光盘文件】：\第 4 章\4.3\431.sql。

EMPNO	ENAME	DEPTNO	DNAME	LOC
7369	SMITH	20	ACCOUNTING	NEW YORK
7499	ALLEN	30	ACCOUNTING	NEW YORK
7521	WARD	30	ACCOUNTING	NEW YORK
7566	JONES	20	ACCOUNTING	NEW YORK
7654	MARTIN	30	ACCOUNTING	NEW YORK
7698	BLAKE	30	ACCOUNTING	NEW YORK
7782	CLARK	10	ACCOUNTING	NEW YORK
7788	SCOTT	20	ACCOUNTING	NEW YORK
7839	KING	10	ACCOUNTING	NEW YORK
7844	TURNER	30	ACCOUNTING	NEW YORK

图 4.16 无条件多表查询

4.3.2 等值多表查询

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.deptno, dept.dname, dept.loc
from scott.emp, scott.dept
where scott.emp.deptno=scott.dept.deptno;
```

单击【执行】按钮，出现如图 4.17 所示的结果。

【参见光盘文件】: \第 4 章\4.3\432.sql。

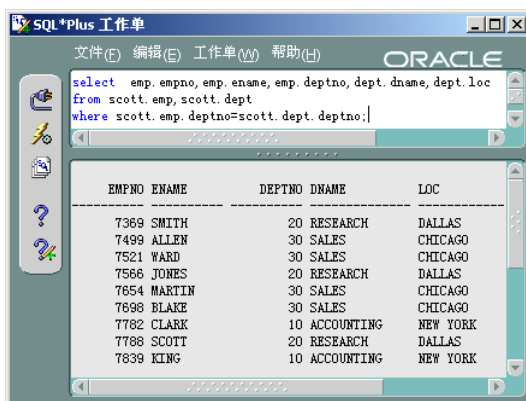


图 4.17 等值多表查询



等值多表查询将按照等值的条件查询多个数据表中关联的数据。要求关联的多个数据表的某些字段具有相同的属性，即具有相同的数据类型、宽度和取值范围。

4.3.3 非等值多表查询

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.deptno, dept.dname, dept.loc
from scott.emp, scott.dept
where scott.emp.deptno!=scott.dept.deptno and scott.emp.deptno=10;
```

单击【执行】按钮，出现如图 4.18 所示的结果。

【参见光盘文件】: \第 4 章\4.3\433.sql。



图 4.18 非等值多表查询

在非等值多表查询中，读者可以使用表 4.1 所示的比较运算符来组合查询条件。

4.4 用 SQL 进行嵌套查询

在 select 查询语句里可以嵌入 select 查询语句，称为嵌套查询。有些书上将内嵌的 select 语句称为子查询，子查询形成的结果又成为父查询的条件。



子查询可以嵌套多层，子查询操作的数据表可以是父查询不操作的数据表。子查询中不能有 order by 分组语句。

4.4.1 简单嵌套查询

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal >= (select sal from scott.emp where ename='WARD');
```

单击【执行】按钮，出现如图 4.19 所示的结果。

【参见光盘文件】：\第 4 章\4.4\441.sql。

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7844	TURNER	SALESMAN	1500
7902	FORD	ANALYST	3000
7934	MILLER	CLERK	1300

图 4.19 简单嵌套查询

在这段代码中，子查询 select sal from scott.emp where ename='WARD' 的含义是从 emp 数据表中查询姓名为 WARD 的员工的薪水，父查询的含义是要找出 emp 数据表中薪水大于等于 WARD 的薪水的员工。上面的查询过程等价于两步的执行过程。

(1) 执行 “select sal from scott.emp where ename='WARD'”，得出 sal=1250；

(2) 执行 “select emp.empno, emp.ename, emp.job, emp.sal from scott.emp where sal >= 1250;”

4.4.2 带【in】的嵌套查询

在【命令编辑区】执行下列语句。

```

select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal in (select sal from scott.emp where ename='WARD');

```

单击【执行】按钮，出现如图 4.20 所示的结果。

【参见光盘文件】：\第 4 章\4.4\442.sql。

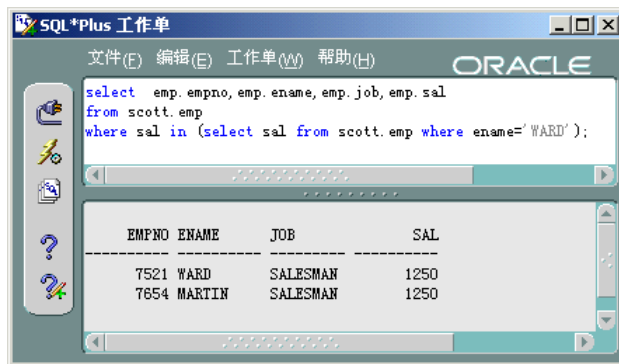


图 4.20 带【in】的嵌套查询

上述语句完成的是查询薪水和 WARD 相等的员工，也可以使用【not in】来进行查询。

4.4.3 带【any】的嵌套查询

在【命令编辑区】执行下列语句。

```

select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal > any(select sal from scott.emp where job='MANAGER');

```

单击【执行】按钮，出现如图 4.21 所示的结果。

【参见光盘文件】：\第 4 章\4.4\443.sql。

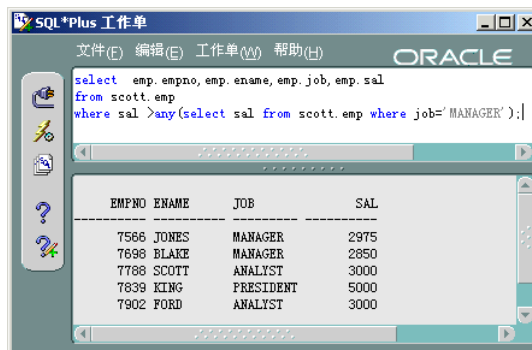


图 4.21 带【any】的嵌套查询

带 any 的查询过程等价于两步的执行过程。

(1) 执行 “select sal from scott.emp where job='MANAGER'”, 其结果如图 4.22 所示。

【参见光盘文件】: \第4章\4.4\443-1.sql。

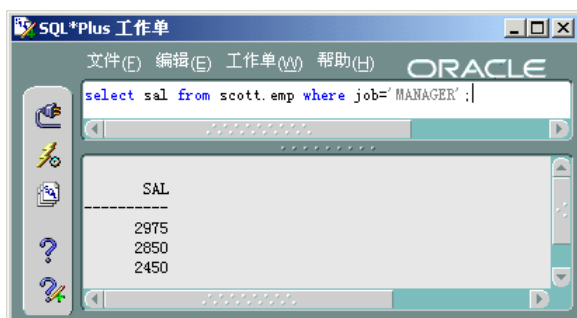


图 4.22 查询工作为 MANAGER 的员工的薪水

(2) 查询到 3 个薪水值 2975、2850 和 2450, 父查询执行下列语句。

【参见光盘文件】: \第4章\4.4\443-2.sql。

```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal >2975 or sal>2850 or sal>2450;
```

4.4.4 带【some】的嵌套查询

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal =some(select sal from scott.emp where job='MANAGER');
```

单击【执行】按钮, 出现如图 4.23 所示的结果。

【参见光盘文件】: \第4章\4.4\444.sql。

带 some 的嵌套查询与 any 的步骤相同。

(1) 子查询, 执行 “select sal from scott.emp where job='MANAGER'”, 其结果如图 4.22 所示。

(2) 父查询执行下列语句。

```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal =2975 or sal=2850 or sal=2450;
```

【参见光盘文件】: \第4章\4.4\444-2.sql。

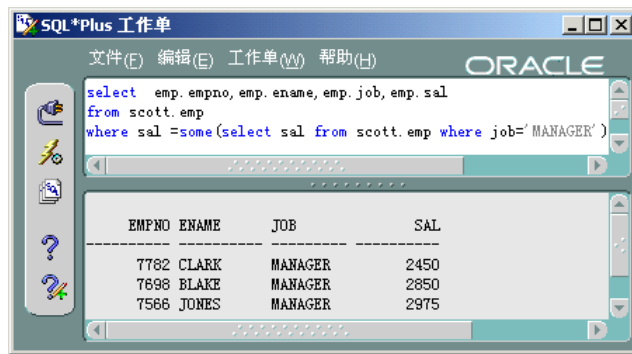


图 4.23 带【some】的嵌套查询



带【any】的嵌套查询和【some】的嵌套查询功能是一样的。早期的 SQL 仅仅允许使用【any】，后来的版本为了和英语的【any】相区分，引入了【some】，同时还保留了【any】关键词。

4.4.5 带【all】的嵌套查询

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal >all(select sal from scott.emp where job='MANAGER');
```

单击【执行】按钮，出现如图 4.24 所示的结果。

【参见光盘文件】：\第 4 章\4.4\445.sql。

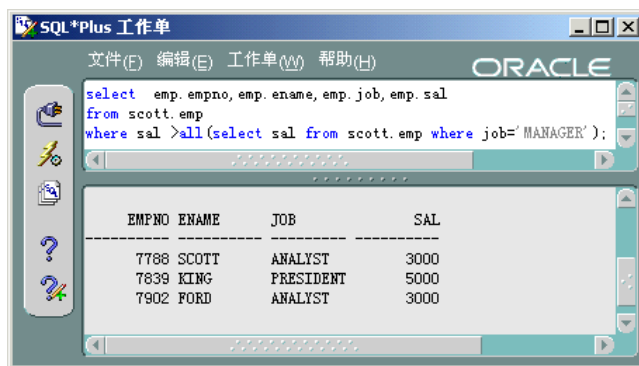


图 4.24 带【all】的嵌套查询

带 all 的嵌套查询与【some】的步骤相同。

- (1) 子查询，结果如图 4.22 所示。
- (2) 父查询执行下列语句。


```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp
where sal >2975 and sal>2850 and sal>2450;
```

【参见光盘文件】：\第4章\4.4\445-2.sql。

4.4.6 带【exists】的嵌套查询

在【命令编辑区】执行下列语句。

```
select emp.empno, emp.ename, emp.job, emp.sal
from scott.emp, scott.dept
where exists
(select * from scott.emp where scott.emp.deptno=scott.dept.deptno);
```

单击【执行】按钮，出现如图 4.25 所示的结果。

【参见光盘文件】：\第4章\4.4\446.sql。

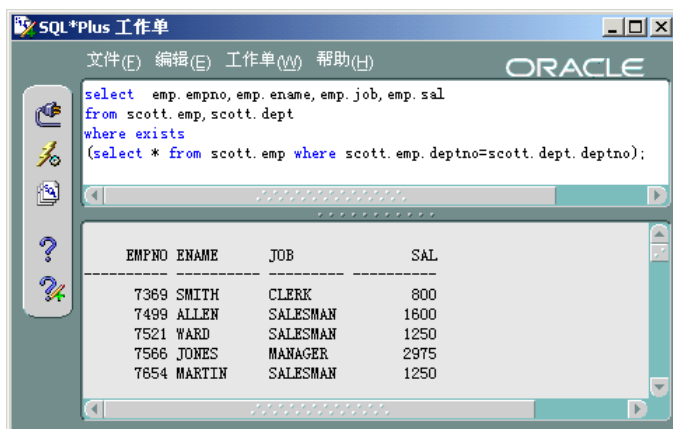


图 4.25 带【exists】的嵌套查询

4.4.7 并操作的嵌套查询

并操作就是集合中并集的概念。属于集合 A 或集合 B 的元素总和就是并集。

在【命令编辑区】执行下列语句。

```
(select deptno from scott.emp)
union
(select deptno from scott.dept);
```

单击【执行】按钮，出现如图 4.26 所示的结果。

【参见光盘文件】：\第4章\4.4\447.sql。

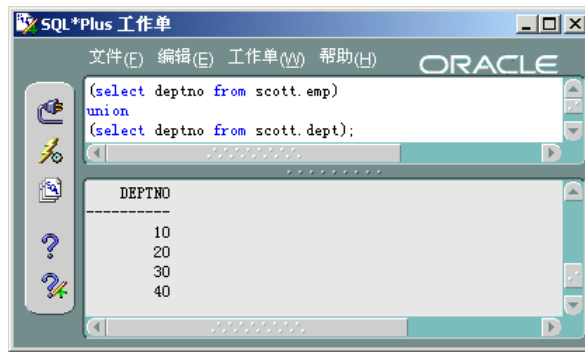


图 4.26 并操作的嵌套查询

4.4.8 交操作的嵌套查询

交操作就是集合中交集的概念。属于集合 A 且属于集合 B 的元素总和就是交集。
在【命令编辑区】执行下列语句。

```
(select deptno from scott.emp)
intersect
(select deptno from scott.dept);
```

单击【执行】按钮，出现如图 4.27 所示的结果。

【参见光盘文件】: \第 4 章\4.4\448.sql。

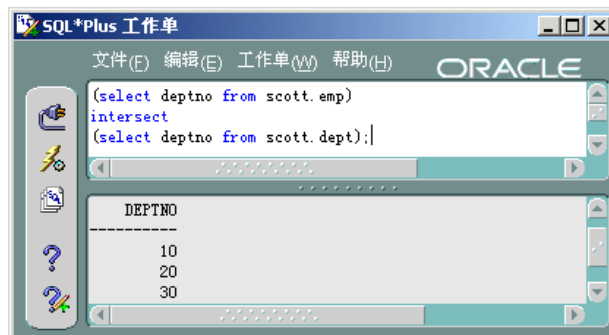


图 4.27 交操作的嵌套查询

4.4.9 差操作的嵌套查询

差操作就是集合中差集的概念。属于集合 A 且不属于集合 B 的元素总和就是差集。
在【命令编辑区】执行下列语句。

```
(select deptno from scott.dept)
minus
```

```
(select deptno from scott.emp);
```

单击【执行】按钮，出现如图 4.28 所示的结果。

【参见光盘文件】：\第 4 章\4.4\449.sql。



图 4.28 差操作的嵌套查询



并、交和差操作的嵌套查询要求属性具有相同的定义，包括类型和取值范围。

4.5 用 SQL 进行函数查询

Oracle 9i 提供了很多函数可以用来辅助数据查询。接下来我们介绍常用的函数功能及使用方法。

4.5.1 【ceil】函数

在【命令编辑区】输入“select mgr, mgr/100,ceil(mgr/100) from scott.emp;”，然后单击【执行】按钮，出现如图 4.29 所示的结果。

【参见光盘文件】：\第 4 章\4.5\451.sql。

MGR	MGR/100	CEIL(MGR/100)
7902	79.02	80
7698	76.98	77
7698	76.98	77
7839	78.39	79
7698	76.98	77
7839	78.39	79
7839	78.39	79
7566	75.66	76

图 4.29 【ceil】函数的使用



【ceil】函数用法：ceil (n)，取大于等于数值 n 的最小整数。

4.5.2 【floor】函数

在【命令编辑区】输入“select mgr, mgr/100,floor(mgr/100) from scott.emp;”，然后单击【执行】按钮，出现如图 4.30 所示的结果。

【参见光盘文件】：\第 4 章\4.5\452.sql。

SQL*Plus 工作单

文件(E) 编辑(E) 工作单(W) 帮助(H) ORACLE

select mgr, mgr/100,floor(mgr/100) from scott.emp;

MGR	MGR/100	FLOOR(MGR/100)
7902	79.02	79
7698	76.98	76
7698	76.98	76
7839	78.39	78
7698	76.98	76
7839	78.39	78
7839	78.39	78
7566	75.66	75

图 4.30 【floor】函数的使用



【floor】函数用法：floor (n)，取小于等于数值 n 的最大整数。

4.5.3 【mod】函数

在【命令编辑区】输入“select mgr, mod(mgr,1000), mod(mgr,100), mod(mgr,10) from scott.emp;”，然后单击【执行】按钮，出现如图 4.31 所示的结果。

【参见光盘文件】：\第 4 章\4.5\453.sql。

SQL*Plus 工作单

文件(E) 编辑(E) 工作单(W) 帮助(H) ORACLE

select mgr, mod(mgr,1000), mod(mgr,100), mod(mgr,10) from scott.emp;

MGR	MOD(MGR, 1000)	MOD(MGR, 100)	MOD(MGR, 10)
7902	902	2	2
7698	698	98	8
7698	698	98	8
7839	839	39	9
7698	698	98	8
7839	839	39	9
7839	839	39	9

图 4.31 【mod】函数的使用



【mod】函数用法：mod (m,n)，取 m 整除 n 后的余数。

4.5.4 【power】函数

在【命令编辑区】输入“select mgr, power(mgr,2),power(mgr,3) from scott.emp;”，然后单击【执行】按钮，出现如图 4.32 所示的结果。

【参见光盘文件】：\第4章\4.5\454.sql。

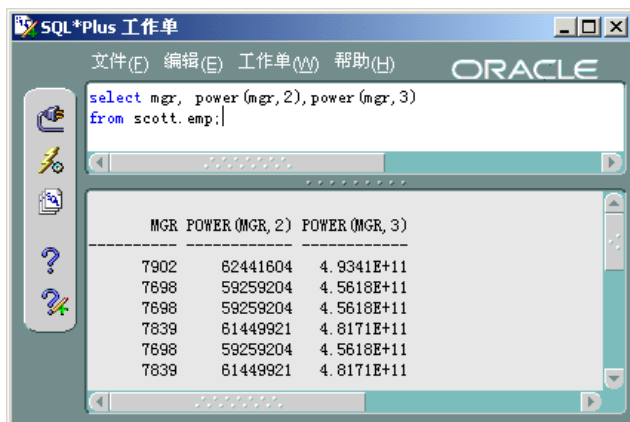


图 4.32 【power】函数的使用



【power】函数用法：power (m,n)，取 m 的 n 次方。

4.5.5 【round】函数

在【命令编辑区】输入“select mgr, round(mgr/100,2),round(mgr/1000,2) from scott.emp;”，然后单击【执行】按钮，出现如图 4.33 所示的结果。

【参见光盘文件】：\第4章\4.5\455.sql。

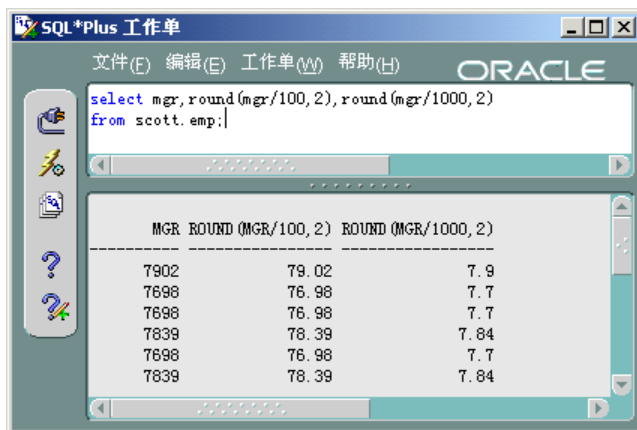


图 4.33 【round】函数的使用

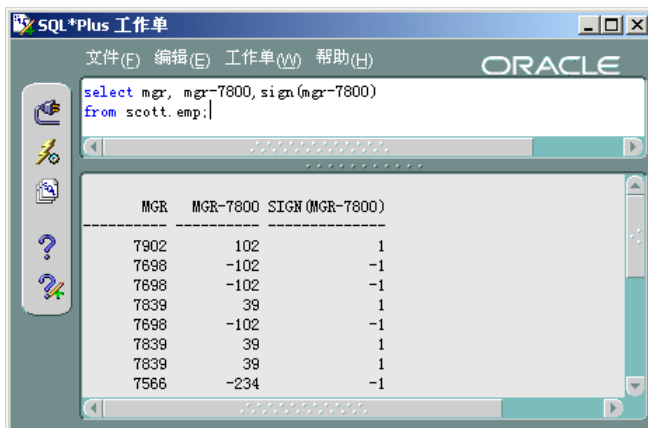


【round】函数用法：round (m,n)，四舍五入，保留 n 位。

4.5.6 【sign】函数

在【命令编辑区】输入“select mgr, mgr-7800, sign(mgr-7800) from scott.emp;”，然后单击【执行】按钮，出现如图 4.34 所示的结果。

【参见光盘文件】：\第 4 章\4.5\456.sql。



The screenshot shows the SQL*Plus interface with the query 'select mgr, mgr-7800, sign(mgr-7800) from scott.emp;' entered in the command editor. The results are displayed in a table with three columns: MGR, MGR-7800, and SIGN(MGR-7800). The data rows are as follows:

MGR	MGR-7800	SIGN(MGR-7800)
7902	102	1
7698	-102	-1
7698	-102	-1
7839	39	1
7698	-102	-1
7839	39	1
7839	39	1
7566	-234	-1

图 4.34 【sign】函数的使用

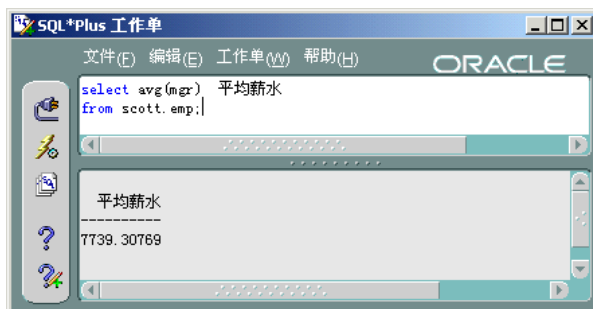


【sign】函数用法：sign (n)。n>0，取 1；n=0，取 0；n<0，取-1。

4.5.7 【avg】函数

在【命令编辑区】输入“select avg(mgr) 平均薪水 from scott.emp;”，然后单击【执行】按钮，出现如图 4.35 所示的结果。

【参见光盘文件】：\第 4 章\4.5\457.sql。



The screenshot shows the SQL*Plus interface with the query 'select avg(mgr) 平均薪水 from scott.emp;' entered in the command editor. The results are displayed in a table with one column: 平均薪水. The data row is as follows:

平均薪水
7739.30769

图 4.35 【avg】函数的使用



【avg】函数用法：avg (字段名)，求平均值。要求字段为数值型。

4.5.8 【count】函数

(1) 在【命令编辑区】输入“select count(*) 记录总数 from scott.emp;”，然后单击【执行】按钮，出现如图 4.36 所示的结果。

【参见光盘文件】: \第4章\4.5\458-1.sql。

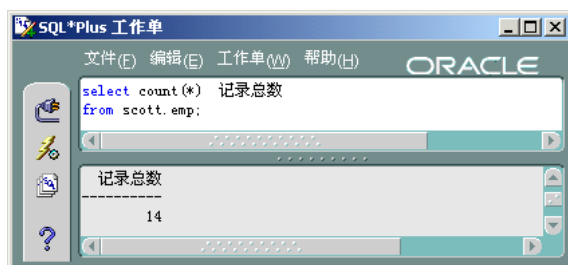


图 4.36 【count(*)】函数的使用

(2) 在【命令编辑区】输入“select count(distinct job) 工作类别总数 from scott.emp;”，然后单击【执行】按钮，出现如图 4.37 所示的结果。

【参见光盘文件】: \第4章\4.5\458-2.sql。



图 4.37 【count(字段名)】函数的使用



【count】函数用法：count (字段名) 或 count (*)，统计总数。

4.5.9 【min】函数

在【命令编辑区】输入“select min(sal) 最少薪水 from scott.emp;”，然后单击【执行】按钮，出现如图 4.38 所示的结果。

【参见光盘文件】: \第4章\4.5\459.sql。

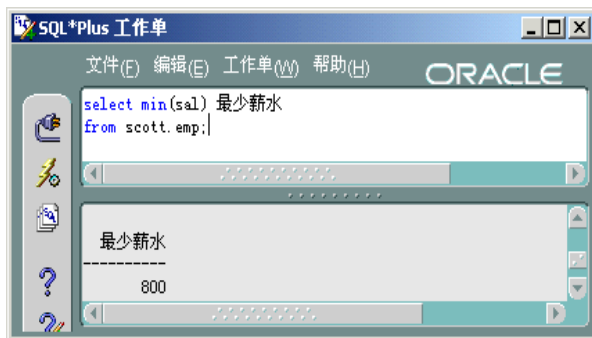


图 4.38 【min】函数的使用



【min】函数用法：min（字段名），计算数值型字段最小数。

4.5.10 【max】函数

在【命令编辑区】输入“select max(sal) 最高薪水 from scott.emp;”，然后单击【执行】按钮，出现如图 4.39 所示的结果。

【参见光盘文件】： \第 4 章\4.5\4510.sql。

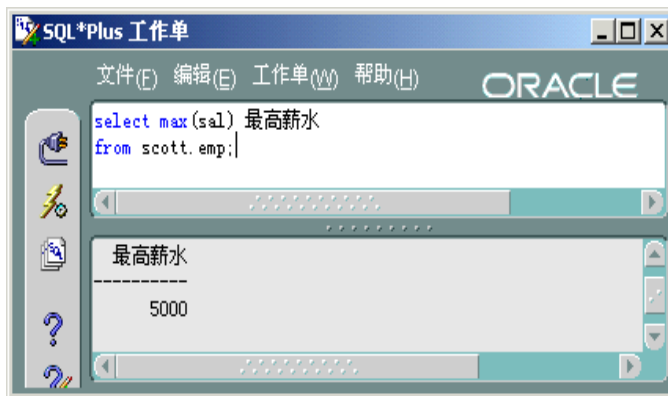


图 4.39 【max】函数的使用



【max】函数用法：max（字段名），计算数值型字段最大数。

4.5.11 【sum】函数

在【命令编辑区】输入“select sum(sal) 薪水总和 from scott.emp;”，然后单击【执行】按钮，出现如图 4.40 所示的结果。

【参见光盘文件】： \第 4 章\4.5\4511.sql。

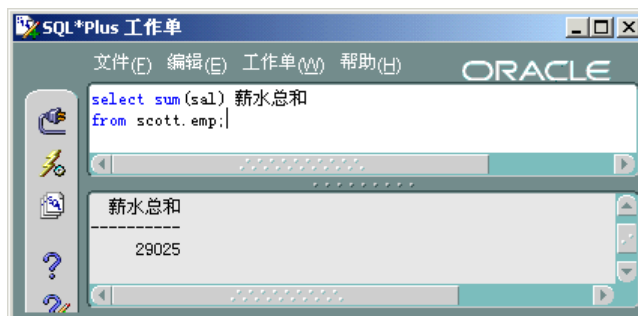


图 4.40 【sum】函数的使用



【sum】函数用法：sum（字段名），计算数值型字段总和。

通过上面 4 类查询实例的学习，读者可以举一反三，灵活运用。用 SQL 进行数据的查询就介绍到这里，下面学习如何录入数据。

4.6 用 SQL 录入数据

数据的录入采用【insert】语句。对应不同的录入方式，【insert】语句的语法会有所变化。

4.6.1 单行记录的录入

1. 语法

insert into 数据表(字段名 1, 字段名 2, ……) values(字段名 1 的值, 字段名 2 的值, ……)。

由于字段的类型不同，在书写字段值的时候要注意格式。

数值型字段，可以直接写值。

字符型字段，其值上要加上单引号。

日期型字段，其值上要加上单引号，同时还要注意年、月、日的排列次序。



在数据的插入语句中，插入列排序和插入值要一一对应。字符型和日期型字段要加上单引号，非空列必须有值。

2. 实例

在 scott.emp 数据表里共包含了 3 种类型的字段。

empno, number (4), NOT NULL, 数值型，长度为 4，不能为空。

ename, varchar2(10), 字符型，长度为 10。

hiredate, date, 日期型。

我们以在这 3 个字段中插入记录为例进行说明。

对于日期型的数据，读者往往会感觉为难，因为不知道年、月、日的排列顺序和格式，这里教大家几个方法。首先查询范例数据表中的数据，然后“依葫芦画瓢”就可以了。

(1) 在【命令编辑区】输入“select empno, ename, hiredate from scott.emp;”，然后单击【执行】按钮，出现如图 4.41 所示的结果。因此，笔者的计算机系统默认的日期型数据格式应该为“日一月一年”。

【参见光盘文件】：\第 4 章\4.6\461-1.sql。

EMPNO	ENAME	HIREDATE
7369	SMITH	17-12月-80
7499	ALLEN	20-2月-81
7521	WARD	22-2月-81
7566	JONES	02-4月-81
7654	MARTIN	28-9月-81
7698	BLAKE	01-5月-81

图 4.41 查询数据表中的数据

(2) 在【命令编辑区】输入“insert into scott.emp(empno, ename, hiredate) values (7999, 'JONE', '25-11月-2002');”，然后单击【执行】按钮，出现如图 4.42 所示的结果。

【参见光盘文件】: \第 4 章\4.6\461-2.sql。



图 4.42 成功插入数据

(3) 在【命令编辑区】输入“select * from scott.emp where empno=7999;”，然后单击【执行】按钮，出现如图 4.43 所示的结果。

【参见光盘文件】: \第 4 章\4.6\461-3.sql。



图 4.43 查询录入的数据

4.6.2 多行记录的录入

在数据的录入中，经常需要将数据表中查询到的数据稍做修改成批录入的情况，这就是多行数据的录入。

1. 语法

```
insert into 数据表(字段名 1,字段名 2,.....)
(select(字段名 1 或运算, 字段名 2 或运算,.....) from 数据表
where 条件)
```



实际上，首先利用子查询语句查询结果，然后再利用 insert 语句将结果插入数据表。子查询和 insert 中的数据表既可以相同，也可以不同，但要求查询结果的字段和 insert 插入的数据表中字段属性完全一致。

2. 实例

在【命令编辑区】执行以下语句。

```
insert into scott.emp(empno,ename,hiredate)
(select empno+100,ename,hiredate from scott.emp
```

```
where empno>=6999
);
```

【参见光盘文件】：\第4章\4.6\462.sql。

单击【执行】按钮，出现如图4.44所示的结果。

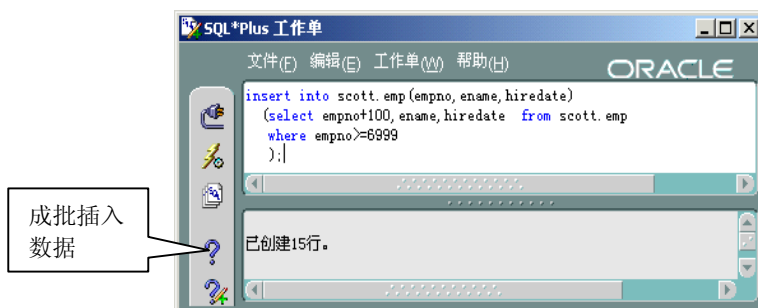


图 4.44 成批插入数据

4.6.3 表间数据复制

可以从一个数据表中选择需要的数据插入到全新的数据表中。

(1) 在【命令编辑区】执行以下语句。

```
create table scott.test
as
(
  select distinct empno,ename,hiredate
  from scott.emp
  where empno>=7000
);
```

【参见光盘文件】：\第4章\4.6\463.sql。



图 4.45 成批创建数据表并复制数据

然后单击【执行】按钮，出现如图 4.45 所示的结果。

上述语句的功能是创建一个名为 `scott.test` 的数据表，表结构包含 3 个字段。并将 `scott.emp` 中具有不同的 `empno` 字段，且 `empno >= 7000` 的数据复制到 `scott.test` 数据表中。

(2) 在【命令编辑区】输入“`select * from scott.test;`”语句，然后单击【执行】按钮，出现如图 4.46 所示的结果。

The screenshot shows the SQL*Plus interface with the command 'select * from scott.test;' entered in the command editor. Below the editor, the query results are displayed in a table format with three columns: EMPNO, ENAME, and HIREDATE. The results list seven employees with their respective IDs, names, and hire dates.

EMPNO	ENAME	HIREDATE
7369	SMITH	17-12月-80
7469	SMITH	17-12月-80
7499	ALLEN	20-2月-81
7521	WARD	22-2月-81
7566	JONES	02-4月-81
7599	ALLEN	20-2月-81
7621	WARD	22-2月-81

图 4.46 查询新数据表 test 的数据



这里的 `create table` 语句的功能是创建新的数据表，上述过程实际是分 3 步执行的。首先查询符合要求的数据，其次建立 3 个字段的名为 `test` 的数据空表，最后是将查询的数据插入到 `test` 数据表中。

4.7 用 SQL 删除数据

使用【`delete`】命令可以删除数据，使用【`truncate`】命令可以删除整表数据但保留结构。

4.7.1 删除记录

在【命令编辑区】输入“`delete from scott.test where empno >= 7500 and empno <= 8000;`”，然后单击【执行】按钮，出现如图 4.47 所示的结果。

【参见光盘文件】：\第 4 章\4.7\471.sql。



图 4.47 删除数据表 test 中的数据



删除记录的语法：`delete from 数据表 where 条件。`

4.7.2 整表数据删除

在【命令编辑区】输入“truncate table scott.test ;”，然后单击【执行】按钮，出现如图 4.48 所示的结果。

【参见光盘文件】：\第 4 章\4.7\472.sql。



图 4.48 删除数据表 test 中的所有数据



truncate table 命令将快速删除数据表中的所有记录，但保留数据表结构。这种快速删除与 delete from 数据表的删除全部数据表记录不一样，delete 命令删除的数据将存储在系统回滚段中，需要的时候，数据可以回滚恢复，而 truncate 命令删除的数据是不可恢复的。

4.8 用 SQL 更新数据

更新数据使用的是【update】命令。

4.8.1 直接赋值更新

1. 语法

update 数据表

set 字段名 1=新的赋值,字段名 2=新的赋值,.....

where 条件

2. 实例

在【命令编辑区】执行以下语句。

```
-----
update scott.emp
set empno=8888,ename='TOM',hiredate='03-9月 -2002 '
where empno=7566;
-----
```

【参见光盘文件】：\第 4 章\4.8\481.sql。

单击【执行】按钮，出现如图 4.49 所示的结果。



图 4.49 更新数据表中的数据

4.8.2 嵌套更新

1. 语法

update 数据表

set 字段名 1=(select 字段列表 from 数据表 where 条件), 字段名 2=(select 字段列表 from 数据表 where 条件),.....

2. 实例

在【命令编辑区】执行以下语句。

```
update scott.emp  
set sal=  
(  
    select sal+300 from scott.emp  
    where empno=7599  
)  
where empno=7599;
```

【参见光盘文件】: \第 4 章\4.8\482.sql。

单击【执行】按钮，出现如图 4.50 所示的结果。



图 4.50 嵌套更新数据表中的数据

以上我们学习了如何利用 SQL 对数据表中的数据进行录入、删除、更新和查询操作。读者利用这些实例可以结合自己的实际快速掌握这些基本的数据管理语句。

4.9 习题

- (1) SQL 有哪些主要特点，SQL 的用途有哪些？
- (2) 在 Oracle 9i 中，SQL 如何访问数据表？
- (3) 结合光盘配套文件练习掌握 SQL 查询语句的使用方法。
- (4) 结合光盘配套文件练习掌握 SQL 录入语句的使用方法。
- (5) 结合光盘配套文件练习掌握 SQL 删除语句的使用方法。
- (6) 结合光盘配套文件练习掌握 SQL 更新语句的使用方法。

第 5 章 智能管理——用好集成管理环境

Oracle 9i 提供了集成化、智能化的基于【管理服务器】的管理环境。

5.1 如何定制集成的管理环境

5.1.1 节点

1. HTTP 服务器

Oracle 9i 数据库服务器软件在安装时，若按照【典型】配置，将在同一台物理计算机上安装一个内嵌的 Apache 1.3 Web 服务器，也称为 HTTP 服务器。

2. 数据库

显示了节点上安装的数据库。

3. 监听程序

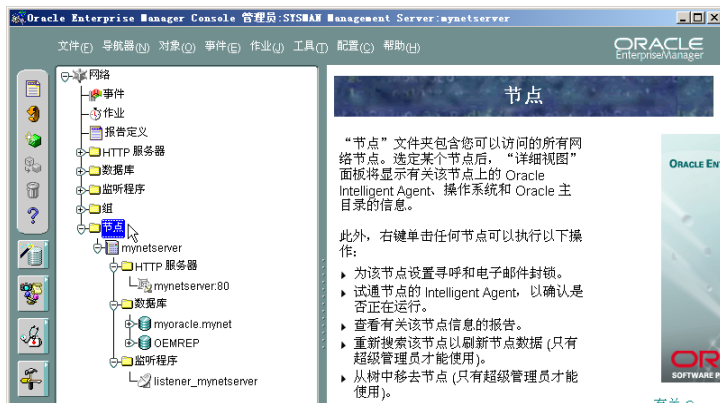


图 5.1 节点管理环境

5.1.2 组

(1) 在【企业管理器】的【管理目标导航树】下选择【网络】/【组】选项，用鼠标右键单击，在出现的快捷菜单里选择【创建】选项，如图 5.2 所示。

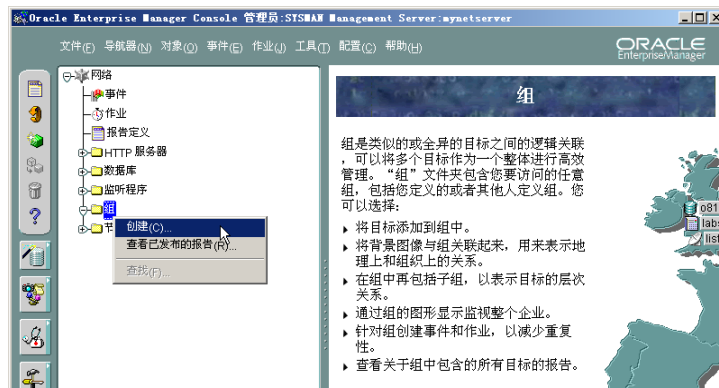


图 5.2 组管理环境

(2) 出现如图 5.3 所示的创建组的【一般信息】选项卡。

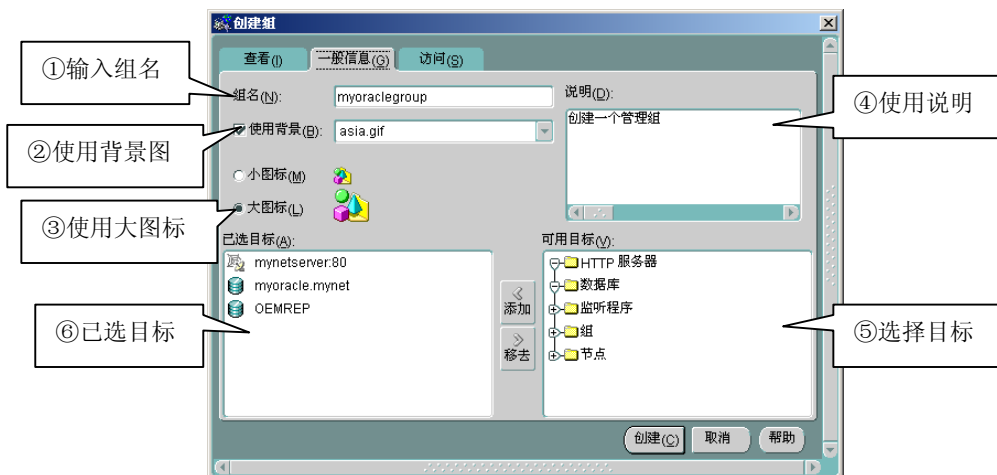


图 5.3 创建组的【一般信息】选项卡

(3) 图 5.4 所示为创建组的【查看】选项卡。



图 5.4 创建组的【一般信息】选项卡

(4) 图 5.5 所示为创建组的【访问】选项卡。

(5) 成功创建组后出现如图 5.6 所示界面。

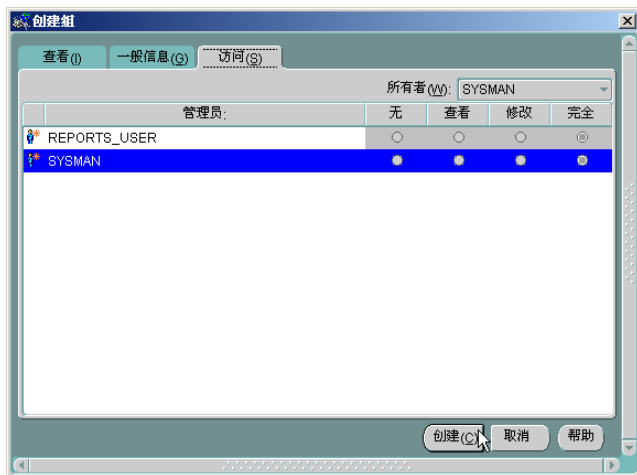


图 5.5 创建组的【访问】选项卡



图 5.6 成功创建组

(5) 在【企业管理器】的【管理目标导航树】下选择【网络】/【组】选项，创建好的组如图 5.7 所示。

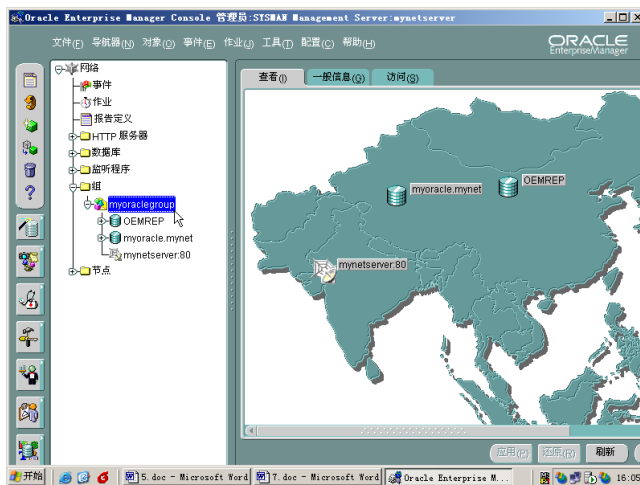


图 5.7 创建好的组

5.1.3 如何发现 OEMREP 数据库

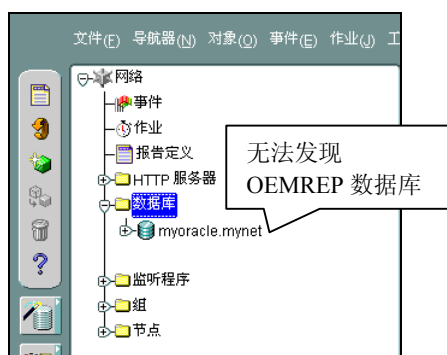


图 5.8 无法发现 OEMREP 数据库

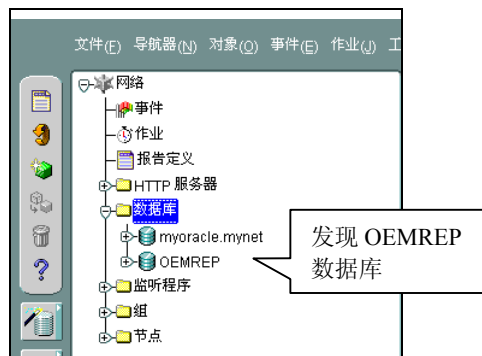


图 5.9 发现 OEMREP 数据库

(1) 在【管理目标导航器】下选择【网络】/节点选项，用鼠标右键单击，在出现的快捷菜单里选择【刷新所有节点】选项，如图 5.10 所示。

(2) 出现如图 5.11 所示的刷新向导操作界面。【管理服务器】将自动刷新节点上的信息，单击 **完成(F)** 按钮。

(3) 这样就可以发现 OEMREP 数据库了。

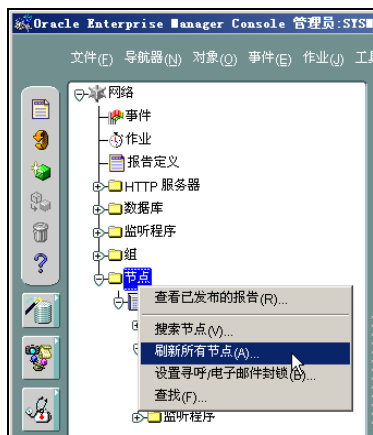


图 5.10 选择刷新所有节点

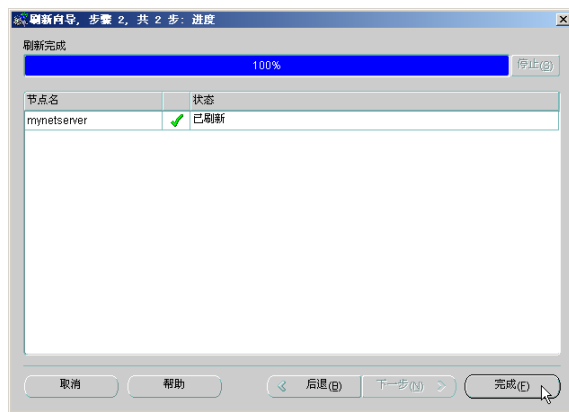


图 5.11 刷新向导操作界面

5.2 如何管理【管理服务器】的管理员

5.2.1 默认建立的管理员

【管理服务器】建立后，默认建立了两个管理员账号，如表 5.1 所示。

表 5.1 默认的管理员

账号	默认口令	权限	作用
sysman	oem_temp	超级管理员	管理员管理
reports_user	oem_temp	超级管理员	Web 报告配置

5.2.2 管理员的 3 种权限

管理员有 3 种权限，如表 5.2 所示。

表 5.2 管理员的 3 种权限

权限	作用
超级管理员	管理员管理、使用作业系统、使用事件系统
访问作业系统	使用【管理服务器】提供的作业系统
访问事件系统	使用【管理服务器】提供的事件系统

5.2.3 建立管理员

(1) 在【企业管理器】的【菜单栏】选择【配置】/【管理员管理】选项，如图 5.12 所示。

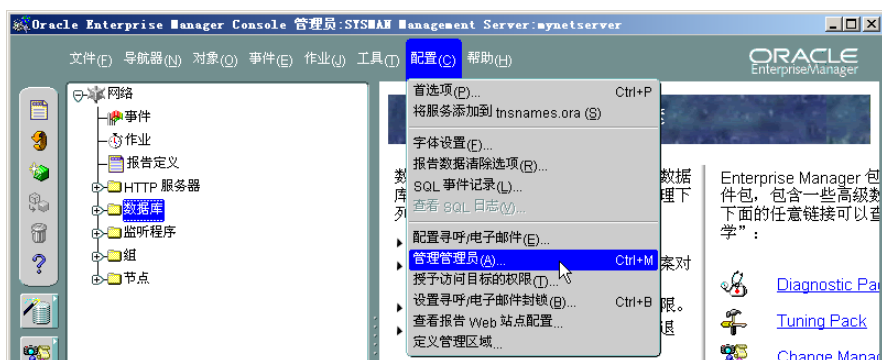


图 5.12 选择管理管理员

(2) 出现如图 5.13 所示的【管理管理员账户】界面。



图 5.13 【管理管理员账户】界面

(3) 出现如图 5.14 所示的【创建管理员账户】界面。

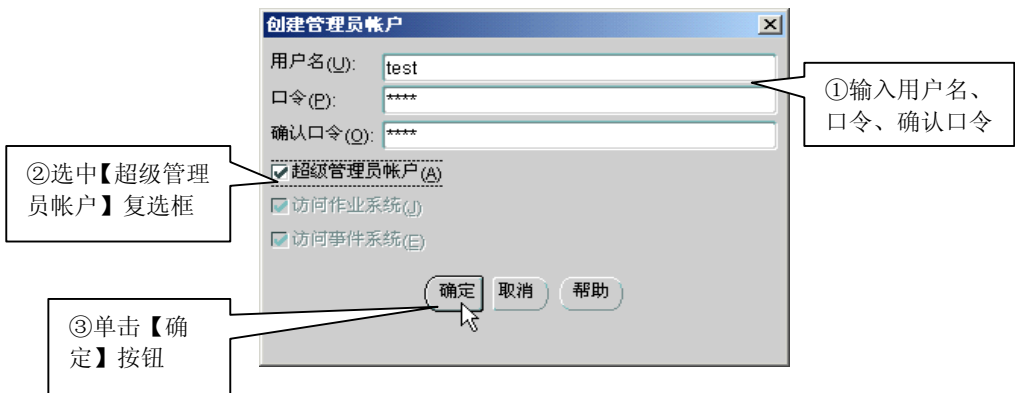


图 5.14 【创建管理员账户】界面

(4) 建立好的管理员账户如图 5.15 所示。



图 5.15 建立好的管理员账户

5.2.4 修改管理员参数

- (1) 在图 5.15 所示界面中，选中要修改的管理员账户 test，单击 **编辑(E)...** 按钮。
- (2) 出现如图 5.16 所示的编辑管理员首选项的【一般信息】选项卡。
- (3) 图 5.17 所示为编辑管理员首选项的【通知】选项卡。



图 5.16 编辑管理员首选项的【一般信息】选项卡 图 5.17 编辑管理员首选项的【通知】选项卡

(4) 图 5.18 所示为编辑管理员首选项的【调度】选项卡。

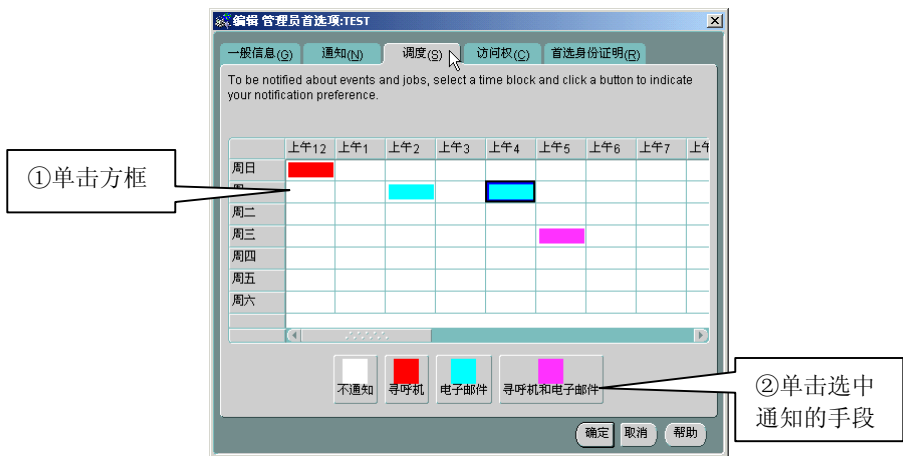


图 5.18 编辑管理员首选项的【调度】选项卡

- (5) 图 5.19 所示为编辑管理员首选项的【访问权】选项卡。
- (6) 图 5.20 所示为编辑管理员首选项的【首选身份证明】选项卡。



图 5.19 编辑管理员首选项的【访问权】选项卡 图 5.20 编辑管理员首选项的【首选身份证明】选项卡

5.2.5 删除管理员

- (1) 在图 5.15 所示界面中，选中要删除的管理员账户 test，单击 **删除(D)** 按钮。
- (2) 出现如图 5.21 所示的【管理员删除】界面，单击 **是(Y)** 按钮。

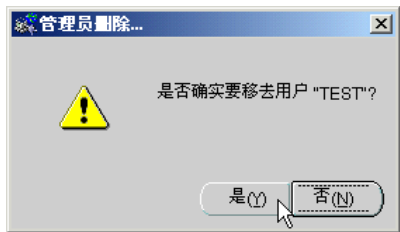


图 5.21 【管理员删除】界面

5.3 自动监控运行情况——事件

5.3.1 什么是事件

事件是在 Oracle 9i 数据库服务器上定义的、可以对服务器的资源进行监控和处理的任務。包括以下主要的内容。

5.3.2 事件的作用

- ❑ 管理员利用事件系统可以高效地监视一个大型的 Oracle 系统。
- ❑ 管理员可以并在检测到故障或特定条件时获得预警通知。
- ❑ 管理员也可以只监视希望监视的服务。
- ❑ 事件系统还可以扩展支持第三方应用程序。
- ❑ 允许大型系统的管理员根据其首选项和任务自定义事件系统。

5.3.3 什么是事件系统

事件系统是 Oracle 9i 网络环境中完成事件功能的一整套运行环境,包括以下几部分构成。

- ❑ 各数据库服务器上的【智能代理】。
- ❑ 在【管理服务器】上定义的各种事件。
- ❑ 【管理服务器】上对事件进行管理的事件库。
- ❑ 【管理服务器】上定义的修复作业。

5.3.4 事件系统什么时候发出通知

事件通知的发出将按照以下原则进行。

- ❑ 如果【事件测试】的阈值超出设置的参数值,就会发出通知。
- ❑ 如果事件没有参数,则在事件发生时发送通知。
- ❑ 如果【事件测试】的条件始终在指定阈值之上,则不会发出新通知。
- ❑ 如果下次运行测试时没有检测到此状况,则事件正常。事件正常时,也会发送通知(电子邮件/寻呼)。

5.3.5 创建事件

(1) 从【企业管理器】的【菜单栏】里选择【事件】/【创建事件】选项,如图 5.22 所示。

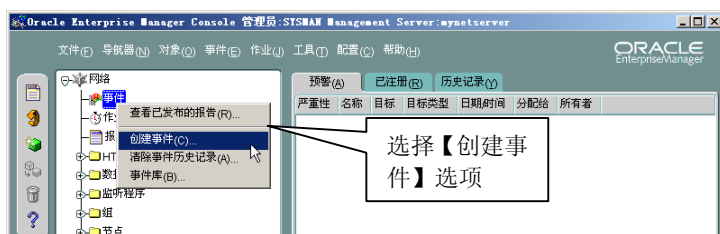


图 5.22 选择创建事件

(2) 出现如图 5.23 所示的创建事件的【一般信息】选项卡。

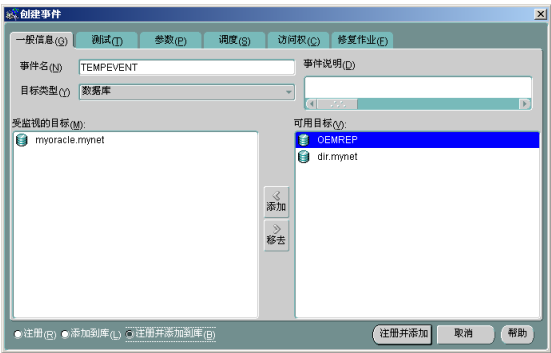


图 5.23 创建事件的【一般信息】选项卡

(3) 图 5.24 所示为创建事件的【测试】选项卡。

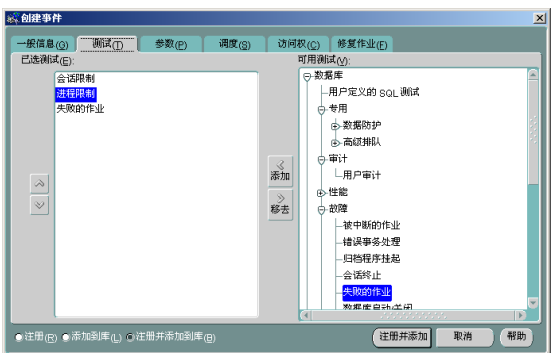


图 5.24 创建事件的【测试】选项卡

(4) 图 5.25 所示为创建事件的【参数】选项卡。



图 5.25 创建事件的【参数】选项卡

(4) 图 5.26 所示为创建事件的【调度】选项卡。

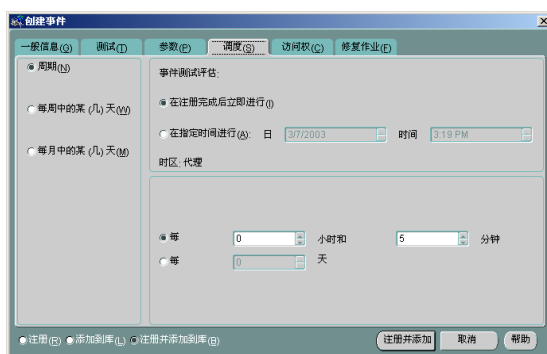


图 5.26 创建事件的【调度】选项卡

(5) 如图 5.27 所示为创建事件的【访问权】选项卡。

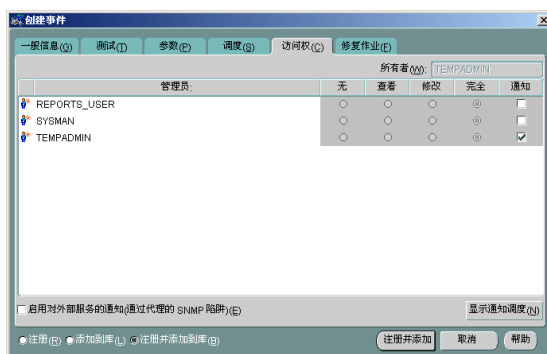


图 5.27 创建事件的【访问权】选项卡

(6) 图 5.28 所示为创建事件的【修复作业】选项卡。

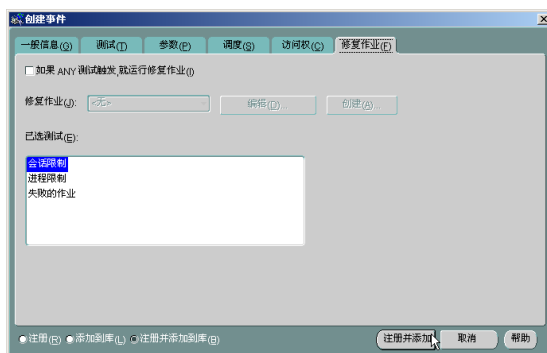


图 5.28 创建事件的【修复作业】选项卡

(7) 创建好的事件如图 5.29 所示。

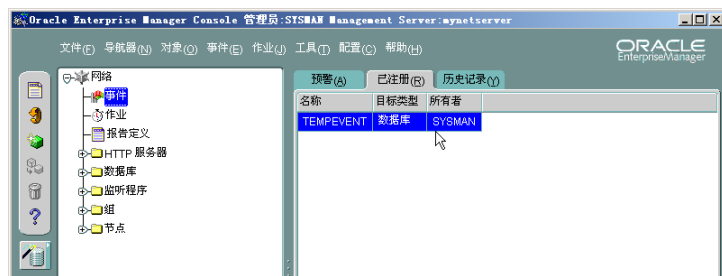


图 5.29 创建好的事件

5.4 自动运行管理任务——作业

5.4.1 什么是作业

作业是数据库服务器的一些自动化、标准化的管理工作。比如定时的数据库关闭或启动、定时执行数据库的恢复或备份等。

5.4.2 作业的作用

- ❑ 可以利用统一的“作业库”进行集中管理。
- ❑ 利用【管理服务器】提供的作业系统，既可以在本地数据库服务器上运行本地作业，也可以通过【管理服务器】来集中管理分布式数据库系统的多节点作业。
- ❑ 在管理服务器上定义的作业，通过【智能代理】分发到各个目标上去，自动执行一些重复性强、完全可以自行完成的任务。

5.4.3 什么是作业系统

- ❑ 各数据库服务器上的【智能代理】。
- ❑ 在【管理服务器】上定义的各种作业。
- ❑ 【管理服务器】上对作业进行管理的作业库。

5.4.4 创建作业

(1) 如图 5.30 所示。

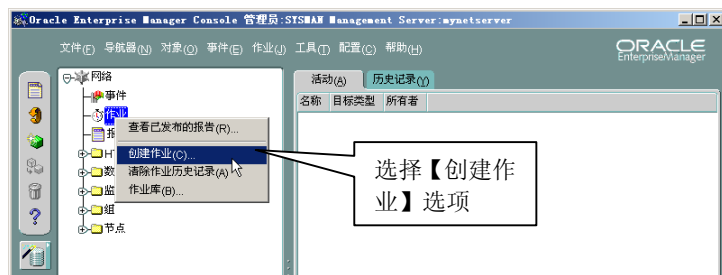


图 5.30 选择创建作业

(2) 出现如图 5.31 所示的创建作业的【一般信息】选项卡。



图 5.31 创建作业的【一般信息】选项卡

(3) 图 5.32 所示为创建作业的【任务】选项卡。

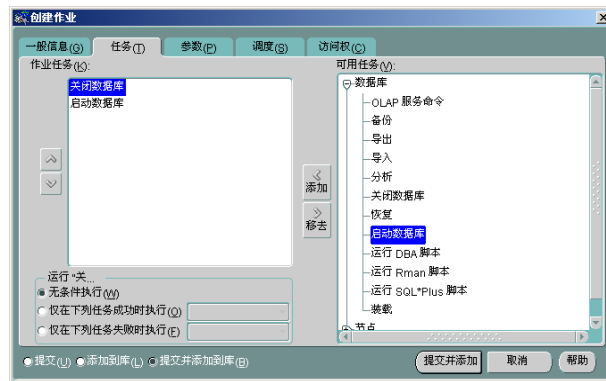


图 5.32 创建作业的【任务】选项卡

(4) 图 5.33 所示为创建作业的【参数】选项卡。

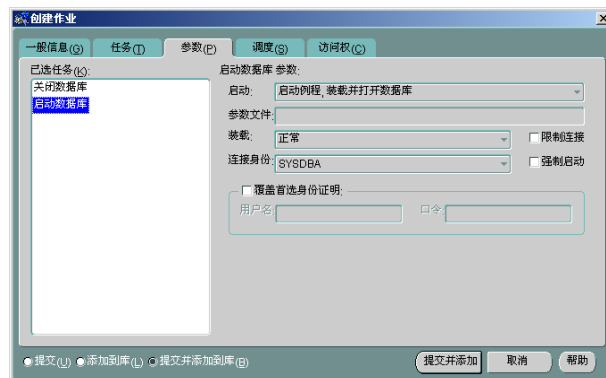


图 5.33 创建作业的【参数】选项卡

(5) 图 5.34 所示为创建作业的【调度】选项卡。

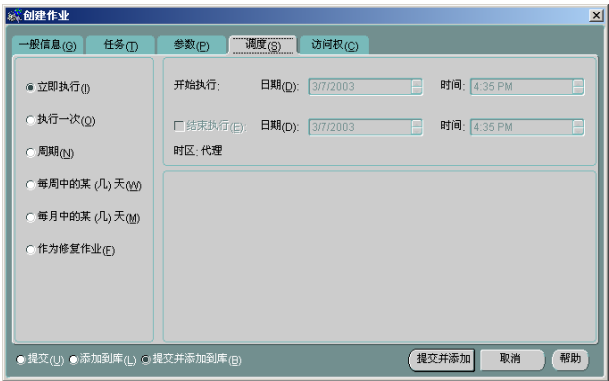


图 5.34 创建作业的【调度】选项卡

(6) 图 5.35 所示为创建作业的【访问权】选项卡。



图 5.35 创建作业的【访问权】选项卡

(7) 创建好的作业如图 5.36 所示。

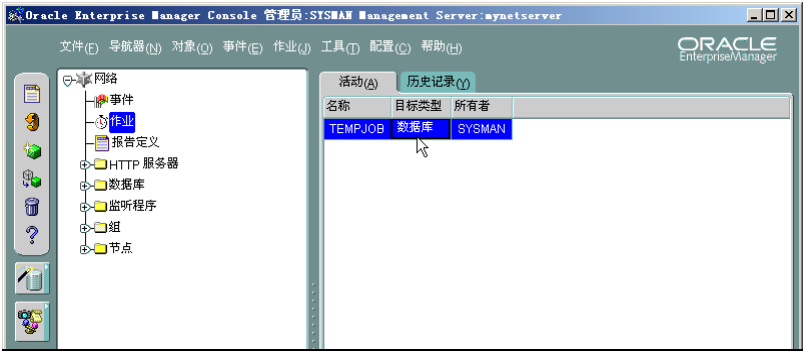


图 5.36 创建好的作业

5.4.5 作业执行失败的原因

作业建立完毕后，在作业系统的支持下，应该能够自动完成。成功完成的作业的状态如图 5.37 所示。

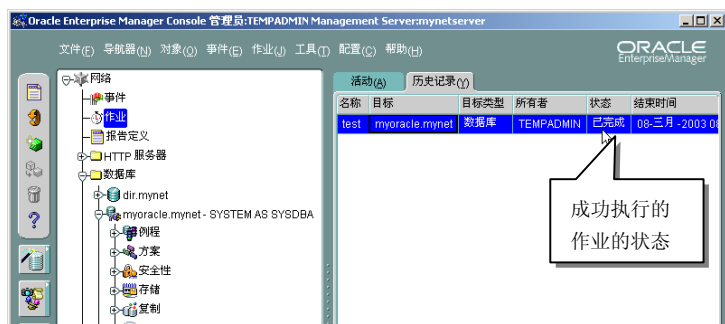


图 5.37 成功执行的作业的状态



通过上面的执行步骤的分析，如果作业执行失败，最可能的原因有 3 个。一是在作业系统中设置的首选身份证明参数有误；二是设置的节点身份在对应的服务器上没有以批处理作业登录的权限；三是设置的数据库首选身份证明对数据库资源没有相应的操作权限。

【管理服务器】的作业系统和节点的操作系统的作业系统是两个概念。

5.4.6 作业执行失败的解决办法

1. 正确设置首选身份证明

(1) 在编辑管理员首选项的【首选身份证明】选项卡中，可以在节点上赋予它这个权限，如图 5.38 所示。



图 5.38 正确设置节点的首选身份证明

(2) 针对作业执行的目标类型不一样，在【首选身份证明】选项卡中必须正确设置具有操作权限的用户及角色。



图 5.39 正确设置数据库的首选身份证明

2. 在节点上赋予用户以批处理作业登录的权限

(1) 在节点所在的服务器的桌面上选择【开始】/【程序】/【管理工具】/【本地安全策略】选项，出现如图 5.40 所示的【本地安全设置】界面。

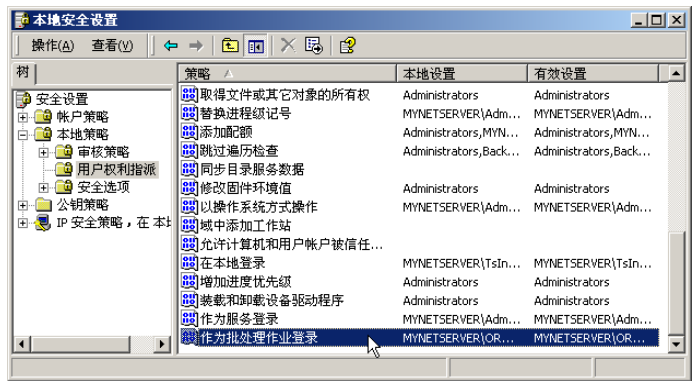


图 5.40 【本地安全设置】界面

(2) 出现如图 5.41 所示的【本地安全策略设置】界面。

(3) 出现如图 5.42 所示的【选择用户或组】界面。

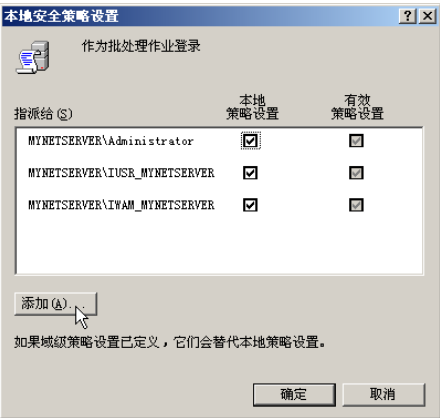


图 5.41 【本地安全策略设置】界面

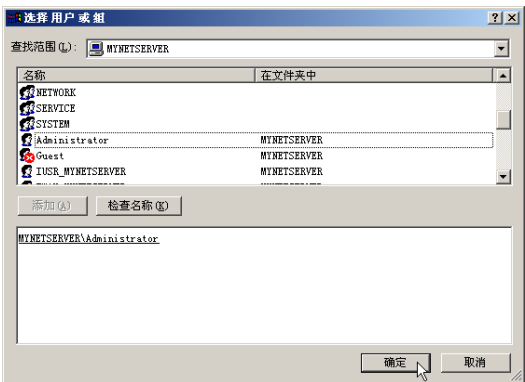


图 5.42 选择给 Administrator 赋予权限

5.5 基于 Web 的管理

5.5.1 基于 Web 管理的原理

1. 客户机

客户机操作系统为 Windows 2000、Windows 98 或 Windows NT，安装 IE 5.0 以上的浏览器，安装 Java 插件 Oracle JInitiator 1.1.8.10 和基于 Applet 技术的【企业管理器】。

2. 中间层

安装【管理服务器】和 Apache HTTP 服务器。

3. 数据库服务器

安装数据库服务器，运行【智能代理】。

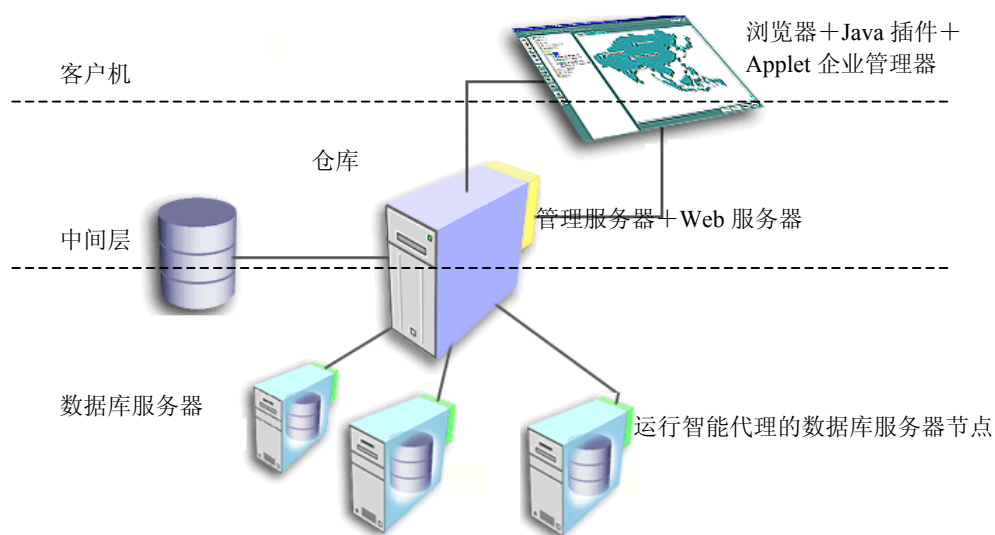


图 5.43 基于 Web 的管理原理

5.5.2 如何构建基于 Web 管理的环境

构建基于 Web 管理的环境共有 4 个步骤。

- ☐ 在数据库服务器上——启动【智能代理】。
- ☐ 启动【管理服务器】上——启动后台服务。
- ☐ 在 Web 服务器上——启动 Apache HTTP 服务器。
- ☐ 在管理客户机上——安装 Java 插件和 Applet【企业管理器】。

5.5.3 在数据库服务器上——启动【智能代理】

1. 测试【智能代理】是否启动

(1) 如图 5.44 所示。

(2) 监听程序将按照设置与数据库服务器上的【智能代理】通信，如图 5.45 所示界面。

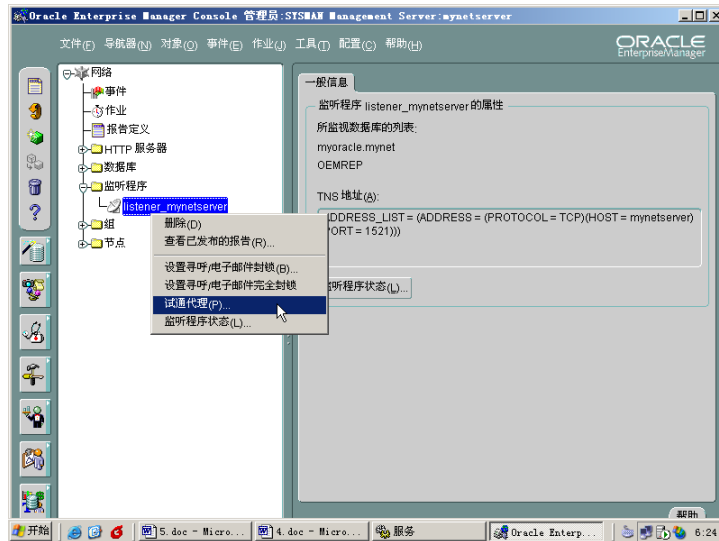


图 5.44 测试【智能代理】是否启动



图 5.45 【智能代理】成功启动

2. 【智能代理】启动不成功的解决方法

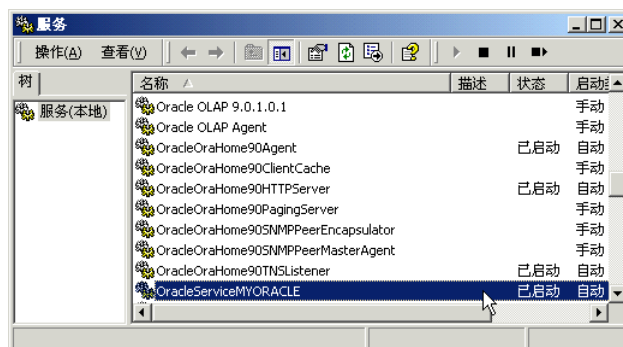


图 5.46 Oracle 9i 的服务

5.5.4 在【管理服务器】上——启动后台服务

在【管理服务器】上启动后台服务的操作参加本书 3.2.3 节的内容。

5.5.5 在 Web 服务器上——启动 Apache HTTP 服务器

1. 测试 Apache HTTP 服务器是否启用

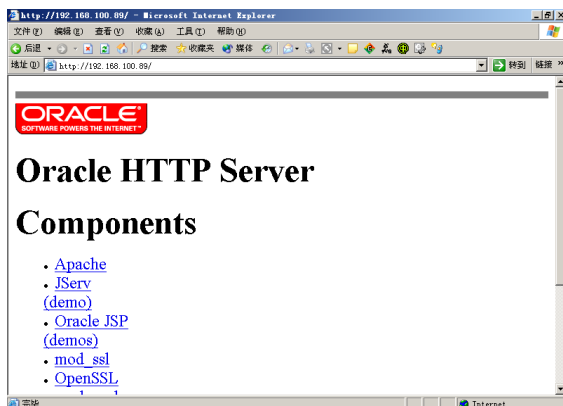


图 5.47 测试 Oracle HTTP 服务器是否正常工作

2. 如何调整 Apache HTTP 服务器的端口

- ☐ httpd.conf: 提供了最基本的服务器配置。
- ☐ access.conf: access.conf 用于配置服务器的访问权限, 控制不同用户和计算机的访问限制。
- ☐ srm.conf: srm.conf 是服务器的资源映射文件, 告诉服务器各种文件的 MIME 类型, 以及如何支持这些文件。

(1) 利用任何文本编辑软件 (如【记事本】等) 打开 httpd.conf 文件, 将“#”符号开头的注释性语句去掉后, 其中主要与端口配置有关的代码如下。

```
-----
Port 80
Listen 80
-----
```

(2) 将这两个 80 修改为你想修改的值 (如 8888), 然后重新启动 Apache 服务器, 利用浏览器访问, 发现已经修改成功。

3. 如何调整 Apache HTTP 服务器的管理端口

Oracle HTTP 服务器提供了通过 IE 进行数据库服务器管理的功能, 默认使用的端口为 3339。

在 IE 浏览器的【地址栏】输入 `http://11.122.132.188:3339/emwebsite_zh_CN.html`, 这是中文主页文件, 出现如图 5.48 所示界面, 这里就可以启动基于 Web 的各种管理功能。



图 5.48 Oracle HTTP 服务器的管理界面

(1) 本书的数据库服务器安装在 c:\oracle 目录下，因此，基于 Web 管理的网站内容存放的目录是 c:\oracle\ora90\oem_webstage。

在该目录下有一个名为 oem.conf 的文本配置文件，可以在里面修改端口设置，内容如下。

说明：配置监听器端口号（可以修改）。

Listen 3339

默认端口设置为 3339

<VirtualHost _default_:3339>

说明：服务器名（默认的值）。

ServerName mynetserver

说明：网站文件目录。

DocumentRoot "C:\oracle\ora90\oem_webstage/"

<Directory "C:\oracle\ora90\oem_webstage/">

Options Indexes FollowSymLinks

AllowOverride None

Order allow,deny

Allow from all

</Directory>

说明：网站默认主页文件。

DirectoryIndex emwebsite.html

ScriptAlias /cgi-bin/ "C:\oracle\ora90\oem_webstage/cgi-bin/"

ScriptAlias /oem_webstage/cgi-bin/ "C:\oracle\ora90\oem_webstage/cgi-bin/"

Alias /oem_webstage/ "C:\oracle\ora90\oem_webstage/"

<Directory "C:\oracle\ora90\oem_webstage/cgi-bin/">

AllowOverride all

Allow from all

</Directory>

ApJServMount /em /oemreporting

</VirtualHost>



监听器 Listen 是一类用于接收客户机信息的程序，Oracle HTTP 服务器里通过设置不同端口号的监听器来处理各种 Web 请求。

(2) 将以上的两句语句的 3339 可以修改成你需要的端口号，如修改为：

```
Listen 4000
```

```
<VirtualHost _default_:4000>
```

重新启动 Apache 服务器，将按照新的端口设置提供 Web 服务，一般情况下无需更改。

4. 如何关闭和启动 Apache HTTP 服务器

(1) 如图 5.49 所示的界面。

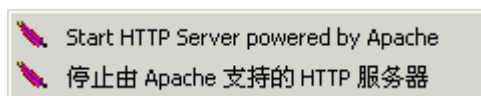


图 5.49 Oracle HTTP Server 程序组

(2) 如图 5.50 所示。

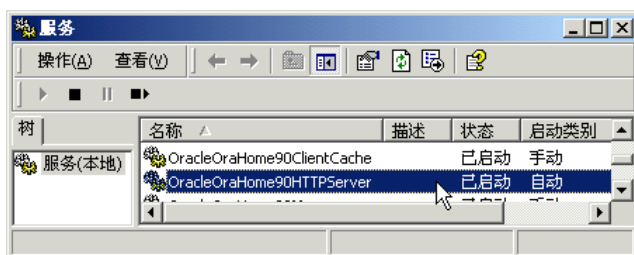


图 5.50 Oracle HTTP Server 对应的后台服务

5.5.6 在管理客户机上——安装 Java 插件和 Applet【企业管理器】

1. 下载 Oracle Jinitiator 插件

2. 安装 Oracle Jinitiator 插件

(1) 双击 jinit11810.exe 文件，出现如图 5.51 所示的【安装】界面。



图 5.51 【安装】界面

(2) 出现如图 5.52 所示的【欢迎】界面。

(3) 出现如图 5.53 所示的【安装路径】界面。

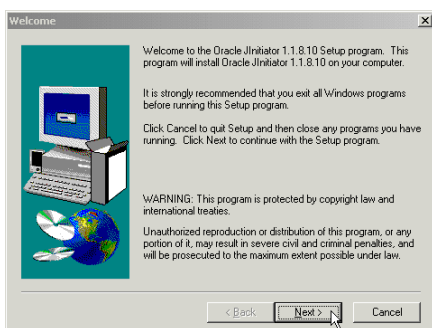


图 5.52 【欢迎】界面

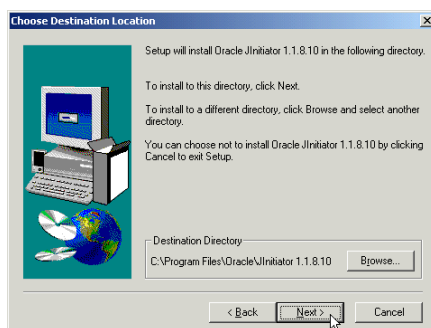


图 5.53 【安装路径】界面

(4) 出现如图 5.54 所示的【完成】界面。

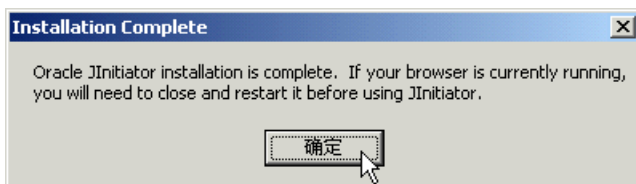


图 5.54 【完成】界面

3. 配置 Oracle JInitiator 插件

(1) 图 5.55 所示的 Jinitiator 属性的【Basic】选项卡，一般无需更改。

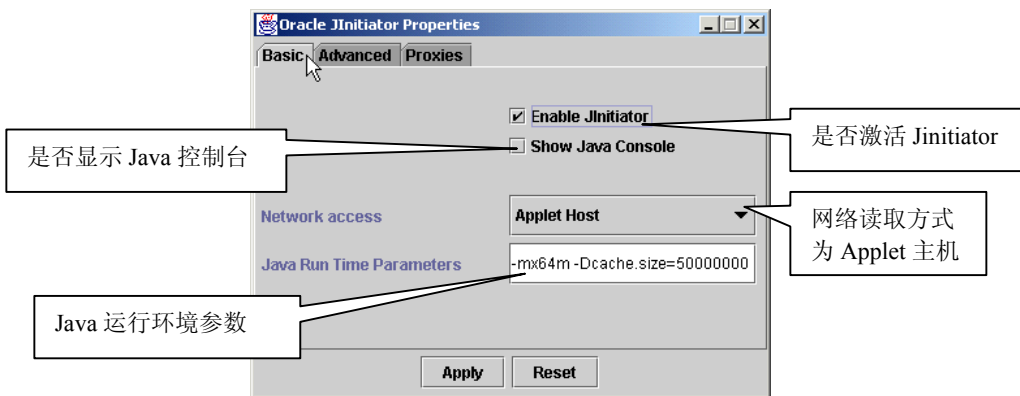


图 5.55 Oracle Jinitiator 属性的【Basic】选项卡

(2) 图 5.56 所示为 Oracle Jinitiator 属性的【Advanced】选项卡。

(3) 图 5.57 所示为 Oracle Jinitiator 属性的【Proxies】选项卡。

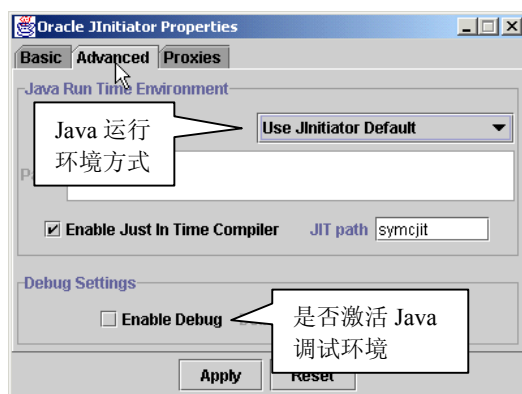


图 5.56 Oracle JInitiator 属性的【Advanced】选项卡 图 5.57 Oracle JInitiator 属性的【Proxies】选项卡

4. 安装 Applet【企业管理器】

(1) 在浏览器里输入 http://192.168.100.89:3339/emwebsite_zh_CN.html，出现如图 5.58 所示界面。

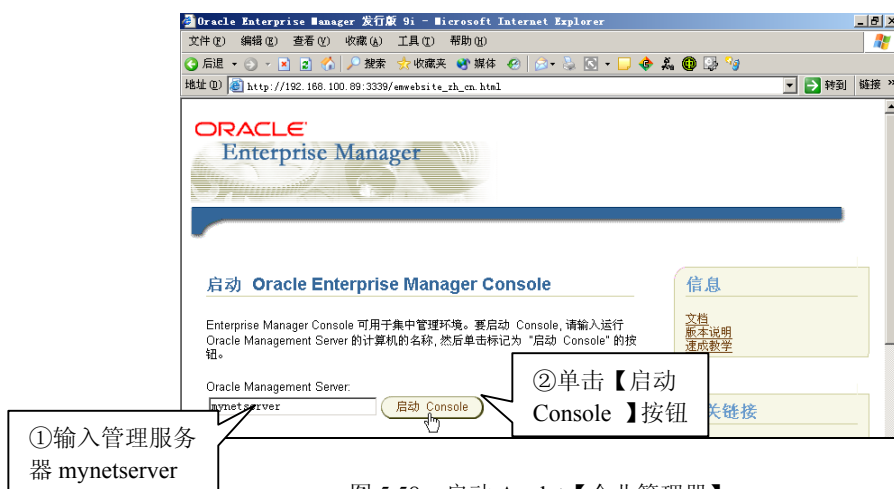


图 5.58 启动 Applet【企业管理器】

(2) 首次使用将出现如图 5.59 所示的【提示信息】界面。



图 5.59 【提示信息】界面

(3) 在浏览器的【菜单栏】选择【工具】/【Internet 选项】选项，选择如图 5.60 所示的 Internet 选项的【安全】选项卡。

(4) 出现如图 5.61 所示的 Oracle Java 控制台 Applet 界面，在整个基于 Web 方式进行管理的过程中，不要关闭图 5.61 所示的界面。



图 5.60 Internet 选项的【安全】选项卡



图 5.61 Oracle 控制台 Applet

(5) 出现如图 5.62 所示的安装 Oracle【企业管理器】组件界面。



图 5.62 安装 Oracle【企业管理器】组件界面

(6) 安装完毕后出现如图 5.63 所示的【基于 Web 的控制台登录】界面，它和集成管理环境下管理控制台的登录界面是有区别的。



图 5.63 【基于 Web 的控制台登录】界面

(8) 出现如图 5.64 所示的基于 Web 的【企业管理器】界面，和专门安装的客户管理程序的【企业管理器】相比，在【集成管理工具区】要少一类服务管理的集成工具。

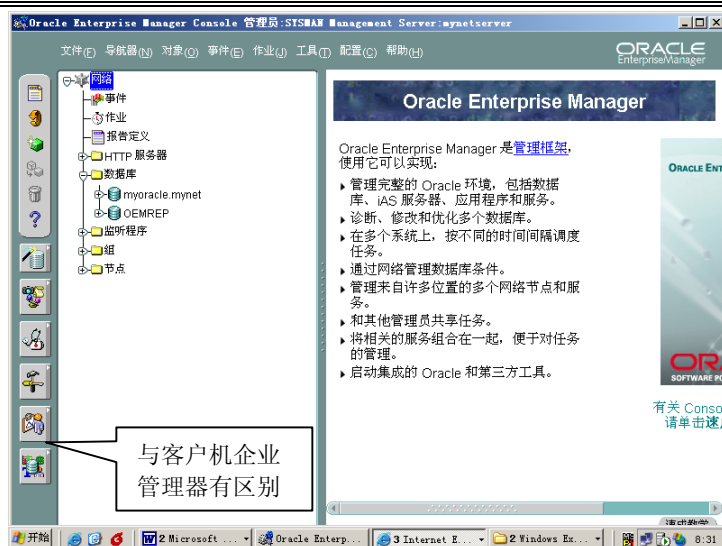


图 5.64 基于 Web 的【企业管理器】界面

5.6 通过 IE 浏览器进行监控——报告

5.6.1 什么是报告定义

【管理服务器】上收集了数据库服务器的各种运行参数。按照什么样的方法收集数据库服务器上的运行参数，称为报告定义。

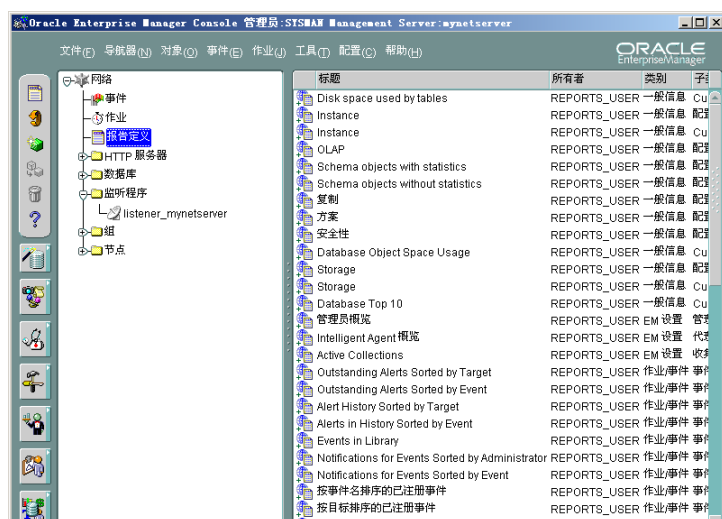


图 5.65 【管理服务器】上的报告定义

5.6.2 什么是报告

报告就是报告定义适用于具体的目标（数据库服务器）时产生的有关服务器的运行参数。

5.6.3 如何构建报告的 Web 发布环境

(1) 默认管理员账户为 “reports_user”，参见表 5.1 的相关内容。

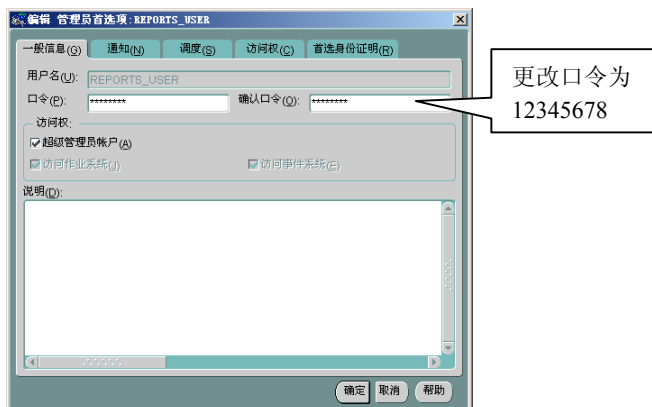


图 5.66 编辑管理员首选项的【一般信息】选项卡

(2) 进入 DOS 方式，在 c:\oracle\ora90\bin 目录下有名为 oemctl.bat 的批处理文件，这就是用于配置 Web 发布环境的批处理文件。

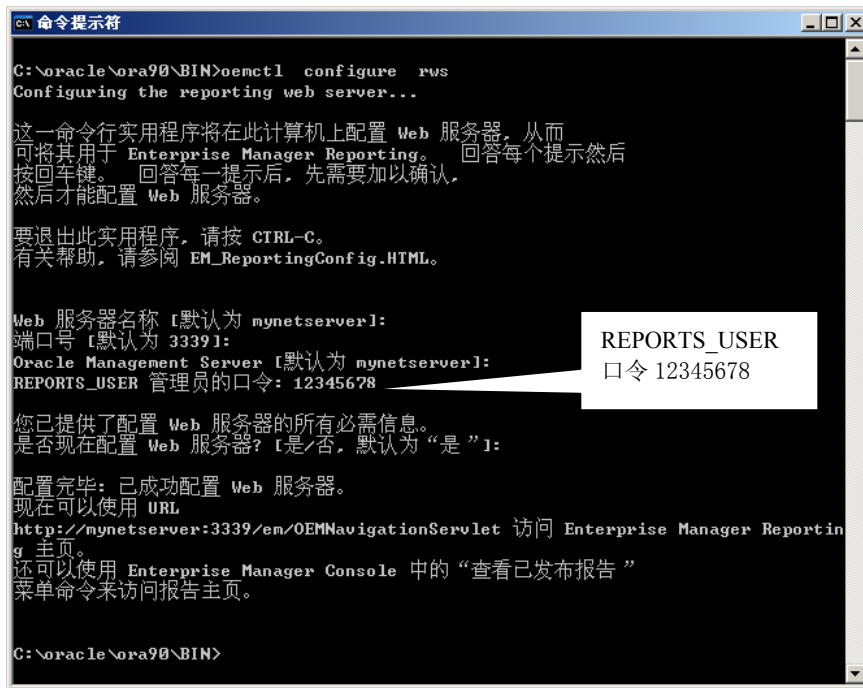


图 5.67 基于 DOS 命令行的配置

5.6.4 如何通过 IE 浏览器浏览报告

(1) 在浏览器的【地址栏】输入 `http://192.168.100.89:3339/emwebsite_zh_CN.html`，出现如图 5.68 所示界面。

(2) 出现如图 5.69 所示的界面。

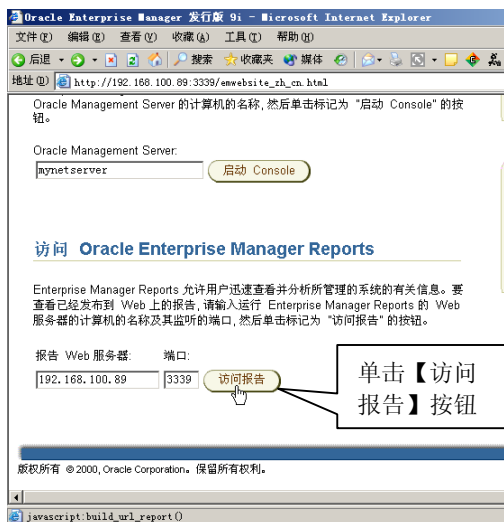


图 5.68 选择访问报告



图 5.69 通过浏览器浏览报告

5.7 习题

- (1) 节点和组是什么含义？有什么样的关系？
- (2) 【管理服务器】的管理员有哪 3 种权限？
- (3) 默认的管理员账户及口令是什么？
- (4) 如何创建管理员？如何修改管理员的口令？
- (5) 什么是事件？事件有什么作用？
- (6) 如何构建事件系统环境？
- (7) 什么是作业？作业有什么作用？
- (8) 如何构建作业系统环境？
- (9) 作业执行失败可能的原因和解决方法是什么？
- (10) 如何构建基于 Web 管理的环境？
- (11) 如何构建报告的 Web 发布环境？

第 6 章 运行维护——DBA 的日常工作

本章通过实例的方法详细介绍了作为 DBA 的日常维护操作。

6.1 确保 Oracle 9i 网络的正常运行

一个完整的 Oracle 9i 的运行网络，包括以下 4 个部分。

- ☐ 数据库服务器
- ☐ 【管理服务器】
- ☐ Oracle HTTP 服务器
- ☐ 管理客户机

6.1.1 确保数据库服务器的正常运行

要确保数据库服务器正常运行，就是要确保数据库服务器的后台服务正常启动。计算机上所有的服务列表，如图 6.1 所示。

其中与数据库服务器有关的服务包括。

- ☐ OracleOraHome90Agent: 【智能代理】的后台服务。
- ☐ OracleOraHome90TNSListener: 监听程序对应的后台服务。
- ☐ OracleServiceMYORACLE: 【全局数据库名】为 myoracle.mynet 的数据库的服务。
- ☐ OracleServiceOEMREP: 【全局数据库名】为 oemrep 的数据库的服务。

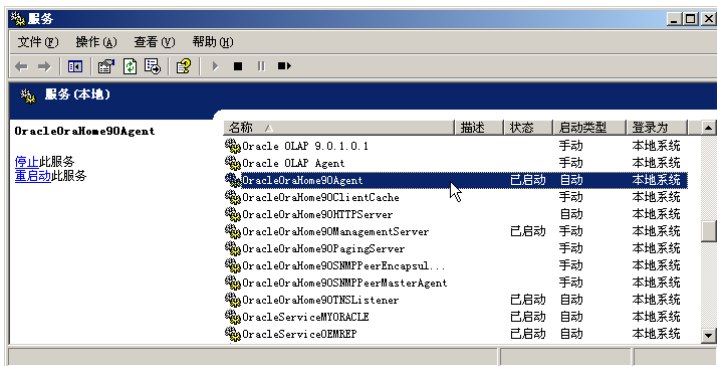


图 6.1 计算机的后台服务列表

6.1.2 确保【管理服务器】的正常运行

要确保【管理服务器】正常运行，就是要确保【管理服务器】对应的后台服务正常启动。在图 6.1 中与【管理服务器】有关的后台服务包括。

- ❑ OracleOraHome90ManagementServer: 【管理服务器】的后台服务。
- ❑ OracleServiceOEMREP : 【管理服务器】需要使用的后台数据库。

6.1.3 确保 Oracle HTTP 服务器的正常运行

要确保 Oracle HTTP 服务器的正常运行，就是要确保 Oracle HTTP 服务器对应的后台服务正常启动。在图 6.1 中与【管理服务器】有关的后台服务为 OracleOraHome90HTTPServer。

6.1.4 确保管理客户机的正常运行

要确保管理客户机的正常运行，主要有两点。

- ❑ 确保通信网络连接通畅，请参见本书 2.1.5 节。
- ❑ 确保【网络服务名】配置无误，请参见本书 2.3.2 节。

6.2 初始化参数文件

6.2.1 Pfile 文件

Pfile (Parameter File) 文件是基于文本格式的参数文件，含有数据库的配置参数。

Oracle 9i 在安装时为每个数据库建立了一个 Pfile，默认的名称为“init+例程名.ora”，这是一个文本文件，可以用任何文本编辑工具打开。

表 6.1 数据库的初始化参数文件分析

内容	说明
# Copyright (c) 1991, 2001 by Oracle Corporation	Oracle 公司版权标识
# MTS	多线程服务器配置标识，在 Oracle 9i 里称为共享服务器配置
dispatchers="(PROTOCOL=TCP)(SER=MODESE)", "(PROTOCOL=TCP)(PRE=oracle.aurora.server.GiopServer)", "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)"	多线程服务器配置
# 其他	配置其他参数
compatible=9.0.0	兼容版本 9.0.0
db_name=myoracle	数据库名称为 myoracle
# 分布式, 复制和快照	配置分布式、复制和快照参数
db_domain=mynet	数据库域名为 mynet，加上数据库名称 db_name 构成全局数据库名称

续表

内容	说明
remote_login_passwordfile=EXCLUSIVE	指定操作系统或口令文件是否具有检查用户口令的权限。设置为 EXCLUSIVE, 将使用数据库的口令文件对每个具有权限的用户进行验证。
# 排序, 散列联接, 位图索引	配置排序、散列联接、位图索引参数
sort_area_size=524288	指定排序区使用的最大内存量为 512KB。排序完成后, 各行将返回, 并且内存将释放。增大该值可以提高大型排序的效率。
# 文件配置	文件配置参数
control_files=("C:\oracle\oradata\myoracle\CONTROL01.CTL", "C:\oracle\oradata\myoracle\CONTROL02.CTL", "C:\oracle\oradata\myoracle\CONTROL03.CTL")	指定控制文件的路径及文件名
# 池	内存配置参数
Java_pool_size=33554432	指定 Java 存储池的大小为 32MB, 用于存储 Java 的方法、类定义和 Java 对象。
large_pool_size=1048576	指定大型池的大小为 1MB, 用于共享服务器的会话内存、并行执行的消息缓冲区以及 RMAN 备份和恢复的磁盘 I/O 缓冲区。
shared_pool_size=33554432	指定共享池的大小为 32MB, 用于存储共享游标、存储的过程、控制结构和并行执行消息缓冲区等对象。较大的值能改善多用户系统的性能
# 游标和库高速缓存	配置游标和高速缓存参数
open_cursors=300	指定一个会话一次可以打开的游标的最大数量为 300, 应将该值设置得足够高, 这样才能防止应用程序耗尽打开的游标
# 系统管理的撤消和回退段	配置系统管理撤消和回滚段参数
undo_management=AUTO	指定系统使用的撤消空间管理方式为 SMU 方式, 在 SMU 方式下, 撤消空间会像撤消表空间一样在外部分配
undo_tablespace=UNDOTBS	指定回滚表空间为 UNDOTBS
# 网络注册	配置网络注册参数
instance_name=myoracle	例程名称为 myoracle
# 诊断和统计	配置诊断和统计参数
background_dump_dest=C:\oracle\admin\myoracle\bdump	后台进程跟踪文件目录
core_dump_dest=C:\oracle\admin\myoracle\cdump	核心转储跟踪文件目录

续表

内容	说明
timed_statistics=TRUE	收集操作系统的计时信息，这些信息可被用来优化数据库和 SQL 语句
user_dump_dest=C:\oracle\admin\myoracle\udump	用户进程跟踪文件目录
# 进程和会话	配置进程和会话信息
processes=150	指定可同时连接到一个 Oracle Server 上的操作系统用户进程的最大数量为 150
# 重做日志和恢复	重做日志和恢复参数设置
Fast_start_mttr_target=300	指定从单个数据库例程崩溃中恢复所需的时间为 300 秒
# 高速缓存和 I/O	配置高速缓存和 I/O 参数
db_block_size=4096	指定数据块大小为 4KB
db_cache_size=33554432	指定数据缓冲区为 32MB，该值越大，可以减少对数据库文件的 I/O 次数，提高效率

6.2.2 SPfile 文件

SPfile (Server Parameter File, 服务器参数文件) 是基于二进制格式的参数文件，含有数据库及例程的参数和数值，但不能用文本编辑工具打开。

下面对两种初始化参数文件进行比较如表 6.2 所示。

表 6.2 SPfile 和 Pfile 文件的比较

比较内容	SPfile	Pfile
格式	二进制格式	文本格式
编辑方式	(1) 利用企业管理器对 Pfile 进行修改，然后转换为 Spfile (2) 在 SQL Plus 里使用 ALTER SYSTEM 语句进行修改	(1) 利用文本工具直接进行修改 (2) 在企业管理器里修改配置后导出形成
默认名称	SPfile+例程名.ora	Init+例程名.ora 实际参数文件 Init.ora
默认路径	Oracle\ora90\database\	Oracle\ora90\database\ Init.ora 位于 Oracle\admin\数据库例程名\pfile\
启动次序	SPfile 优先于 Pfile	Pfile 低于 Spfile

6.3 数据库的配置参数

如图 6.2 所示。

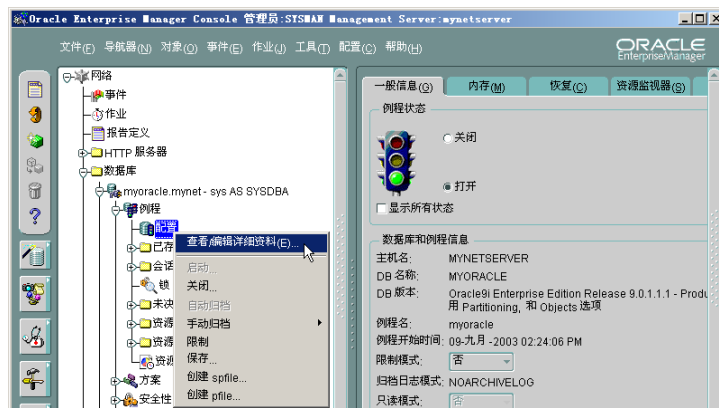


图 6.2 选择查看数据库配置参数

6.3.1 【一般信息】选项卡的参数

出现如图 6.3 所示的编辑数据库配置的【一般信息】选项卡。



图 6.3 编辑数据库配置的【一般信息】选项卡

6.3.2 【内存】选项卡的参数

图 6.4 所示为编辑数据库配置的【内存】选项卡。

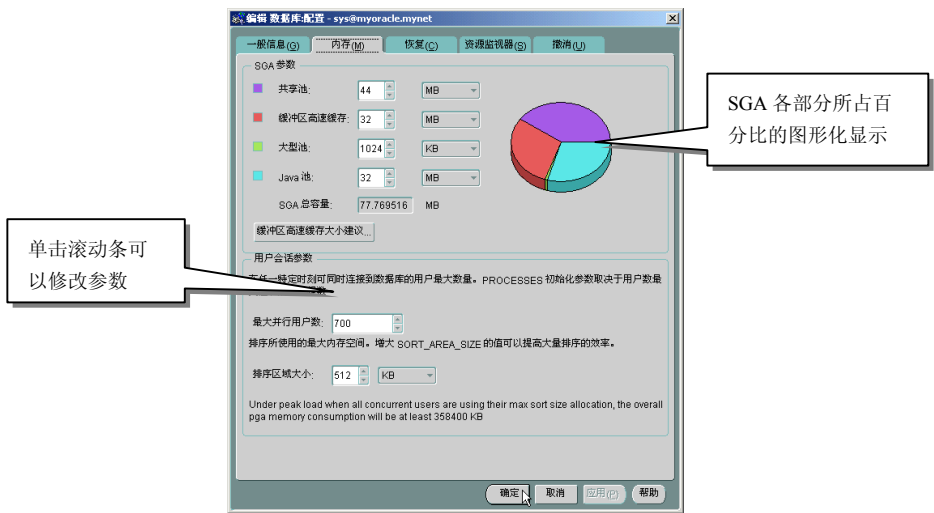


图 6.4 编辑数据库配置的【内存】选项卡

6.3.3 【恢复】选项卡的参数

图 6.5 所示为编辑数据库配置的【恢复】选项卡。



图 6.5 编辑数据库配置的【恢复】选项卡

6.3.4 【撤消】选项卡的参数

图 6.6 所示为编辑数据库配置的【撤消】选项卡。

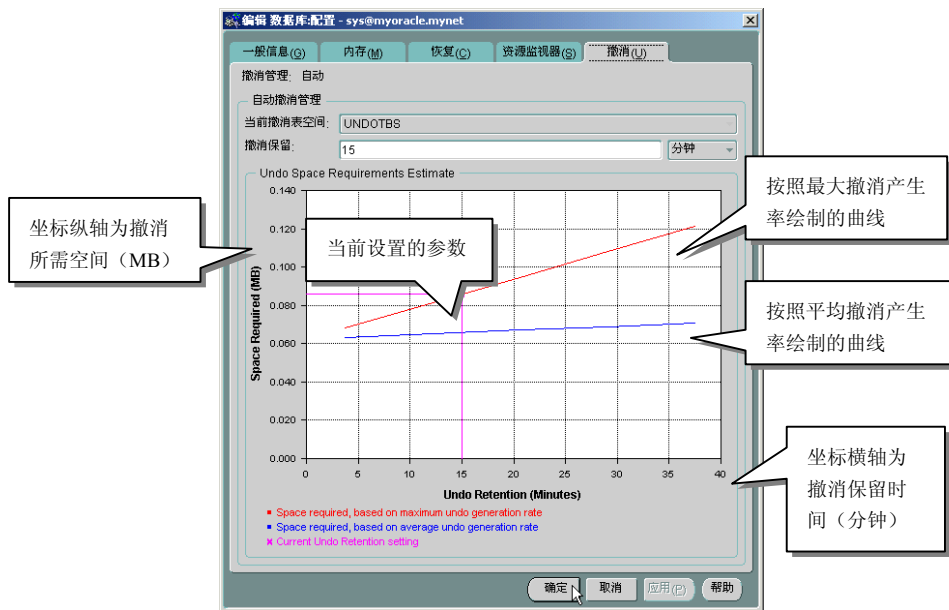


图 6.6 编辑数据库配置的【撤销】选项卡

6.4 切换数据库的日志模式

6.4.1 数据库的两种日志模式

Oracle 数据库有两种日志工作模式。

- ☐ 非归档日志模式
- ☐ 归档日志模式

6.4.2 切换到归档日志模式

(1) 在如图 6.7 所示的编辑数据库配置的【一般信息】选项卡中可以查看数据库的工作状态，默认情况下为“非归档模式”(NOARCHIVELOG)。

(2) 切换到如图 6.8 所示的编辑数据库配置的【恢复】选项卡。

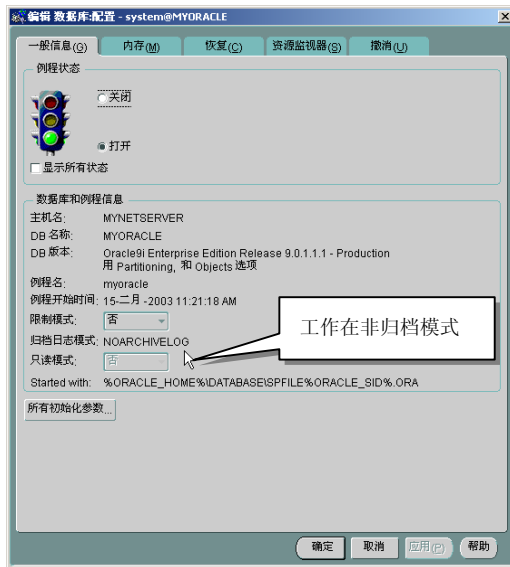


图 6.7 编辑数据库配置的【一般信息】选项卡

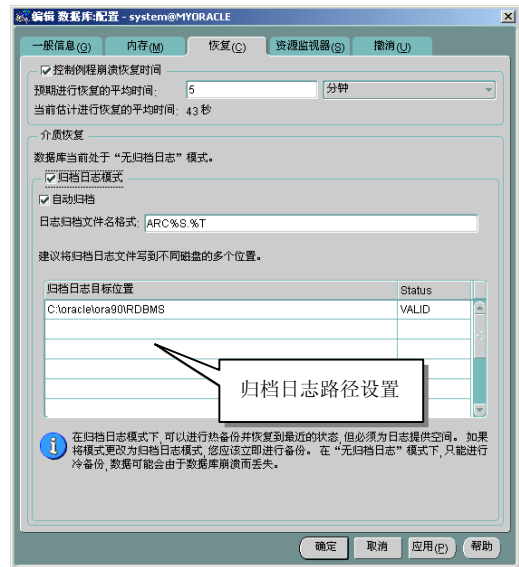


图 6.8 编辑数据库配置的【恢复】选项卡

(3) 出现如图 6.9 所示的【数据库颤动】界面。

(4) 切换到如图 6.10 所示的编辑数据库配置的【一般信息】选项卡。



图 6.9 【数据库颤动】界面

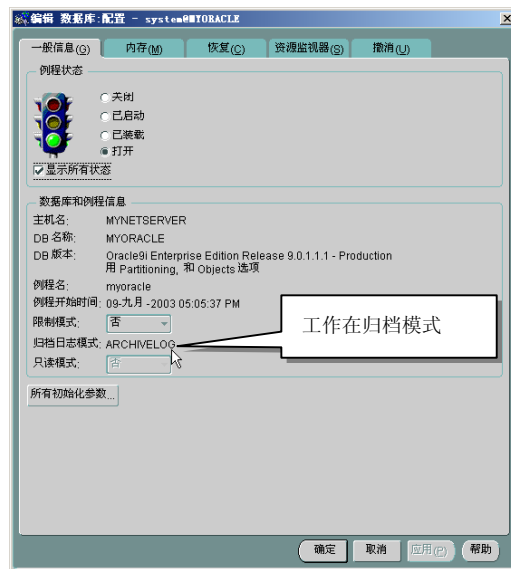


图 6.10 编辑数据库配置的【一般信息】选项卡

6.5 查询使用数据库的用户

6.5.1 什么是会话

会话 (Session) 是 Oracle 数据库服务器对连接数据库的用户进行记录的一种手段。

6.5.2 会话有什么内容

(1) 如图 6.11 所示的会话界面。如果数据库会话很多，显示的将是使用数据库例程资源的前几个会话。

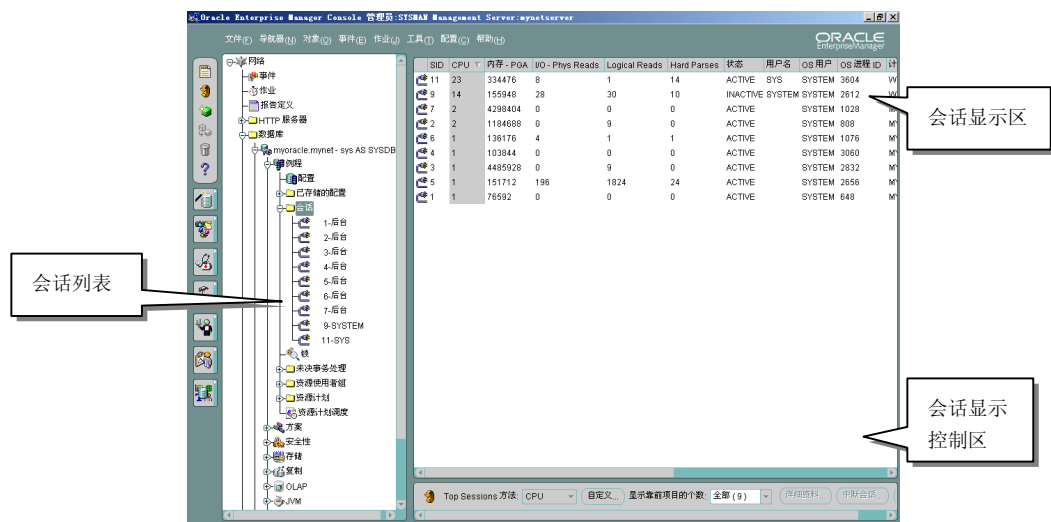


图 6.11 查询数据库的会话

(2) 在【会话显示控制区】里可以设置显示会话的方法。

(3) 单击【自定义】按钮按照任何所选的会话统计信息排序。

(4) 在【会话显示区】下选择会话【11】单击鼠标右键，在出现的快捷菜单里选择【查看/编辑详细资料】菜单选项，如图 6.12 所示。

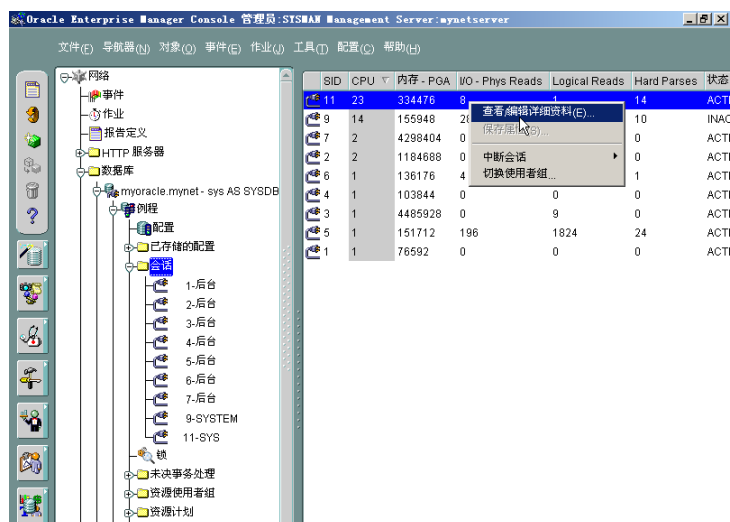


图 6.12 选择查看会话信息

(5) 出现如图 6.13 所示的编辑会话的【一般信息】选项卡。

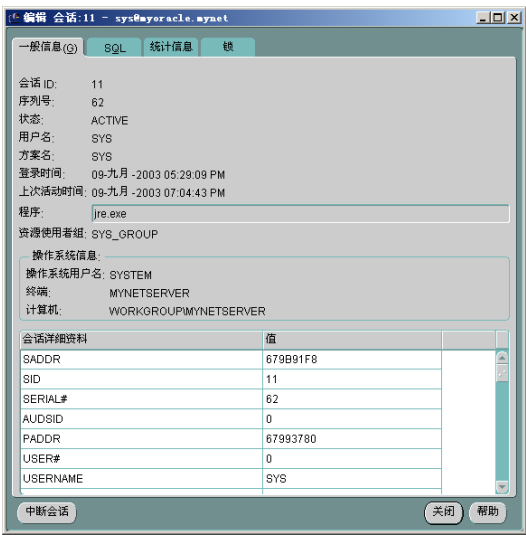


图 6.13 编辑会话的【一般信息】选项卡

主要的信息如表 6.3 所示。

表 6.3 会话的主要信息

名称	含义
SID	会话标识符
CPU	会话所使用的 CPU 资源量
PGA 内存	会话使用的 PGA 内存大小
I/O 物理读取数	会话期间读取的 I/O 物理块数
逻辑读取数	会话期间读取的数据块数，包括从内存和磁盘读取的块数
状态	表示用户会话处于活动状态还是非活动状态
用户名	与会话相关联的 Oracle 用户 ID
OS 用户	操作系统用户名
OS 进程 ID	操作系统进程标识号
计算机名	操作系统计算机，用户通过它实现连接
程序	当前正在运行的程序

(6) 图 6.14 所示为编辑会话的【SQL】选项卡。



图 6.14 编辑会话的【SQL】选项卡

(7) 图 6.15 所示为编辑会话的【统计信息】选项卡。

(8) 图 6.16 所示为编辑会话的【锁】选项卡。

统计信息	值	类别
CPU used by this session	41	用户
CPU used when call started	41	调试
CR blocks created	0	高速缓存
Cached Commit SCN referenced	0	调试
Commit SCN cached	0	调试
DBWR buffers scanned	0	高速缓存
DBWR checkpoint buffers written	0	高速缓存
DBWR checkpoints	0	高速缓存
DBWR cross instance writes	0	全局高速缓存
DBWR free buffers found	0	高速缓存
DBWR fusion writes	0	全局高速缓存
DBWR lru scans	0	高速缓存
DBWR make free requests	0	高速缓存
DBWR revisited being-written buffer	0	高速缓存
DBWR summed scan depth	0	高速缓存
DBWR transaction table writes	0	高速缓存
DBWR undo block writes	0	高速缓存
DOL statements parallelized	0	集群数据库
DFO trees parallelized	0	集群数据库
DML statements parallelized	0	集群数据库
PX Inital messages rcv'd	0	集群数据库

图 6.15 编辑会话的【统计信息】选项卡

用户名	SID	锁类型	占用的模式	请求的模式	对象名	限制的 ROWID	对象所有者	对象类型
-----	-----	-----	-------	-------	-----	-----------	-------	------

图 6.16 编辑会话的【锁】选项卡

6.6 创建数据库

6.6.1 用【数据库配置助手】创建数据库

(1) 出现如图 6.18 所示的【欢迎使用】界面。



图 6.18 【欢迎使用】界面

(2) 出现如图 6.19 所示的【操作】界面。

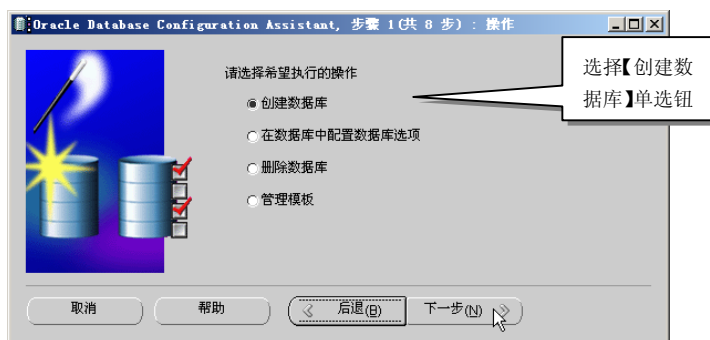


图 6.19 【操作】界面

(3) 出现如图 6.20 所示的【数据库模板】界面。



图 6.20 【数据库模板】界面

(4) 出现如图 6.21 所示的【数据库标识】界面。

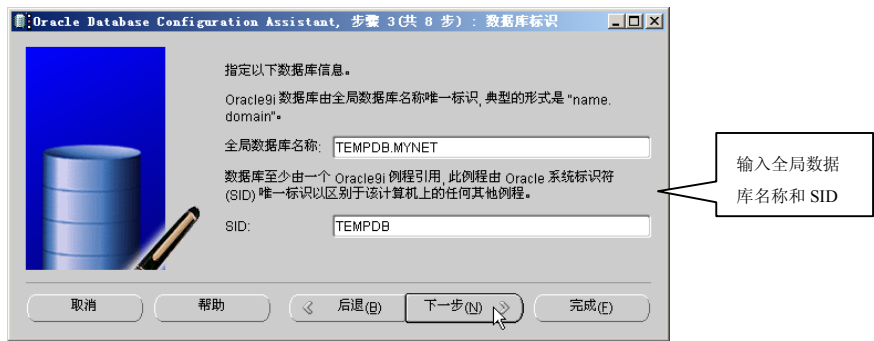


图 6.21 【数据库标识】界面

(5) 出现如图 6.22 所示的【数据库连接选项】界面。

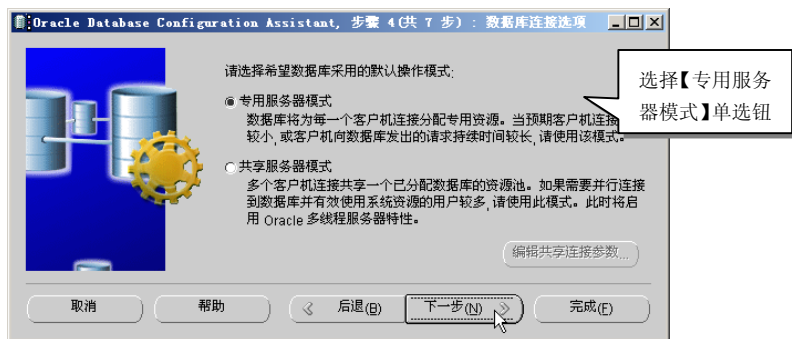


图 6.22 【数据库连接选项】界面

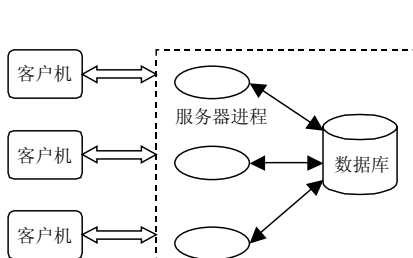


图 6.23 专用服务器工作模式原理

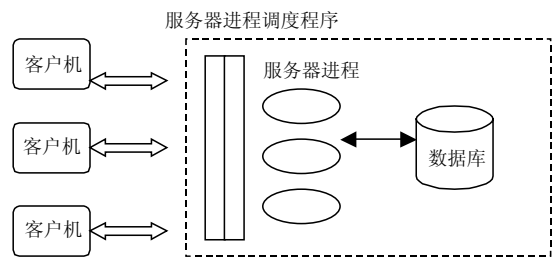


图 6.24 共享服务器工作模式原理

(6) 出现如图 6.25 所示的初始化参数的【内存】选项卡。



图 6.25 初始化参数的【内存】选项卡

表 6.4 【典型】内存配置参数

参数	含义
【最大并发连接用户数】文本框	设置要在任意给定时间并发（同时）连接到数据库的大概用户数
【用于 Oracle 的物理内存的百分比】文本框	输入可分配给数据库的全部物理内存的百分比
【数据库类型】下拉列表框	<p>选择数据库的大致用途，这样将影响初始化参数 DB_CACHE_SIZE（数据块大小）、PROCESSES（进程数）、SHARED_POOL_SIZE（共享池）和回退表空间信息的设置。</p> <p>包括联机事务处理（OLTP）、多用途（默认选项）和数据仓库共 3 种，默认为多用途。</p>

(7) 图 6.26 所示为初始化参数的【归档】选项卡。



图 6.26 初始化参数的【归档】选项卡

(8) 图 6.27 所示为初始化参数的【数据库大小】选项卡。



图 6.27 初始化参数的【数据库大小】选项卡

(9) 图 6.28 所示为初始化参数的【文件位置】选项卡。



图 6.28 初始化参数的【文件位置】选项卡

(10) 出现如图 6.29 所示的【数据库存储】界面。

(11) 出现如图 6.30 所示的【创建选项】界面。



图 6.29 【数据库存储】界面



图 6.30 【创建选项】界面

(12) 出现如图 6.31 所示的【概要】界面。

(13) 出现如图 6.32 所示的【创建过程】界面。



图 6.31 【概要】界面



图 6.32 【创建过程】界面

(14) 出现如图 6.33 所示的【成功创建数据库】界面。

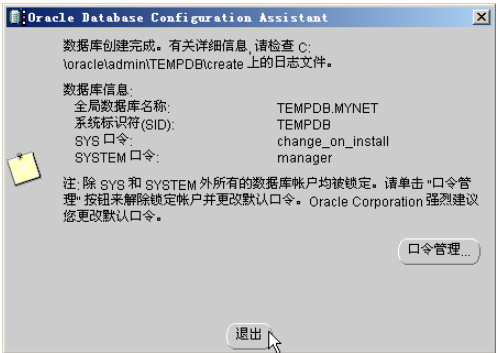


图 6.33 【成功创建数据库】界面

6.6.2 创建的数据库的参数

打开 C:\ORACLE\ADMIN\TEMPDB\PFIL\INIT.ORA 文件，可以查看最重要的初始化参数的设置如下。

【参见配套文件】：\第 6 章\int.ora。

```
# Copyright (c) 1991, 2001 by Oracle Corporation
# MTS
dispatchers="(PROTOCOL=TCP) (SER=MODESE)",
"(PROTOCOL=TCP) (PRE=oracle.aurora.server.GiopServer)",
"(PROTOCOL=TCP) (PRE=oracle.aurora.server.SGiopServer)"
# 其他
compatible=9.0.0
```

```
db_name=TEMPDB
# 分布式, 复制和快照
db_domain=MYPNET
remote_login_passwordfile=EXCLUSIVE
# 归档
log_archive_dest_1='LOCATION=C:\oracle\oradata\TEMPDB\archive'
log_archive_format=%t_%.dbf
log_archive_start=true
# 排序, 散列联接, 位图索引
sort_area_size=524288
# 文件配置
control_files=('C:\oracle\oradata\TEMPDB\CONTROL01.CTL',
"C:\oracle\oradata\TEMPDB\CONTROL02.CTL", "C:\oracle\oradata\TEMPDB\CONTROL03.CTL")
# 池
java_pool_size=52428800
large_pool_size=10485760
shared_pool_size=63868928
# 游标和库高速缓存
open_cursors=300
# 系统管理的撤消和回退段
undo_management=AUTO
undo_tablespace=UNDOTBS
# 网络注册
instance_name=TEMPDB
# 诊断和统计
background_dump_dest=C:\oracle\admin\TEMPDB\bdump
core_dump_dest=C:\oracle\admin\TEMPDB\cdump
timed_statistics=TRUE
user_dump_dest=C:\oracle\admin\TEMPDB\udump
# 进程和会话
processes=150
# 重做日志和恢复
fast_start_mttr_target=300
# 高速缓存和 I/O
db_block_size=4096
db_cache_size=86749184
-----
```

6.7 修改数据库的工作模式

- (1) 启动【数据库配置助手】，一直到出现如图 6.34 所示的【操作】界面。
- (2) 出现如图 6.35 所示的【数据库】界面。

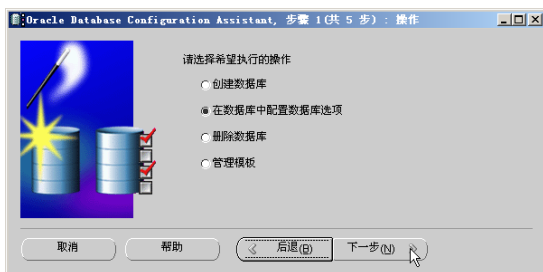


图 6.34 【操作】界面

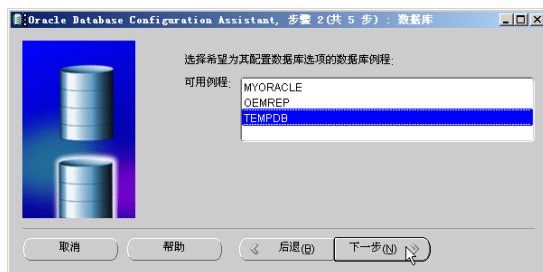


图 6.35 【数据库】界面

- (3) 出现如图 6.36 所示的数据库选项的【数据库特性】选项卡。
- (4) 出现如图 6.37 所示的【数据库连接选项】界面。



图 6.36 数据库选项的【数据库特性】选项卡



图 6.37 【数据库连接选项】界面

- (5) 出现如图 6.38 所示的共享服务器模式的【基本】选项卡。
- (6) 切换到如图 6.39 所示的共享服务器模式的【高级】选项卡。

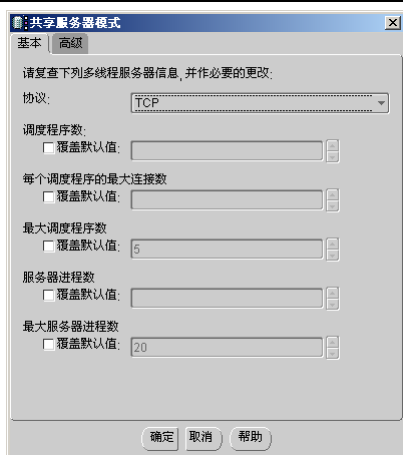


图 6.38 共享服务器模式的【基本】选项卡 图 6.39 共享服务器模式的【高级】选项卡

(7) 返回图 6.37 所示界面。

(8) 出现如图 6.40 所示的【确定初始化文件的位置】界面。

(9) 出现如图 6.41 所示的【概要】界面。

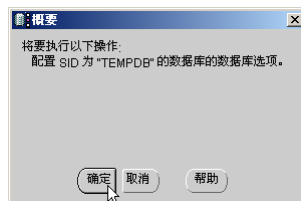


图 6.40 【确定初始化文件的位置】界面

图 6.41 【概要】界面

(10) 出现如图 6.42 所示的【配置过程】界面。

(11) 配置完毕，出现如图 6.43 所示的【配置完成】界面。

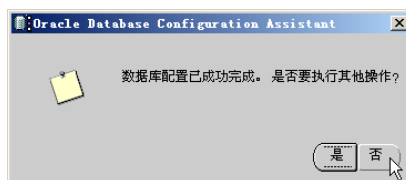


图 6.42 【配置过程】界面

图 6.43 【配置完成】界面

6.8 删除数据库

- (1) 启动【数据库配置助手】，一直到出现如图 6.44 所示的【操作】界面。
- (2) 出现如图 6.45 所示的【数据库】界面。

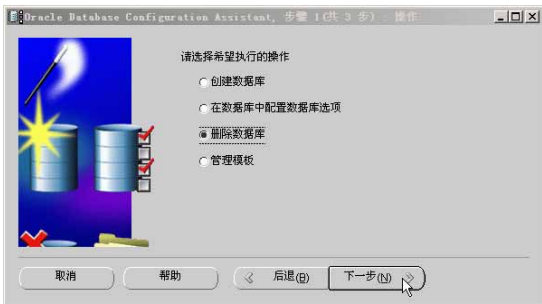


图 6.44 【操作】界面



图 6.45 【数据库】界面

- (3) 出现如图 6.46 所示的【概要】界面。
- (4) 出现如图 6.47 所示的【删除确认】界面。

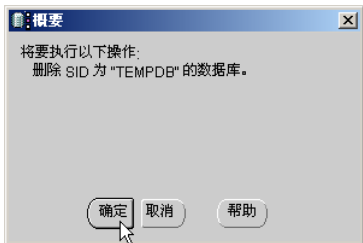


图 6.46 【概要】界面

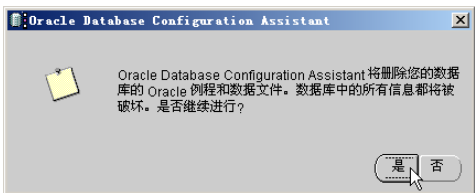


图 6.47 【删除确认】界面

- (5) 成功删除数据库后出现如图 6.48 所示的【成功删除】界面。单击 按钮。

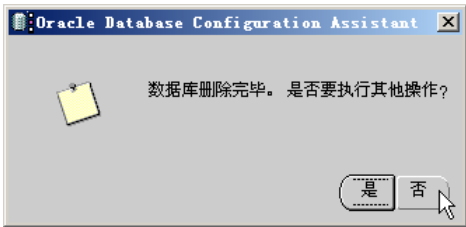


图 6.48 【成功删除】界面

6.9 定制数据库模板

6.9.1 什么是模板

数据库的模板与数据库的关系就犹如空白的档案登记表和实际的人员档案之间的关系。

6.9.2 系统默认的 3 种模板

选择【典型】安装数据库服务器时，已经内置了 3 种模板。如表 6.5 所示。

表 6.5 内置的 3 种模板

名称	用途
----	----

事务处理模板	适合联机事务处理（OLTP）数据库应用，比如各银行的储蓄业务系统、电信局的电话计费系统就是典型的 OLTP 的例子
数据仓库模板	适合数据仓库及联机分析处理（OLAP）应用，比如银行的管理层要制订业务发展计划时，决策人员就可以从数据仓库中提取有利于决策的数据
通用模板	创建的数据库可以适合作为事务处理应用，也可以作为数据仓库应用，两种应用兼而有之，这是数据库安装时的默认选项

6.9.3 如何定制个性化的模板

(1) 启动【数据库配置助手】，一直到出现如图 6.49 所示的【操作】界面。

(2) 出现如图 6.50 所示的【模板管理】界面。

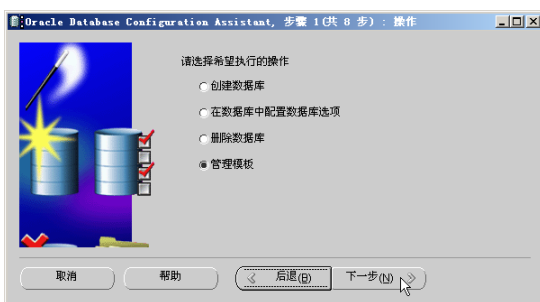


图 6.49 【操作】界面

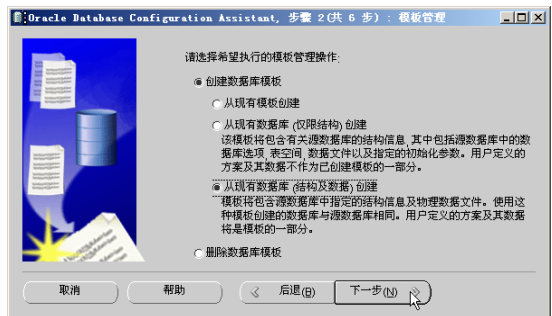


图 6.50 【模板管理】界面

(3) 出现如图 6.51 所示的【源数据库】界面。

(4) 出现如图 6.52 所示的【模板属性】界面。



图 6.51 【源数据库】界面



图 6.52 【模板属性】界面

(5) 出现如图 6.53 所示的【数据库相关文件位置】界面。

(6) 出现如图 6.54 所示的【概要】界面。

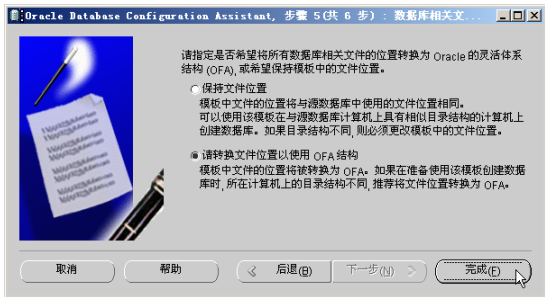


图 6.53 【数据库相关文件位置】界面

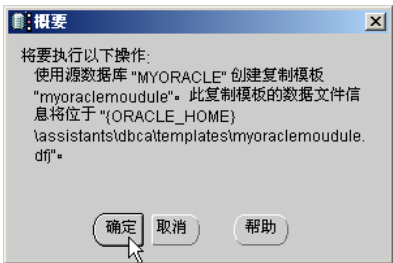


图 6.54 【概要】界面

(7) 出现如图 6.55 所示的【创建过程】界面。

(8) 成功创建数据库模板后出现如图 6.56 所示的【创建完成】界面。



图 6.55 【创建过程】界面

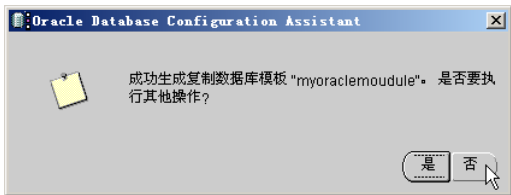


图 6.56 【创建完成】界面

(9) 在默认的安装目录下可以查看创建的模板文件，如图 6.57 所示。

名称	大小	类型	修改日期	属性
Data_warehouse.dbc	6 KB	DBC 文件	2001-09-12 17:10	A
Data_warehouse.dfn	286,370 KB	DFJ 文件	2001-09-04 20:55	A
General_Purpose.dbc	5 KB	DBC 文件	2001-09-12 17:10	A
New_Database.dbt	18 KB	DBT 文件	2001-09-12 17:10	A
Transaction_Processing.dbc	6 KB	DBC 文件	2001-09-12 17:10	A
Transaction_Processing.dfn	229,205 KB	DFJ 文件	2001-09-04 18:09	A
myoraclemodule.dbc	6 KB	DBC 文件	2003-09-10 8:28	A
myoraclemodule.dfn	159,577 KB	DFJ 文件	2003-09-10 8:28	A

图 6.57 模板文件

6.9.4 如何使用模板

(1) 将图 6.57 所示的模板文件复制到其他硬盘等载体上。

(2) 在要使用该模板的 Oracle 9i 数据库服务器的模板目录下将这两个文件拷贝进去。

(3) 调用【数据库配置助手】创建数据库，直到出现如图 6.58 所示的【数据库模板】界面。在模板文件目录下有的模板将出现在列表框中。



图 6.58 【数据库模板】界面

6.10 升级数据库

6.10.1 什么是升级

升级是将安装的 Oracle 数据库发行版转换为更高版本的发行版。例如，将 Oracle 9i 数据库 9.0.1.1.0 转换为 Oracle 9i 9.0.1.1.1 就是升级数据库系统。

数据库的移植和升级是有区别的。移植是将 Oracle 数据库的已安装版本转换为更高版本的过程。例如，将 Oracle 8 数据库转换为 Oracle 9i 数据库就是移植数据库系统。

6.10.2 升级的过程

(1) 如图 6.59 所示的【欢迎】界面。

(2) 出现如图 6.60 所示的【移植或升级准备】界面。



图 6.59 【欢迎】界面

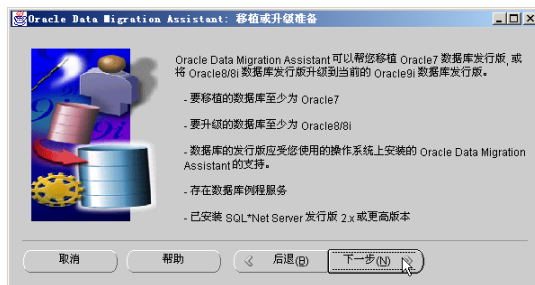


图 6.60 【移植或升级准备】界面

(3) 出现如图 6.61 所示的【请选择数据库例程】界面。

(4) 出现如图 6.62 所示的【数据库口令和 INIT.ORA 文件】界面。



图 6.61 【请选择数据库例程】界面

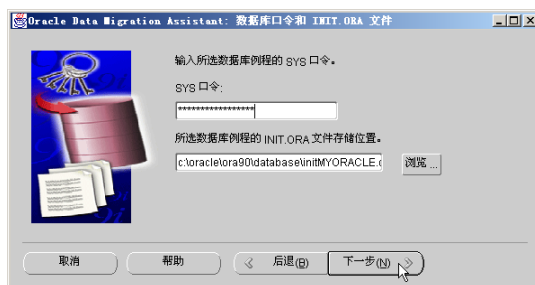


图 6.62 【数据库口令和 INIT.ORA 文件】界面

(5) 出现如图 6.63 所示的【获取数据库信息】界面。

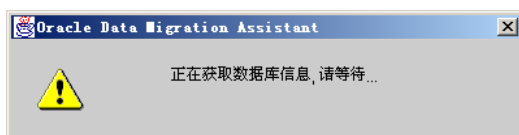


图 6.63 【获取数据库信息】界面

(6) 出现如图 6.64 所示的【错误】界面。

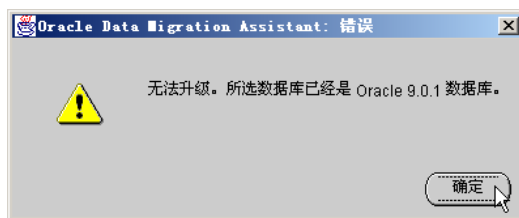


图 6.64 【错误】界面

6.11 使用数据字典

6.11.1 什么是数据字典

数据字典实际上也是以数据表和视图为主要存在形式的。可以这样说，数据字典是关于数据的数据表和视图。

6.11.1 如何使用数据字典

(1) 如图 6.65 所示。

(2) 出现如图 6.66 所示的【表编辑器】界面。

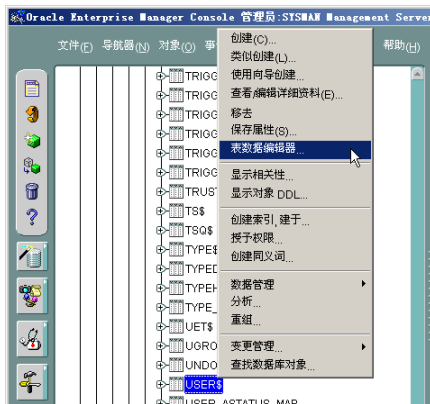


图 6.65 选择查看数据字典表 sys.user\$ 的数据

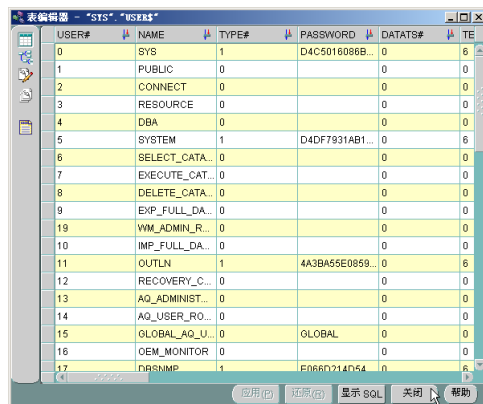


图 6.66 【表编辑器】界面

(3) 其他数据字典表和视图的内容查看这里不再一一列举。总之，管理员可以通过数据字典获得全面的数据库信息。

6.12 处理未决事务

6.12.1 为什么引入事务概念

事务就是当前主流的数据库系统普遍采用的并发控制机制。

6.12.2 什么是事务

事务具有以下 4 个最重要的特征，按照每个特性的英文单词的首字母组合成为 ACID 属性。

1. 原子性 (Atomicity)

原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

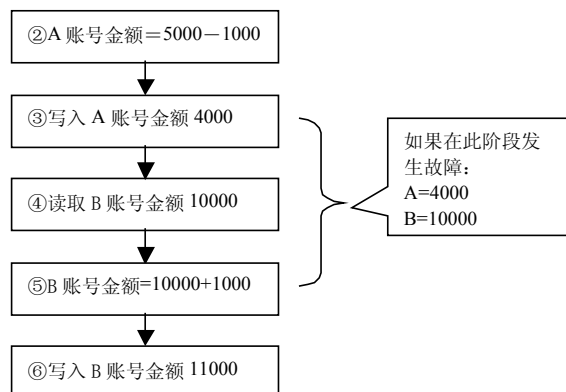


图 6.67 事务的处理流程

2. 一致性（Consistency）

事务必须使数据库从一个一致性状态变换到另外一个一致性状态。

3. 隔离性（Isolation）

事务的隔离性是指一个事务的执行不能被其他事务干扰，即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

4. 持久性（Durability）

持久性是指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来的其他操作和数据库故障不应该对其有任何影响。

6.12.3 什么是未决事务

对数据库进行操作的各种事务共有 5 种状态，如图 6.68 所示。

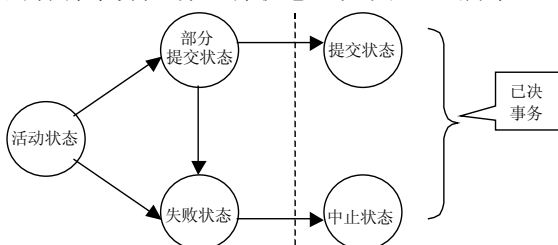


图 6.68 事务的状态

1. 活动状态

事务在执行时的状态叫活动状态。

2. 部分提交状态

事务中最后一条语句被执行后的状态叫部分提交状态。

3. 失败状态

事务不能正常执行的状态叫失败状态。

4. 提交状态

事务在部分提交后，将往硬盘上写入数据，当最后一条信息写入后的状态叫提交状态。进入提交状态的事务就成功完成了。

5. 中止状态

事务回滚并且数据库已经恢复到事务开始执行前的状态叫中止状态。

6.12.4 未决事务的处理方法——回滚

(1) 在【企业管理器】中编辑数据库的配置参数，切换到如图 6.69 所示的编辑数据库的【撤消】选项卡。

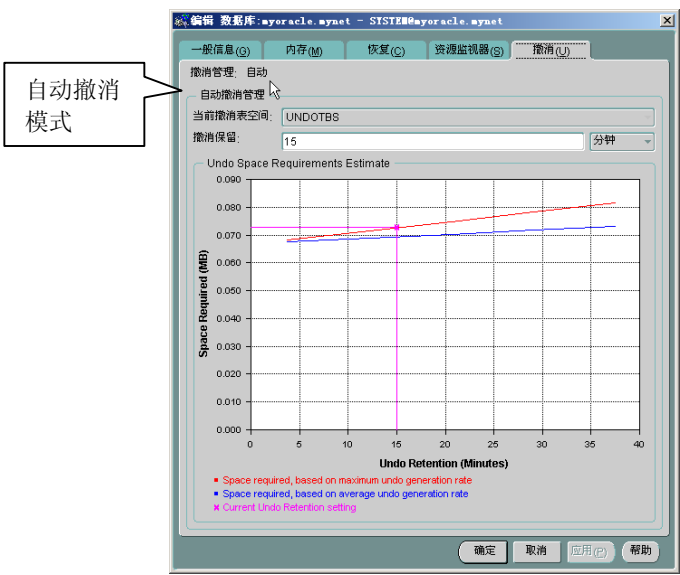


图 6.69 编辑数据库的【撤消】选项卡

(2) 如图 6.70 所示界面。

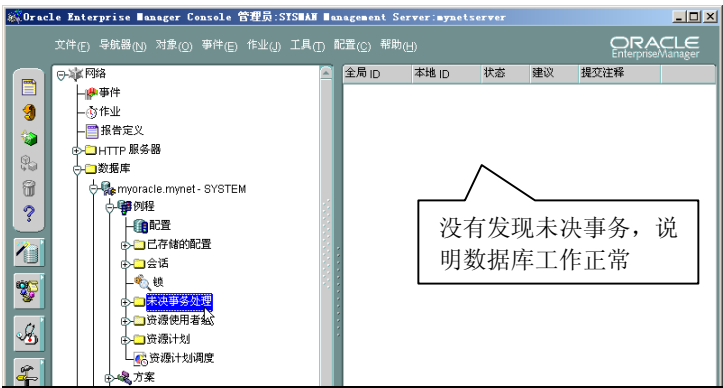


图 6.70 处理未决事务

6.13 锁

6.13.1 为什么引入锁概念

6.13.2 什么是锁

Oracle 9i 所有的锁的管理和分配都是由数据库管理系统自动完成的，不需要用户进行干预，同时也提供了手工加锁的命令，供有经验的用户使用。

6.13.3 锁的分类

- 1. 按照权限划分
 - ❑ 排它锁
 - ❑ 共享锁
- 2. 按照锁分配的资源划分
 - ❑ 数据锁
 - ❑ 字典锁

6.13.4 查询锁信息

(1) 如图 6.71 所示的数据库锁界面。

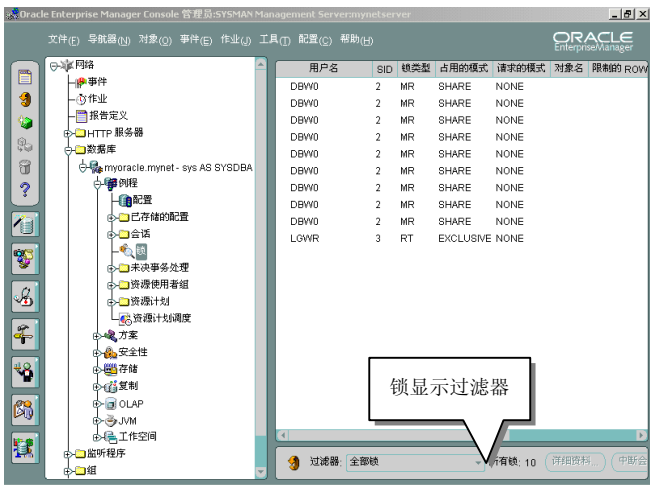


图 6.71 数据库的锁

(2) 在如图所示的【过滤器】下拉列表框里可以选择显示锁的类型，包括 3 种，默认为用户类型锁。

表 6.6 锁的主要信息

名称	含义
用户名	与占用或获取锁的会话相关联的 Oracle UserID
会话 ID	占用或获取锁的会话
锁类型	用户锁或系统锁的类型。 用户应用程序获取用户类型的锁。任何阻塞其他用户的进程就可能占用这些锁之一。用户类型锁包括 3 种。TM（DML 入队）、TX（事务处理入队）和 UL（所提供的用户）
占用的方式	会话所占用锁的锁定方式
请求的方式	进程请求锁的锁定方式
对象名	锁定对象的名称
限制的 ROWID	锁定的当前 ROWID

对象所有者	对象所有者的用户名
对象类型	对象类型
资源 ID1	锁标识符 #1 （取决于类型）
资源 ID2	锁标识符 #2 （取决于类型）

6.14 习题

（1）一个完整的 Oracle 网络环境有哪 4 个部分，各自的作用是什么？如何确保它们的正常运行？

（2）Pfile 和 Spfile 在数据库启动时有什么作用？和 init.ora 文件是什么关系？

（3）pfile 和 init.ora 文件的配合能否正常启动数据库？Spfile 和 init.ora 文件的配合能否正常启动数据库？试通过实验进行验证。

（4）什么情况下需要切换数据库的日志工作模式？如何切换？

（5）什么是会话？会话的主要内容是什么？

（6）创建数据库过程的主要步骤有哪些？

（7）如何删除数据库？

（8）什么是数据库的专用服务器模式？什么是共享服务器模式？二者各适用什么样的网络环境？

（9）Oracle 数据库的升级和移植有什么主要不同？

（10）模板的含义是什么？如何使用模板？试通过实验将本书配套光盘的模板文件分发自己的服务器环境。

（11）数据字典有什么作用？

（12）什么是事务？

（13）事务的 4 个特性是什么含义？

（14）什么是未决事务？Oracle 中是如何处理未决事务的？

（15）锁的主要含义和用途是什么？

第7章 数据管理——常用数据方案对象

本章通过实例的方法详细介绍 Oracle 9i 数据库中常用的数据方案对象的使用和管理方法。

7.1 Oracle 9i 的方案对象

7.1.1 数据方案对象

表 7.1 Oracle 9i 的数据方案对象

名称	含义
数据表	表是用于存放用户数据的数据库对象。数据库中的表按照功能的不同可以分为系统表和用户表两类。系统表用于存储管理用户数据和数据库本身的数据，又称数据字典表；用户表用于存放用户的数据信息，默认建立的数据表就是用户表
索引	索引就犹如一本书的目录，利用它可以快速找到所需要的内容。索引总是和数据表紧密相关联的
视图	视图是查看数据表中数据的一种方法，使用视图的主要目的就是确保数据表的安全性和隐蔽数据的复杂性。视图不是数据表，仅仅是一些 SQL 查询语句的定义

7.1.2 管理方案对象

表 7.2 Oracle 9i 的管理方案对象

名称	含义
数据库链接	管理分布式网络数据库服务器环境的登录用户名、口令和数据库位置
同义词	同义词用于隐藏表的用户名和数据表名，提供安全性
序列	直接产生唯一的顺序序号的一种方案对象
实体化视图	包含了对一个或多个数据表的查询结果的数据表，又称快照
实体化视图日志	记录上一次快照刷新后对数据表所做的所有修改信息的数据表，又称快照日志
刷新组	设置对数据库用户下的所有对象什么时间、按照什么条件进行刷新
簇	将一些互相关联，具有相同字段的数据表集中存储的一种管理结构
表类型	表类型也可以称为嵌套表类型，在表的定义里可以包含嵌套表

7.1.3 PL/SQL 编程方案对象

表 7.3 Oracle 9i 的 PL/SQL 编程方案对象

名称	含义
过程	过程也叫做存储过程,是由 SQL 语句和 PL/SQL 语句组合在一起为执行某一个任务的一个可执行单位,类似于高级程序设计语言中的模块
函数	函数和过程的结构类似。过程和函数差别在于,函数总返回单个值给调用者,而过程没有值返回给调用者
程序包	程序包也称为包,是被集中到一个单独的单元的一组过程、函数、变量和 SQL 语句的定义
程序包体	程序包体也称为包体,是与对应的程序包同名的,关于程序包内声明的函数、过程的详细执行代码
触发器	触发器是一种特殊类型的存储过程,由一些 SQL 语句组成,主要用于执行强制性的业务规则或要求,但不返回结果
对象类型	对象是面向对象分析与设计的基本概念,具有相同的属性和服务的对象被称为类,在 Oracle 9i 中把类称为对象类型,在一些数据库书籍里也称为抽象数据类型
数组类型	在 Oracle 8 中被称为可变数组。提供了自定义数组功能,包括数组元素的个数、元素的类型、长度和精度等。
高级队列	一种数据的存储结构,特点是先进先出
Java 源	一些 Java 源代码,这些源代码可以作为 Java 共享过程相互调用
维	对数据仓库中的数据进行分类的逻辑结构

7.1.4 如何查看方案对象

如图 7.1 所示界面。列举了 Oracle 9i 中的 21 种方案对象。



图 7.1 Oracle 9i 的方案对象

7.2 Oracle 9i 的数据类型

Oracle 9i 共提供了 16 种标量数据类型，如表 7.4 所示。

表 7.4 Oracle 9i 的标量数据类型

名称	含义
Char	用于描述定长的字符型数据，长度≤2000 字节
varchar2	用于描述变长的字符型数据，长度≤4000 字节
nchar	用来存储 Unicode 字符集的定长字符型数据，长度≤1000 字节
nvarchar2	用来存储 Unicode 字符集的变长字符型数据，长度≤1000 字节
number	用来存储整型或者浮点型数值
Date	用来存储日期数据
Long	用来存储最大长度为 2GB 的变长字符数据
Raw	用来存储非结构化数据的变长字符数据，长度≤2000 字节
Long raw	用来存储非结构化数据的变长字符数据，长度≤2GB
rowid	用来存储表中列的物理地址的二进制数据，占用固定的 10 个字节
Blob	用来存储多达 4GB 的非结构化的二进制数据
Clob	用来存储多达 4GB 的字符数据
nclob	用来存储多达 4GB 的 Unicode 字符数据
Bfile	用来把非结构化的二进制数据存储在数据库以外的操作系统文件中
urowid	用来存储表示任何类型列地址的二进制数据
float	用来存储浮点数

7.3 Oracle 9i 的数据表类型

表 7.5 Oracle 9i 的数据表类型

方式	特点
关系表	默认的表类型，存储永久性的数据，可以被分区，这样可以改善表的性能并易于管理
临时表	存储私有数据或一个会话中特定的数据，数据库中的其他用户不能使用这些数据
索引表	按照结构化主关键字进行排序的方式存储数据，和关系表不同的是不能把表和主关键字分开存储
外部表	数据存储在 Oracle 数据库外部的文件中，只能读，因此任何索引都不能存储在外部表中
对象表	支持面向对象的数据表

7.4 创建数据表

7.4.1 要创建的两个数据表

1. 研究生信息表

表名：scott.student。

表结构如表 7.6 所示。

表 7.6 scott.student 数据表结构

名称	数据类型	大小	小数位数	说明
STUDENT_ID	NUMBER	8	0	学生编号（主码）
NAME	VARCHAR2	10		姓名
PROFESSIONAL	VARCHAR2	20		专业
BIRTHDAY	DATE			生日
DIRECTOR_ID	NUMBER	6	0	导师编号（外码）

2. 导师信息表

表名：scott.director。

表结构如表 7.7 所示。

表 7.7 scott.director 数据表结构

名称	数据类型	大小	小数位数	说明
DIRECTOR_ID	NUMBER	6	0	导师编号（主码）
NAME	VARCHAR2	10		姓名
ZHICHENG	VARCHAR2	20		职称
ZHIWU	VARCHAR2	20		职务

3. 两个数据表的关系

导师信息表（scott.director）的主码“DIRECTOR_ID”是研究生信息表（scott.student）的外码。也就是说，当导师信息表的“DIRECTOR_ID”字段发生变化时，研究生信息表的字段“DIRECTOR_ID”也会随之自动发生变化。

7.4.2 创建的步骤

1. 创建导师信息表的步骤

（1）如图 7.2 所示界面。

（2）出现如图 7.3 所示的创建表的【一般信息】选项卡。

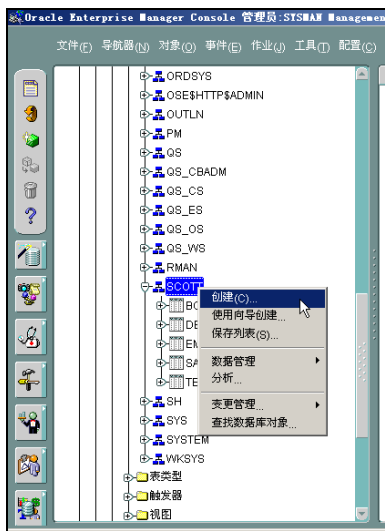


图 7.2 选择创建表

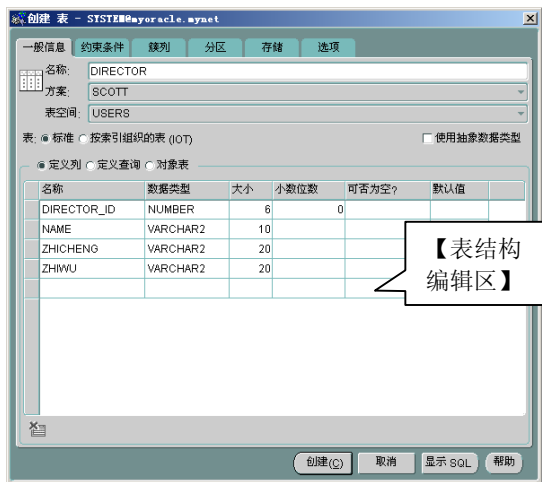


图 7.3 创建表的【一般信息】选项卡

(3) 切换到图 7.4 所示的创建表的【约束条件】选项卡。

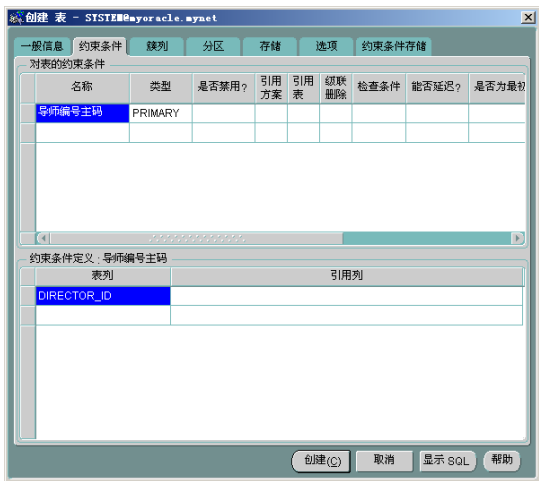


图 7.4 创建表的【约束条件】选项卡

在【类型】单元格下拉列表框中有 5 个选项，如表 7.8 所示。

表 7.8 约束条件类型

名称	含义
PRIMARY	主关键字，即主码
FOREIGN	外关键字，即外码
NOT NULL	非空，字段值不能为空值
UNIQUE	惟一，字段取值不能重复
CHECK	检查，检查是否满足指定的条件

- (4) 切换到图 7.5 所示的创建表的【簇列】选项卡。
(5) 切换到图 7.6 所示的创建表的【分区】选项卡。

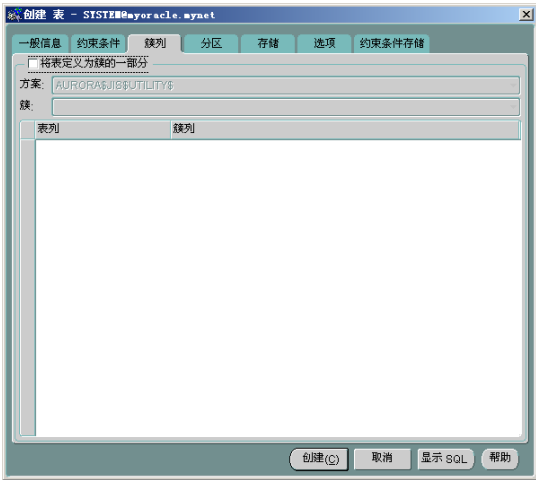


图 7.5 创建表的【簇列】选项卡



图 7.6 创建表的【分区】选项卡

- (6) 切换到图 7.7 所示的创建表的【存储】选项卡。
(7) 切换到图 7.8 所示的创建表的【选项】选项卡。

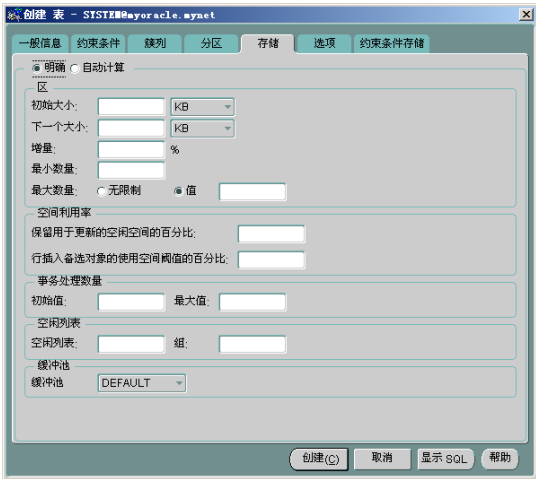


图 7.7 创建表的【存储】选项卡

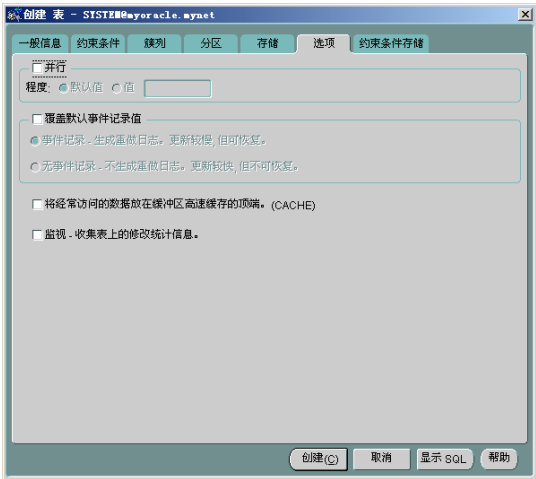


图 7.8 创建表的【选项】选项卡

- (8) 切换到图 7.9 所示的创建表的【约束条件存储】选项卡。
(9) 出现如图 7.10 所示界面。

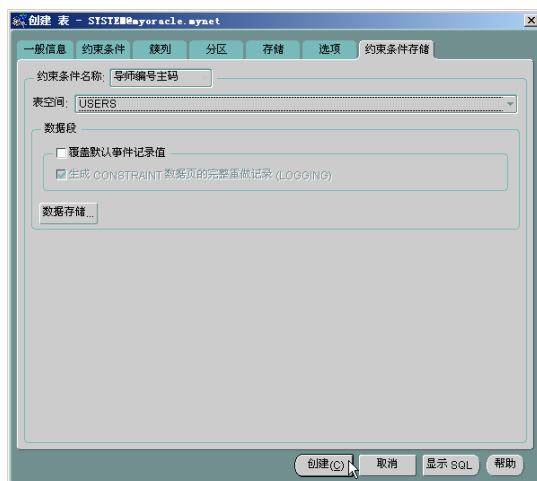


图 7.9 创建表的【约束条件存储】选项卡



图 7.10 成功创建表 scott.director

(10) 在【企业管理器】中可以查看创建的数据表，如图 7.11 所示。

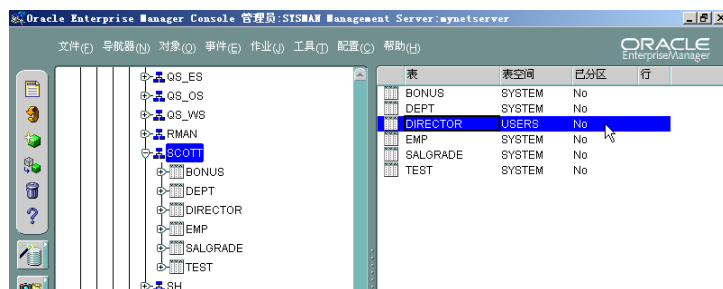


图 7.11 查询表 scott.director

(11) 上述步骤创建 scott.director 数据表的 SQL 代码如下。

```
CREATE TABLE "SCOTT"."DIRECTOR" (
    "DIRECTOR_ID"    NUMBER(6)        NOT NULL,
    "NAME"           VARCHAR2(10)     NOT NULL,
    "ZHICHENG"       VARCHAR2(20)    NOT NULL,
    "ZHIWU"          VARCHAR2(20)    NOT NULL,
    CONSTRAINT "导师编号主码" PRIMARY KEY("DIRECTOR_ID")
    USING INDEX
    TABLESPACE "USERS")
    TABLESPACE "USERS"
```

【参见光盘文件】：\第7章\createdirector.sql。



在创建主关键字约束条件时将自动建立该字段的索引。

(12) 读者也可以在【SQL Plus Worksheet】中直接执行 createdirector.sql 文件创建数据

表 scott.director，如图 7.12 所示。

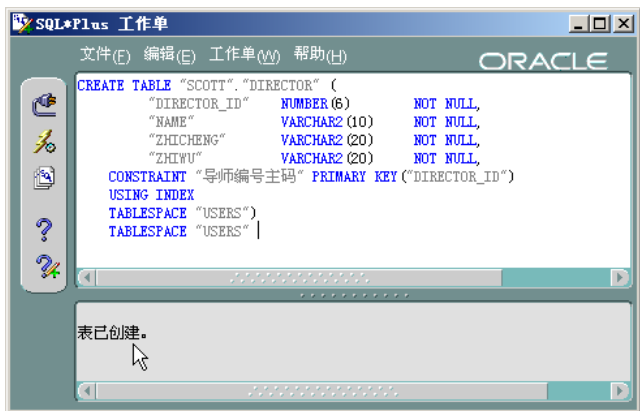


图 7.12 在【SQLPlus Worksheet】中创建数据表 scott.director

2. 创建研究生信息表的步骤

- (1) 与创建导师信息表一样进行操作。
- (2) 在图 7.13 所示的创建表的【一般信息】选项卡中按照如下步骤进行配置。
- (3) 图 7.14 所示为创建表的主码的【约束条件】选项卡。



图 7.13 创建表的【一般信息】选项卡

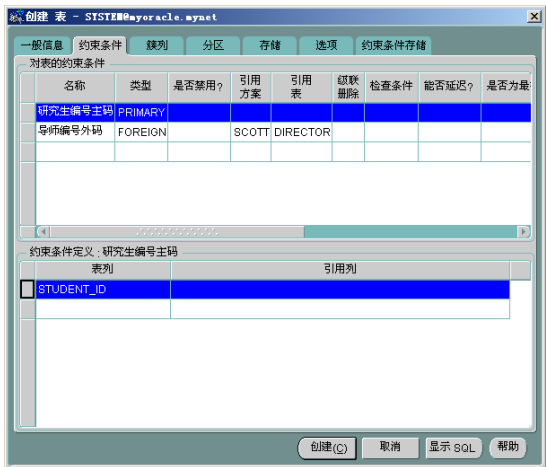


图 7.14 创建表的主码的【约束条件】选项卡

- (4) 图 7.15 所示为创建表的外码的【约束条件】选项卡。
- (5) 【簇列】、【分区】、【存储】和【选项】选项卡按照默认设置即可。
- (6) 图 7.16 所示的【约束条件存储】选项卡按照如下配置。

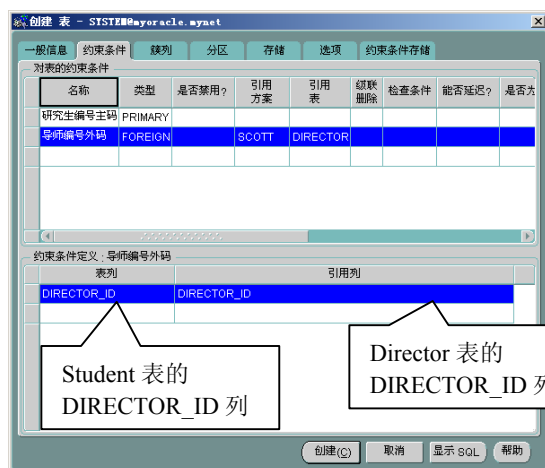


图 7.15 创建表的外码的【约束条件】选项卡

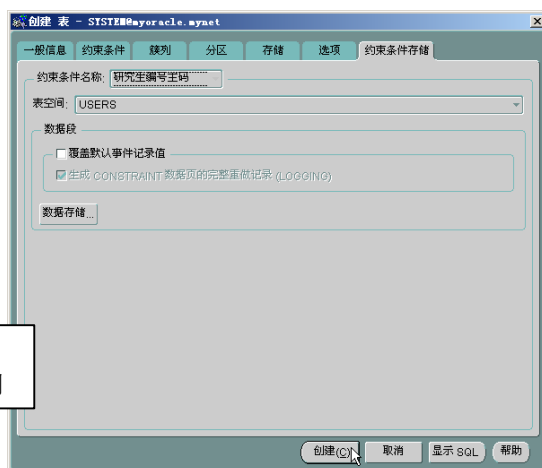


图 7.16 创建表的【约束条件存储】选项卡

(7) 系统将成功创建数据表 scott.student。对应的 SQL 代码如下。

```
CREATE TABLE "SCOTT"."STUDENT" (
    "STUDENT_ID" NUMBER(8) NOT NULL,
    "NAME" VARCHAR2(10) NOT NULL,
    "PROFESSIONAL" VARCHAR2(10) NOT NULL,
    "BIRTHDAY" DATE NOT NULL,
    "DIRECTOR_ID" NUMBER(6) NOT NULL,
    CONSTRAINT "研究生编号主码" PRIMARY KEY("STUDENT_ID")
    USING INDEX
    TABLESPACE "USERS",
    CONSTRAINT "导师编号外码" FOREIGN KEY("DIRECTOR_ID")
    REFERENCES "SCOTT"."DIRECTOR"("DIRECTOR_ID"))
    TABLESPACE "USERS"
```

【参见光盘文件】: \第 7 章

\createstudent.sql。

(8) 读者也可以在【SQL Plus Worksheet】中直接执行 createstudent.sql 文件创建数据表 scott.student, 如图 7.17 所示。

(9) 在【企业管理器】中可以查看建立的两个范例数据表, 如图 7.18 所示。

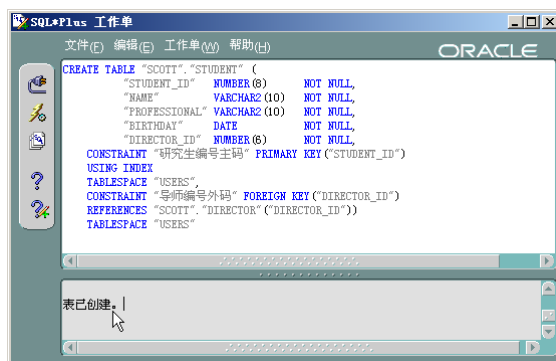


图 7.17 在【SQLPlus Worksheet】中创建数据表 scott.student

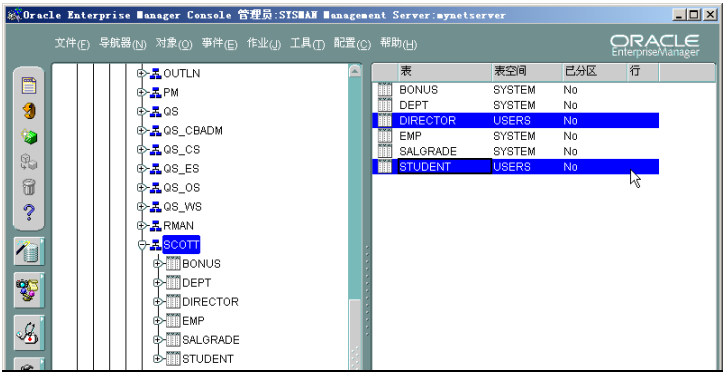


图 7.18 成功建立的两个范例数据表

7.5 修改数据表结构

(1) 如图 7.19 所示。

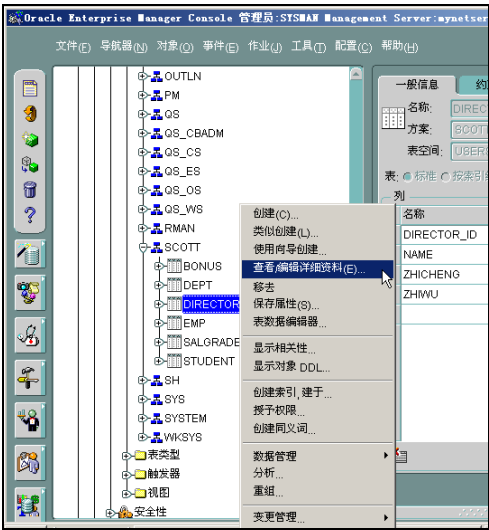


图 7.19 选择修改数据表结构



图 7.20 编辑表的【一般信息】选项卡

(2) 出现如图 7.20 所示的编辑表的【一般信息】选项卡。

(3) 其他选项卡的操作比较简单，这里不再赘述。

7.6 数据表中数据的管理

下面介绍如何操作数据表中的数据。

7.6.1 向数据表中插入数据

1. 向导师信息表中插入数据

(1) 在图 7.19 所示的快捷菜单中选择【表数据编辑器】选项，出现如图 7.21 所示的界面。可以按照二维表格输入数据。

(2) 单击 **显示 SQL** 按钮可以查看输入的数据对应的 SQL 代码。

(3) 数据输入完毕后，单击 **应用(P)** 按钮可以完成数据的插入。



图 7.21 表数据编辑器

(4) 上述过程对应的 SQL 代码如下。

```

INSERT INTO "SCOTT"."DIRECTOR" ("DIRECTOR_ID", "NAME", "ZHICHENG", "ZHIWU" )
VALUES (200201, '张三丰', '博导', '系主任');
INSERT INTO "SCOTT"."DIRECTOR" ("DIRECTOR_ID", "NAME", "ZHICHENG", "ZHIWU" )
VALUES (200202, '张翠山', '硕导', '室主任');
INSERT INTO "SCOTT"."DIRECTOR" ("DIRECTOR_ID", "NAME", "ZHICHENG", "ZHIWU" )
VALUES (200203, '张无忌', '硕导', '所主任');
INSERT INTO "SCOTT"."DIRECTOR" ("DIRECTOR_ID", "NAME", "ZHICHENG", "ZHIWU" )
VALUES (200204, '宋远桥', '博导', '系副主任');

```

【参见光盘文件】：\第7章\insertdirector.sql。

(5) 读者也可以在【SQL Plus Worksheet】中直接执行 insertdirector.sql 文件向数据表 scott.director 中插入数据。

2. 向研究生信息表中插入数据

(1) 对研究生信息表进行同样的数据插入操作。如图 7.22 所示。

STUDENT_ID	NAME	PROFESSIONAL	BIRTHDAY	DIRECTOR_ID
20020101	周芷若	软件工程		200201
20020102	赵敏	计算机安全		200202
20020103	小昭	图形图像		200203
20020104	蛛儿	电子商务		200204
20030101	金花婆婆	数据库		200201
20030102	胡青牛	网络安全		200203
20030103	丁敏君	Web技术		200201
20030104	殷素素	Web安全		200202

图 7.22 向 scott.student 表中插入的数据

(2) 字段“BIRTHDAY”是日期型字段，读者可能不知道该如何插入日期型的数据。单击 **显示 SQL** 按钮，出现如图 7.23 所示的界面。

可以观察到这样的日期型字段数据插入格式。

TO_DATE(' ', 'dd-Mon-yyyy HH:MI:SS AM')

这表明日期型字段是按照“日期-月份-一年 小时 分钟 秒”来插入的。

```
INSERT INTO 'SCOTT'.'STUDENT' ('STUDENT_ID', 'NAME', 'PROFESSIONAL', 'BIRTHDAY',
'DIRECTOR_ID') VALUES (20020101, '周芷若', '软件工程', TO_DATE(' ', 'dd-Mon-yyyy HH:MI:SS AM'), 200201)
INSERT INTO 'SCOTT'.'STUDENT' ('STUDENT_ID', 'NAME', 'PROFESSIONAL', 'BIRTHDAY',
'DIRECTOR_ID') VALUES (20020102, '赵敏', '计算机安全', TO_DATE(' ', 'dd-Mon-yyyy HH:MI:SS AM'), 200202)
INSERT INTO 'SCOTT'.'STUDENT' ('STUDENT_ID', 'NAME', 'PROFESSIONAL', 'BIRTHDAY',
'DIRECTOR_ID') VALUES (20020103, '小昭', '图形图像', TO_DATE(' ', 'dd-Mon-yyyy HH:MI:SS AM'), 200203)
INSERT INTO 'SCOTT'.'STUDENT' ('STUDENT_ID', 'NAME', 'PROFESSIONAL', 'BIRTHDAY',
'DIRECTOR_ID') VALUES (20020104, '蛛儿', '电子商务', TO_DATE(' ', 'dd-Mon-yyyy HH:MI:SS AM'), 200204)
```

图 7.23 分析日期型数据的插入格式

(3) 分析出格式还不一定能够正确录入日期型数据。这里告诉读者一个笔者摸索出来的简便的办法。在 scott 用户下有一个数据表 emp，其中有一个 hiredate 字段是日期型的。



图 7.24 日期型数据的范例

(4) 最后完成完整的 scott.student 数据表数据的插入，如图 7.25 所示。



图 7.25 完整的日期型数据插入实例

【参见光盘文件】：\第7章\insertstudent.sql。

(5) 上述过程对应的 SQL 代码如下。读者也可以在【SQLPlus Worksheet】中直接执行 insertstudent.sql 文件完成数据的插入。

```
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID", "NAME", "PROFESSIONAL", "BIRTHDAY", "DIRECTOR_ID")
VALUES (20020101, '周芷若', '软件工程', TO_DATE('20-11月-1976', 'dd-Mon-yyyy HH:MI:SS AM'), 200201);

INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID", "NAME", "PROFESSIONAL", "BIRTHDAY", "DIRECTOR_ID")
VALUES (20020102, '赵敏', '计算机安全', TO_DATE('7-10月-1980', 'dd-Mon-yyyy HH:MI:SS
```

```
AM'),200202 );
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID","NAME","PROFESSIONAL","BIRTHDAY","DIRECTOR_ID")
VALUES (20020103,'小昭','图形图像',TO_DATE('22-10月-1973','dd-Mon-yyyy HH:MI:SS
AM'),200203 );
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID","NAME","PROFESSIONAL","BIRTHDAY","DIRECTOR_ID")
VALUES (20020104,'蛛儿','电子商务',TO_DATE('11-1月-1971','dd-Mon-yyyy HH:MI:SS
AM'),200204 );
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID","NAME","PROFESSIONAL","BIRTHDAY","DIRECTOR_ID")
VALUES (20030101,'金花婆婆','数据库',TO_DATE('6-8月-1945','dd-Mon-yyyy HH:MI:SS
AM'),200201 );
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID","NAME","PROFESSIONAL","BIRTHDAY","DIRECTOR_ID")
VALUES (20030102,'胡青牛','网络安全',TO_DATE('2-5月-1923','dd-Mon-yyyy HH:MI:SS
AM'),200203 );
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID","NAME","PROFESSIONAL","BIRTHDAY","DIRECTOR_ID")
VALUES (20030103,'丁敏君','Web技术',TO_DATE('12-4月-1967','dd-Mon-yyyy HH:MI:SS
AM'),200201 );
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID","NAME","PROFESSIONAL","BIRTHDAY","DIRECTOR_ID")
VALUES (20030104,'殷素素','Web安全',TO_DATE('14-7月-1971','dd-Mon-yyyy HH:MI:SS
AM'),200202 );
Commit;
```

7.6.2 查询数据表的数据

1. 查询导师信息表数据

(1) 利用【SQLPlus Worksheet】执行下列语句。

```
select * from scott.director;
```

【参见光盘文件】: \第 7 章\selectdirector.sql。

(2) 查询结果如图 7.26 所示。

2. 查询研究生信息表数据

(1) 利用【SQLPlus Worksheet】执行下列语句。

```
select * from scott.student;
```

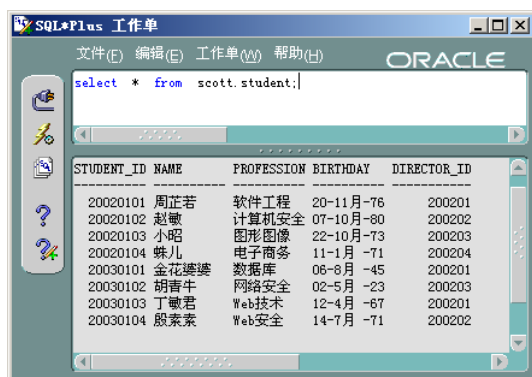
【参见光盘文件】：\第7章\selectstudent.sql。

(2) 查询结果如图 7.27 所示。



DIRECTOR_ID	NAME	ZHICHENG	ZHIWU
200201	张三丰	博导	系主任
200202	张翠山	硕导	室主任
200203	张无忌	硕导	所主任
200204	宋远桥	博导	系副主任

图 7.26 查询 scott.director 数据表的数据



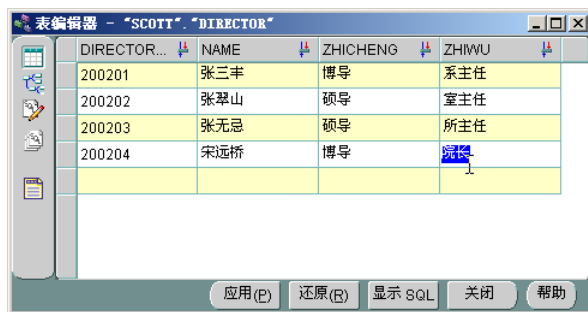
STUDENT_ID	NAME	PROFESSION	BIRTHDAY	DIRECTOR_ID
20020101	周正若	软件工程	20-11月-76	200201
20020102	赵敏	计算机安全	07-10月-80	200202
20020103	小昭	图形图像	22-10月-73	200203
20020104	蛛儿	电子商务	11-1月-71	200204
20030101	金花婆婆	数据库	06-8月-45	200201
20030102	胡青牛	网络安全	02-5月-23	200203
20030103	丁敏君	Web技术	12-4月-67	200201
20030104	殷素素	Web安全	14-7月-71	200202

图 7.27 查询 scott.student 数据表的数据

7.6.3 更新数据表的数据

1. 更新导师信息表数据

(1) 图 7.28 所示为数据表编辑器界面。在表格中直接将“DIRECTOR_ID”为“200204”的导师的“ZHIWU”由“系副主任”更改为“院长”。



DIRECTOR_ID	NAME	ZHICHENG	ZHIWU
200201	张三丰	博导	系主任
200202	张翠山	硕导	室主任
200203	张无忌	硕导	所主任
200204	宋远桥	博导	院长

图 7.28 在表数据编辑器界面中更新 scott.director 数据表的数据

(2) 单击 **显示 SQL** 按钮，出现的更新数据的 SQL 语句如下。

```
UPDATE "SCOTT"."DIRECTOR"
```

```
SET ZHIWU = '院长'
```

```
WHERE rowid = 'AAAH2kAAIAAAAYAAD'
```

这是什么东西？

(3) 读者可能会疑惑，rowid = 'AAAH2kAAIAAAAYAAD' 代表什么意思？

(4) Oracle 9i 在创建数据表时，默认会为每个数据表建立一个隐含的字段，叫 ROWID。在向数据表中插入记录时，系统将自动为每条记录分配惟一的一个 ROWID 号，利用这个

ROWID 号，可以快速定位到记录。

(5) 但是，利用 ROWID 更新数据的 SQL 语句只适用在同一台计算机上。不同的计算机环境可能为数据会分配不同的 ROWID 号，因此，我们需要给出一种普遍适用的更新数据表数据的语法。

(6) 在【SQLPlus Worksheet】中执行下列语句，作用是一样的。这是通过主码 DIRECTOR_ID 来定位记录的。这样的更新语句适合在不同的计算机环境上使用。

执行结果如图 7.29 所示。

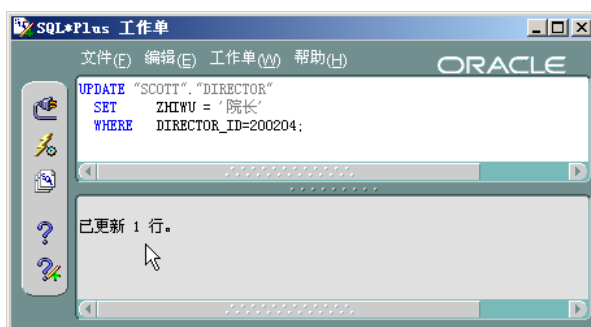


图 7.29 在【SQLPlus Worksheet】中更新 scott.director 的数据

【参见光盘文件】：\第 7 章\updatedirector.sql。

```
UPDATE "SCOTT"."DIRECTOR"
SET     ZHIWU = '院长'
WHERE   DIRECTOR_ID=200204;
```

2. 更新研究生信息表数据

按照同样的方法可以更新研究生数据表的数据。

(1) 在表数据编辑器中将研究生编号 STUDENT_ID 为“20030103”的研究生的导师编号字段 DIRECTOR 更改为 200204，如图 7.30 所示。

STUDENT_ID	NAME	PROFESSOR	BIRTHDAY	DIRECTOR
20020101	周正若	软件工程	20-十一月-1976...	200201
20020102	赵敏	计算机安全	07-十月-1980 ...	200202
20020103	小昭	图形图像	22-十月-1973 ...	200203
20020104	蛛儿	电子商务	11-一月-1971 ...	200204
20030101	金花婆婆	数据库	06-八月-1945 ...	200201
20030102	胡青牛	网络安全	02-五月-1923 ...	200203
20030103	丁敏君	Web技术	12-四月-1967 ...	200204
20030104	殷素素	Web安全	14-七月-1971 ...	200202

图 7.30 在表数据编辑器界面中更新 scott.student 数据表的数据

(2) 对应的 SQL 代码如下。

```
UPDATE "SCOTT"."STUDENT"
SET    DIRECTOR_ID = 200204
WHERE  STUDENT_ID = 20030103;
```

【参见光盘文件】: \第7章\updatestudent.sql。

7.6.4 删除数据表的数据

(1) 在图 7.31 所示的表数据编辑器界面中执行下列操作。



图 7.31 在表数据编辑器界面中更新 scott.student 数据表的数据

(2) 删除数据的 SQL 语句如下。

```
DELETE FROM "SCOTT"."STUDENT"
WHERE  STUDENT_ID = 20030101;
```

【参见光盘文件】: \第7章\deletestudent.sql。

7.7 截断数据表

截断操作的 SQL 语法如下。

```
TRUNCATE TABLE 用户名.表名 [DROP|REUSE STORAGE]
```

其中,若使用“DROP STORAGE”子句,显式指明释放数据表和索引的空间。若使用“REUSE STORAGE”子句,显式指明不释放数据表和索引的空间。下面以截断数据表 scott.director 为例进行介绍。

(1) 在【SQLPlus Worksheet】中执行 SQL 代码,结果如图 7.32 所示。

```
truncate table scott.director drop storage;
```

【参见光盘文件】: \第 7 章\truncatedirector.sql。

(2) 结果表明无法完成截断操作。这是因为导师信息表是父表, 研究生信息表引用导师信息表字段作为外码。

(3) 因此, 必须首先将 scott.student 的外码关系失效。在【SQLPlus Worksheet】中执行下列代码。执行结果如图 7.33 所示。

```
-----
alter table scott.student
disable constraint "导师编号外码";
-----
```

【参见光盘文件】: \第 7 章\disableconstraint.sql。

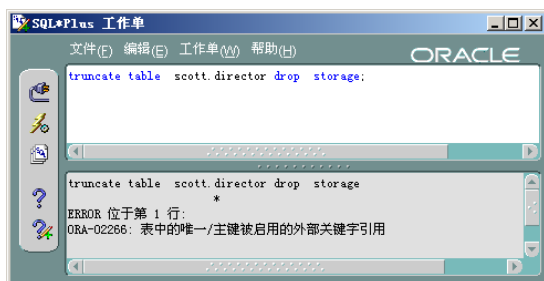


图 7.32 无法截断数据表的提示信息



图 7.33 断开子表 scott.student 的外码关系

(4) 重新在【SQLPlus Worksheet】中执行 truncatedirector.sql, 结果如图 7.34 所示。表明成功完成表的截断。

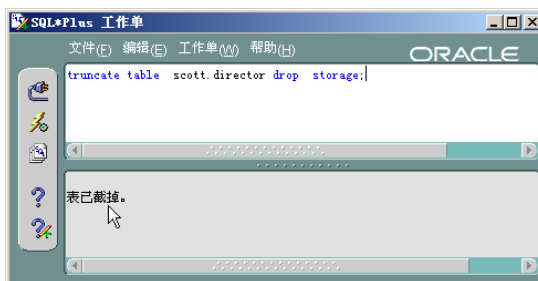


图 7.34 成功完成数据表的截断

7.8 删除数据表

数据表的删除比较简单, 表删除后其占用的空间就被系统释放和回收, 表的删除是无法回滚的操作。可删除的内容包括。

- ☐ 表的定义 表中的数据
- ☐ 表中的索引 表中的约束条件
- ☐ 表上的触发器 表中的权限

7.8.1 在【企业管理器】里删除表

- (1) 如图 7.35 所示。
- (2) 出现如图 7.36 所示界面。



图 7.35 选择删除表



图 7.36 确定删除表

- (3) 对应上述删除数据表 director 的 SQL 代码为如下。

```
DROP TABLE SCOTT.DIRECTOR CASCADE CONSTRAINTS;
```

【参见光盘文件】: \第 7 章\dropdirector.sql。

- (4) 在【SQLPlus Worksheet】中执行 dropdirector.sql 的结果如图 7.37 所示。

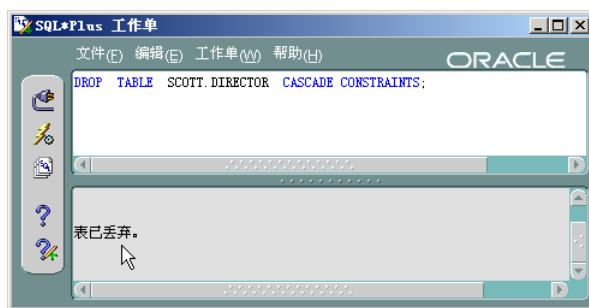


图 7.37 在【SQLPlus Worksheet】删除表

7.8.2 截断和删除的比较

TRUNCATE 对所有的表的操作速度都很快，DELETE 操作由于要产生大量的回滚信息，所以如果表很大，则删除速度较慢。

TRUNCATE 是 DDL 语言（数据定义语言），隐含了提交操作，因此不能回滚。

TRUNCATE 在表上或者在所有的索引中重新设置阈值,由于全部扫描操作和索引全表的快速操作将读所有未超过阈值的数据块,因此 DELETE 操作以后全表扫描的性能不会改进,但 TRUNCATE 操作以后全表扫描速度将加快。

截断表时,表和所有索引的存储参数可以设置为初始值,但 DELETE 操作不能缩小一个表及索引的大小。

截断操作不能删除完整性约束条件,而 DELETE 操作可以删除。

7.9 索引

索引是若干数据行的关键字的列表,查询数据时,通过索引中的关键字可以快速定位到要访问的记录所在的数据块,从而大大减少读取数据块的 I/O 次数,因此可以显著提高性能。

7.9.1 索引的原理

下面通过查询数据表“scott.student”的 ROWID 列为例,在【SQLPlus Worksheet】中执行下面的语句,查询结果如图 7.38 所示。

```
select rowid,student_id,name,professional,birthday,director_id
from scott.student;
```

【参见光盘文件】: \第 7 章\selectrowid.sql。

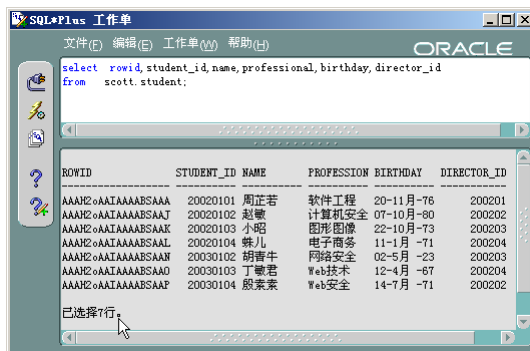


图 7.38 查询 ROWID 数据列

以数据表“scott.student”的索引为例,下面的数据就是以“student_id”数据列为例建立索引后的部分数据。

ROWID	STUDENT_ID
AAAH2oAAIAAAABSAAA	20020101
AAAH2oAAIAAAABSAAJ	20020102
AAAH2oAAIAAAABSAAK	20020103

AAAH2oAAIAAAABSAAL	20020104
AAAH2oAAIAAAABSAAN	20030102
AAAH2oAAIAAAABSAAO	20030103
AAAH2oAAIAAAABSAAP	20030104

7.9.2 Oracle 9i 支持的索引

Oracle 9i 中的索引可以分为两大类：B-树索引和位图索引。

1. B-树索引

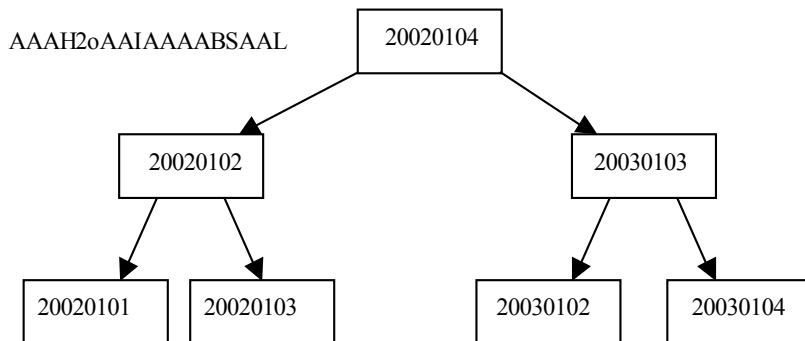


图 7.39 B-树索引的原理

表 7.9 B-树索引的分类

名称	含义
Non-Unique	非惟一索引，默认的 B-树索引，索引列值可以不是惟一的
Unique	惟一索引，在创建索引时指定“UNIQUE”关键字可以创建惟一索引。当建立“主码约束条件”时也会建立惟一索引，索引列值是惟一的
Reverse Key	反向关键字索引，通过在创建索引时指定“REVERSE”关键字，可以创建反向关键字索引，被索引的每个数据列中的数据都是反向存储的，但仍然保持原来数据列的次序
Function-based	基于函数的索引，对数据列使用表达式，按照表达式结果来创建 B-树索引的各节点，适合特定的，经常使用该表达式进行类似查询的数据表的索引的组织

2. 位图索引

对“scott.student”数据表的数据列“professional”建立位图索引可能的实例如下。

记录号	professional取值	位图索引值
1	软件工程	1
2	计算机安全	2
3	图形图像	3
.....		

7.9.3 主码自动建立的索引

(1) 如图 7.40 所示。

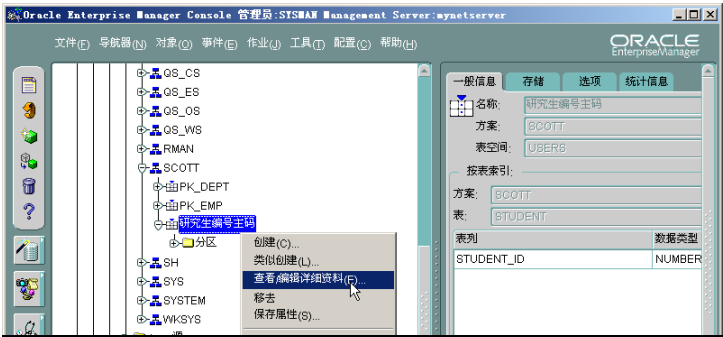


图 7.40 选择查看主码的索引

(2) 出现如图 7.41 所示的编辑索引的【一般信息】选项卡。



图 7.41 编辑索引的【一般信息】选项卡

7.9.4 如何创建索引

(1) 如图 7.42 所示。

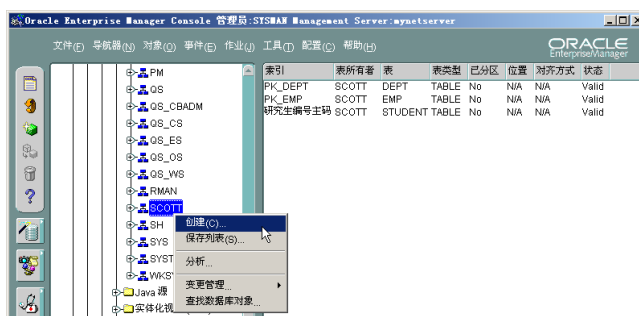


图 7.42 选择创建索引

(2) 出现如图 7.43 所示的创建索引的【一般信息】选项卡。

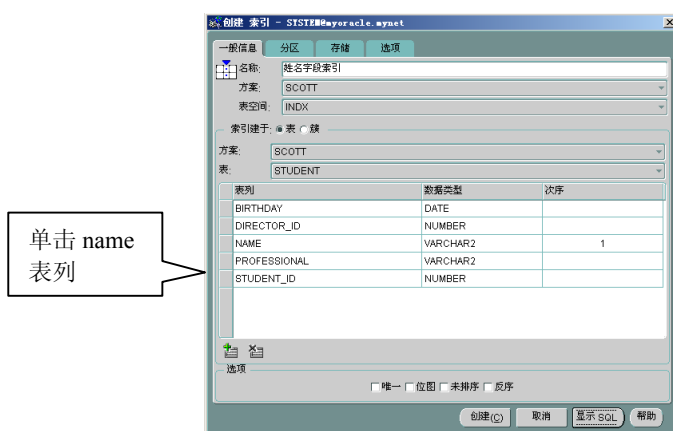


图 7.43 创建索引的【一般信息】选项卡

(3) 切换到图 7.44 所示的创建索引的【分区】选项卡。

(4) 切换到图 7.45 所示的创建索引的【存储】选项卡。

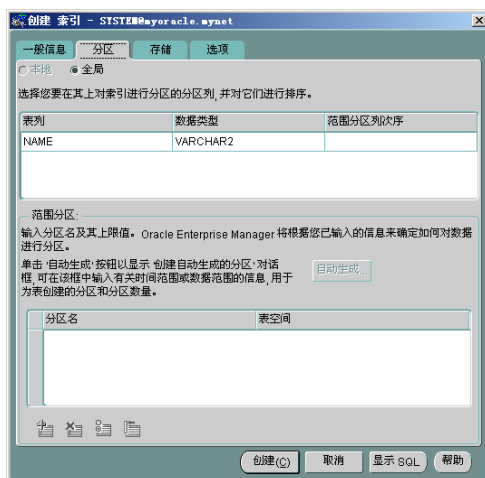


图 7.44 创建索引的【分区】选项卡

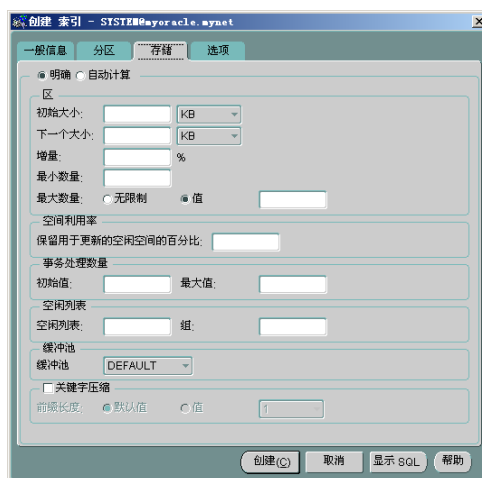


图 7.45 创建索引的【分区】选项卡

- (5) 切换到图 7.46 所示的创建索引的【选项】选项卡。
(6) 成功创建索引后出现如图 7.47 所示界面。

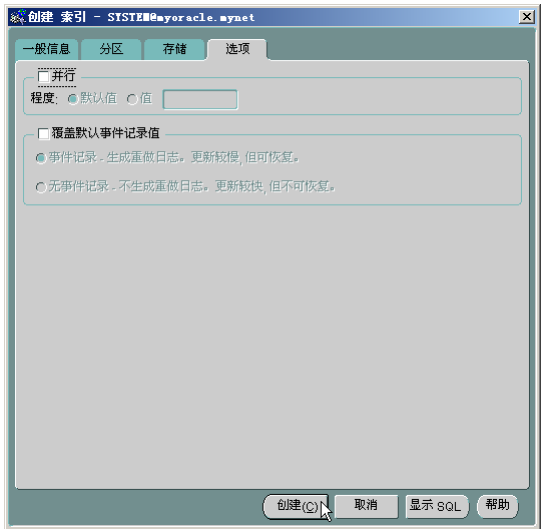


图 7.46 创建索引的【选项】选项卡



图 7.47 【成功创建索引】界面

- (7) 在【企业管理器】中可以查看创建的索引，如图 7.48 所示。

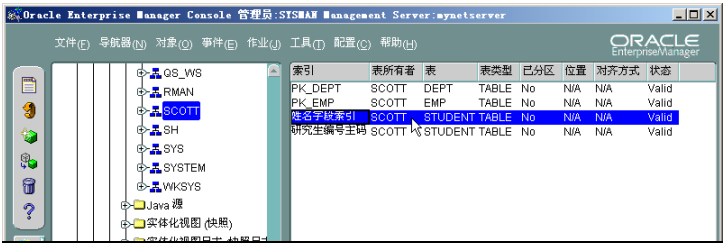


图 7.48 创建的索引

- (8) 上述创建索引对应的 SQL 代码如下。

```
CREATE INDEX "SCOTT"."姓名字段索引"
ON "SCOTT"."STUDENT" ("NAME")
TABLESPACE "INDX";
```

【参见光盘文件】：\第 7 章\createindex.sql。

- (9) 读者也可以直接在【SQLPlus Worksheet】中执行 createindex.sql 文件完成索引的创建，如图 7.49 所示。

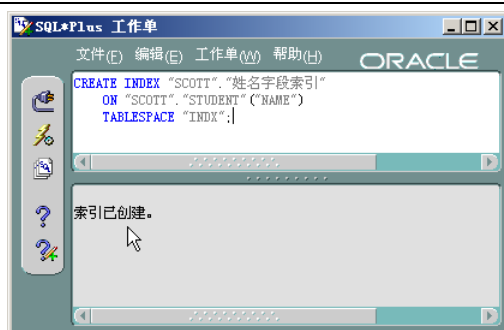


图 7.49 在【SQLPlus Worksheet】中创建索引

7.9.5 如何删除索引

- (1) 如图 7.50 所示。
- (2) 出现如图 7.51 所示的【索引删除确认】界面。

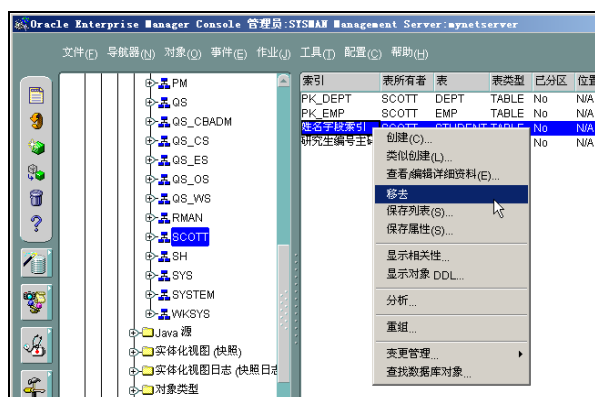


图 7.50 选择删除索引



图 7.51 【索引删除确认】界面

7.10 视图

7.10.1 什么是视图

视图犹如数据表的窗户，管理员定义这些“窗户”的位置后，用户就只能查看他可以看到的数据库数据。视图不是数据表，它仅是一些 SQL 查询语句的集合，作用是按照不同的要求从数据表中提取不同的数据。

7.10.2 如何创建视图

- (1) 如图 7.52 所示。
- (2) 出现如图 7.53 所示的创建视图的【一般信息】选项卡。

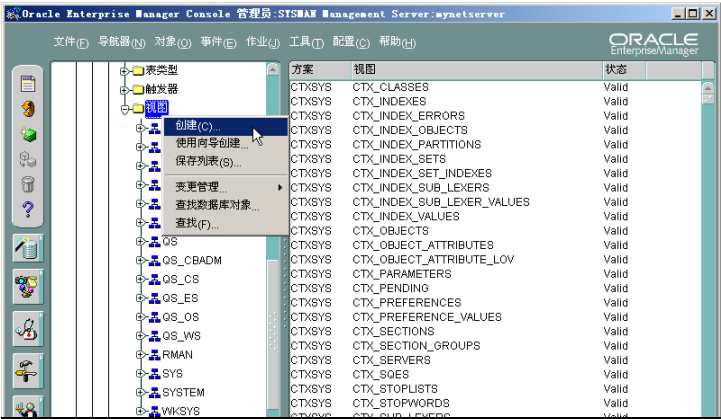


图 7.52 选择创建视图

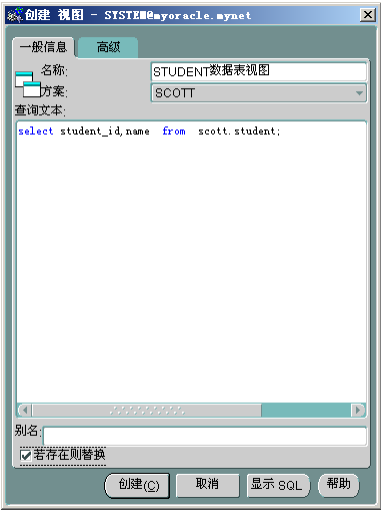



图 7.53 创建视图的【一般信息】选项卡

- (3) 切换到如图 7.54 所示的创建视图的【高级】选项卡。
- (4) 成功创建视图后出现如图 7.55 所示界面。单击  按钮。

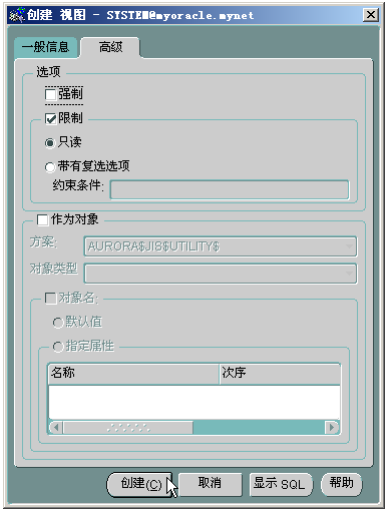


图 7.54 创建视图的【高级】选项卡



图 7.55 【成功创建视图】界面

- (5) 在【企业管理器】中可以查看创建的视图，如图 7.56 所示。

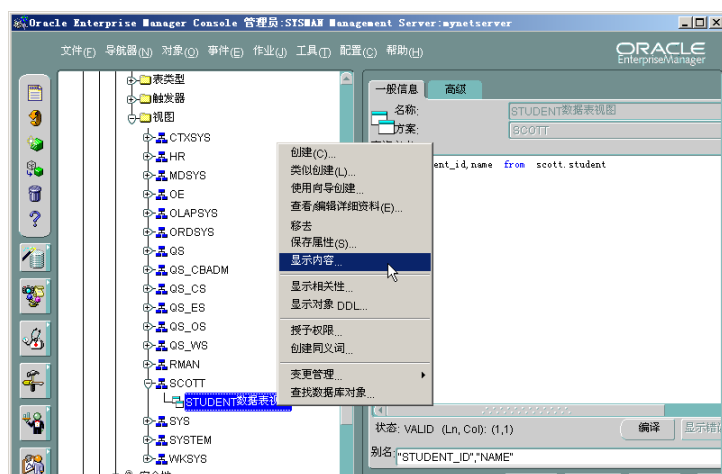


图 7.56 成功创建的视图

(6) 上述创建视图的 SQL 代码如下。

```
CREATE OR REPLACE VIEW "SCOTT"."STUDENT数据表视图" AS
  select student_id,name
  from scott.student WITH READ ONLY
```

【参见光盘文件】: \第7章\createview.sql。

(7) 读者也可以直接在【SQLPlus Worksheet】中执行 createview.sql 文件完成视图的创建，如图 7.57 所示。

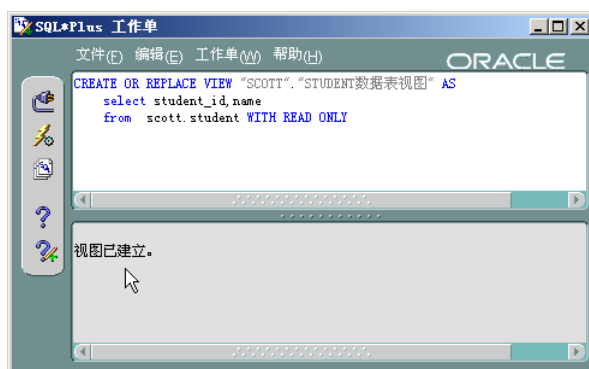


图 7.57 在【SQLPlus Worksheet】中创建视图

7.10.3 如何查询视图的数据

(1) 如图 7.56 所示。

(2) 出现如图 7.58 所示的【内容查看器】界面。显示了视图对应的数据。



图 7.58 【内容查看器】界面

(3) 上述过程对应的 SQL 代码如下。

```
select "SCOTT"."STUDENT数据表视图"."STUDENT_ID",
       "SCOTT"."STUDENT数据表视图"."NAME"
from "SCOTT"."STUDENT数据表视图"
```

【参见光盘文件】: \第 7 章\selectview.sql。

(4) 读者也可以直接在【SQLPlus Worksheet】中执行 selectview.sql 文件完成视图的查询，如图 7.59 所示。

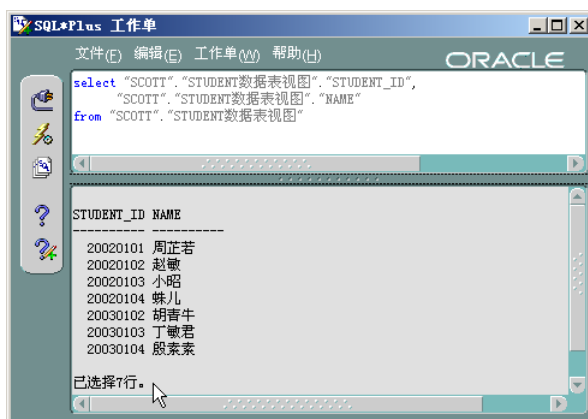


图 7.59 【内容查看器】界面

7.10.4 如何删除视图

(1) 如图 7.56 所示。

(2) 出现如图 7.60 所示的【视图删除确认】界面。

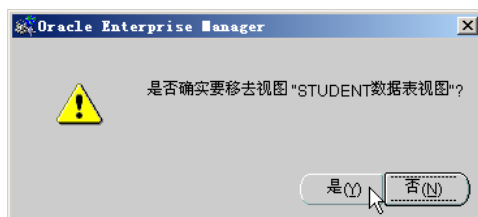


图 7.60 【视图删除确认】界面

(3) 这样，视图就被成功删除。

7.11 约束条件

约束条件就是 Oracle 数据库系统提供的对数据的完整性进行制约的机制。Oracle 9i 允许创建 5 种约束条件。参见表 7.8。

7.10.1 创建检查约束条件

- (1) 在【管理目标导航器】中按照 7.6 节修改数据表结构的步骤进行操作。
- (2) 切换到图 7.61 所示的编辑表的【约束条件】选项卡。
- (3) 上述创建检查约束条件的 SQL 代码如下。

```
ALTER TABLE "SCOTT"."STUDENT"
  ADD (CONSTRAINT "研究生编号检查约束条件"
  CHECK(student_id>=20020101 and student_id<=20030909))
```

【参见光盘文件】：\第 7 章\createcheck.sql。



图 7.61 编辑表的【约束条件】选项卡

(4) 读者也可以直接在【SQLPlus Worksheet】中执行 createcheck.sql 文件完成检查约束条件的创建, 如图 7.62 所示。

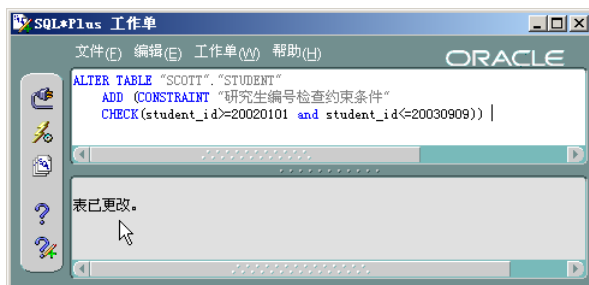


图 7.62 在【SQLPlus Worksheet】中创建检查约束条件

7.10.2 测试检查约束条件

(1) 在 7.63 所示的【表数据编辑器】界面中按照图示内容输入, 单击 **应用(P)** 按钮。

(2) 上述输入数据的 SQL 代码如下。

```
INSERT INTO "SCOTT"."STUDENT"
("STUDENT_ID", "NAME", "PROFESSIONAL", "BIRTHDAY", "DIRECTOR_ID")
VALUES (20010101, '纪晓芙', '软件工程', TO_DATE('15-7月 -1971', 'dd-Mon-yyyy HH:MI:SS
AM'), 200201)
```

【参见光盘文件】: \第 7 章\testcheck.sql。



图 7.63 在【表数据编辑器】中输入数据

(3) 出现如图 7.64 所示界面。



图 7.64 检查约束条件的执行结果

(4) 读者也可以直接在【SQLPlus Worksheet】中执行 testcheck.sql 文件完成检查约束条件的测试，结果如图 7.65 所示。

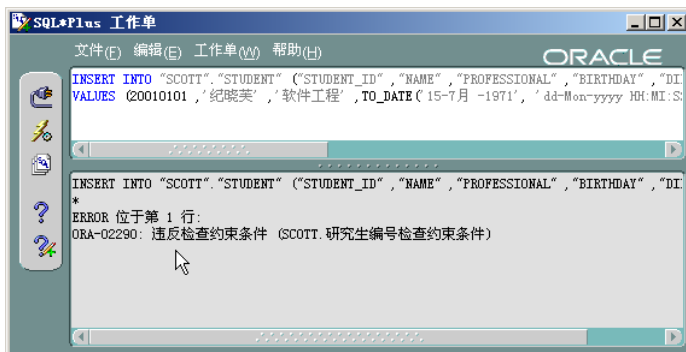


图 7.65 在【SQLPlus Worksheet】中测试检查约束条件

7.12 习题

- (1) Oracle 9i 中的 21 种方案对象，各有什么用途？
- (2) 通过实验熟悉 Oracle 9i 的 16 种基本数据类型的使用。
- (3) Oracle 9i 的数据表有哪些类型？各有什么用途？
- (4) 什么是主码？什么是外码？
- (5) 设计数据表的结构，通过实验完成数据表的创建。
- (6) 通过实验，熟悉数据表的数据的日常管理操作。
- (7) 在录入日期型字段的数据时，如何处理？
- (8) 截断和删除数据表有什么区别？
- (9) 索引的原理是什么？
- (10) B-树索引为什么能够加快数据查询速度？试通过实例说明 Oracle 9i 中的索引数据存储机制。
- (11) 视图有什么作用？
- (12) 检查约束条件的作用是什么？如何创建检查约束条件？
- (13) 自行设计有数据关联关系的至少两个数据表，独立完成数据表的创建、删除、数据的关联、索引的建立、视图的建立等环节，写出 SQL 代码报告，并阐述索引和视图建立的理由。
- (14) 自行设计数据表，设计主关键字、外关键字、惟一和检查约束条件，写出实验的 SQL 代码报告，并阐明设计约束条件的理由。
- (15) 通过实验，理解约束条件、视图、索引和数据表的关系。

第 8 章 安全管理——给用户授权

本章介绍 Oracle 9i 中安全性管理的内容。通过本章的学习，管理员可以全面掌握 Oracle 9i 数据库的安全性管理。

8.1 Oracle 数据库系统的安全性

8.1.1 Oracle 9i 的安全性体系

1. 物理层的安全性

数据库所在节点必须在物理上得到可靠的保护。

2. 用户层的安全性

哪些用户可以使用数据库，使用数据库的哪些数据对象，具有什么样的权限等。

3. 操作系统层的安全性

数据库所在的主机的操作系统的弱点将可能提供恶意攻击数据库的入口。

4. 网络层的安全性

Oracle 9i 数据库主要是面向网络提供服务，因此，网络软件的安全性和网络数据传输的安全性至关重要。

5. 数据库系统层的安全性

通过对用户授予特定的访问数据库对象的权利的办法来确保数据库系统层的安全。

8.1.2 Oracle 9i 的安全性机制

1. 系统安全性机制

系统安全性机制是指在整个的数据库系统级控制数据库的存取和使用的机制。

2. 数据安全性机制

数据安全性机制是指在对象级控制数据库的存取和使用的机制。

8.2 用户的管理

8.2.1 Oracle 9i 默认的用户

表 8.1 Oracle 9i 默认的主要用户

用户名	口令	登录身份及说明
sys	change_on_install	SYSDBA 或 SYSOPER，但不能以 NORMAL 登录，可作为默认的系统管理员
system	Manager	SYSDBA 或 NORMAL，但不能以 SYSOPER 登录，可作为默认的系统管理员
scott	Tiger	NORMAL，普通用户
aqadm	aqadm	SYSDBA 或 NORMAL，高级队列管理员。
Dbsnmp	dbsnmp	SYSDBA 或 NORMAL，复制管理员。

【参见光盘文件】：第 8 章\selectdbasusers.sql 和 selectuserusers.sql。

8.2.2 在【企业管理器】中如何创建用户

(1) 如图 8.1 所示。

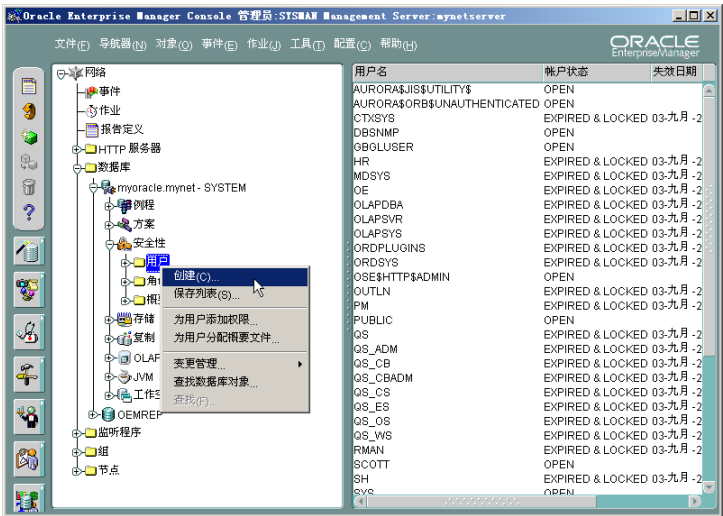


图 8.1 选择创建用户

(2) 出现如图 8.2 所示的创建用户的【一般信息】选项卡。

(3) 图 8.3 所示为创建用户的【角色】选项卡。

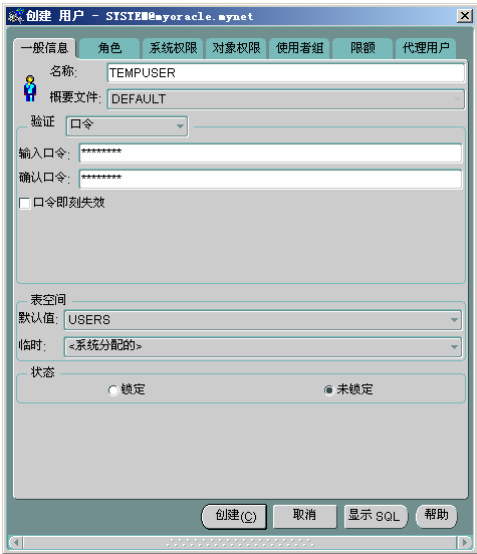


图 8.2 创建用户的【一般信息】选项卡

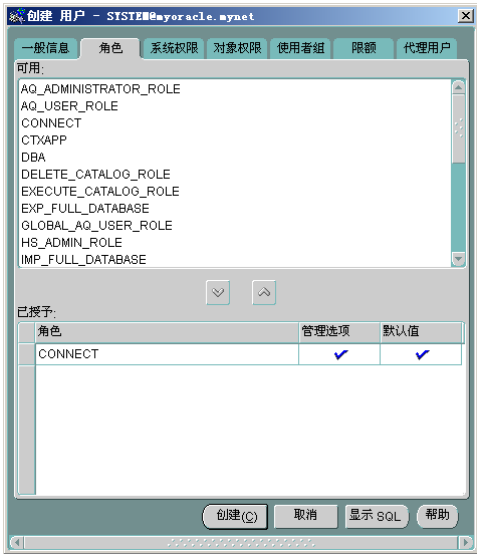


图 8.3 创建用户的【角色】选项卡

(4) 图 8.4 所示为创建用户的【系统权限】选项卡。

(5) 如图 8.5 所示为创建用户的【对象权限】选项卡。

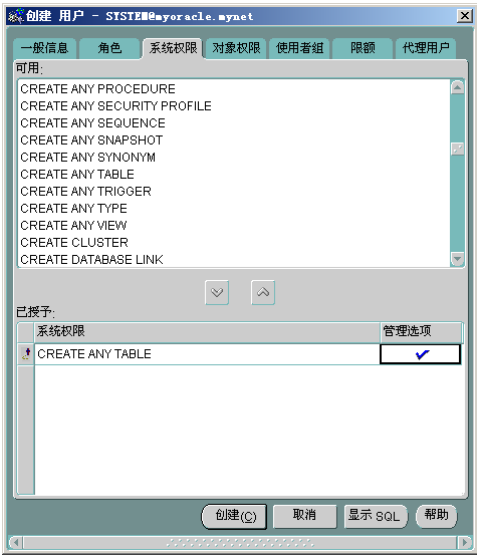


图 8.4 创建用户的【系统权限】选项卡

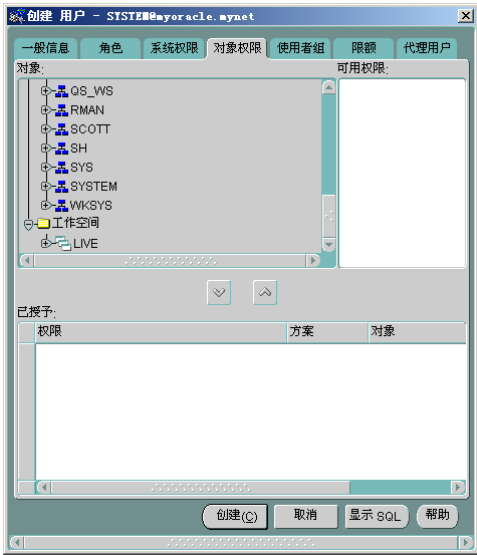


图 8.5 创建用户的【对象权限】选项卡

(6) 图 8.6 所示为创建用户的【使用者组】选项卡。

(7) 图 8.7 所示为创建用户的【限额】选项卡。

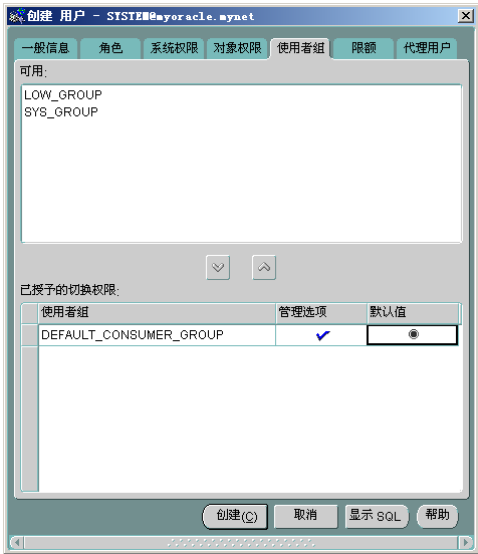


图 8.6 创建用户的【使用者组】选项卡



图 8.7 创建用户的【限额】选项卡

(8) 如图 8.8 所示为创建用户的【代理用户】选项卡。

(9) 成功创建用户后出现如图 8.9 所示界面。

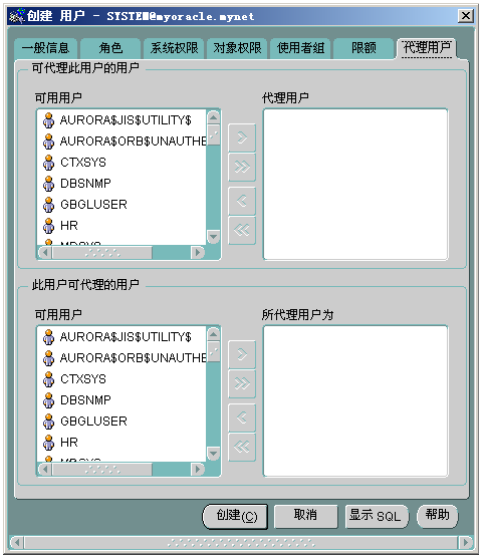


图 8.8 创建用户的【代理用户】选项卡



图 8.9 【成功创建用户】界面

(10) 上述过程对应的 SQL 代码如下。

```
CREATE USER "TEMPUSER" PROFILE "DEFAULT"
  IDENTIFIED BY "tempuser" DEFAULT TABLESPACE "USERS"
  ACCOUNT UNLOCK;
GRANT CREATE ANY TABLE TO "TEMPUSER" WITH ADMIN OPTION;
GRANT "CONNECT" TO "TEMPUSER" WITH ADMIN OPTION;
```

```

BEGIN
    dbms_resource_manager_privs.grant_switch_consumer_group(
        grantee_name => 'TEMPUSER',
        consumer_group => 'DEFAULT_CONSUMER_GROUP',
        grant_option => TRUE
    );
END;
BEGIN
    dbms_resource_manager.set_initial_consumer_group(
        user => 'TEMPUSER',
        consumer_group => 'DEFAULT_CONSUMER_GROUP'
    );
END;

```

【参见光盘文件】: 第 8 章\createtempuser.sql。

8.2.3 在【SQLPlus Worksheet】中如何创建用户

(1) 在【SQLPlus Worksheet】中不能直接执行 createtempuser.sql 文件完成用户的创建, 否则将出现错误。

(2) 将 createtempuser.sql 文件的执行分成 3 个步骤。

(3) 首先执行以下代码, 执行结果如图 8.10 所示。

```

-----
/*【一般信息】选项卡的配置*/
CREATE USER "TEMPUSER" PROFILE "DEFAULT"
    IDENTIFIED BY "tempuser" DEFAULT TABLESPACE "USERS"
    ACCOUNT UNLOCK;
/*【系统权限】选项卡的配置*/
GRANT CREATE ANY TABLE TO "TEMPUSER" WITH ADMIN OPTION;
/*【对象权限】选项卡的配置*/
GRANT "CONNECT" TO "TEMPUSER" WITH ADMIN OPTION;

```

【参见光盘文件】: 第 8 章\createtempuser-1.sql。

(4) 然后在【SQLPlus Worksheet】中执行下列代码, 执行结果如图 8.11 所示。

```

-----
/*【使用者组】选项卡的配置, 授予切换资源使用者组的权限*/
BEGIN
    dbms_resource_manager_privs.grant_switch_consumer_group(
        grantee_name => 'TEMPUSER',
        consumer_group => 'DEFAULT_CONSUMER_GROUP',
        grant_option => TRUE
    );
END;

```

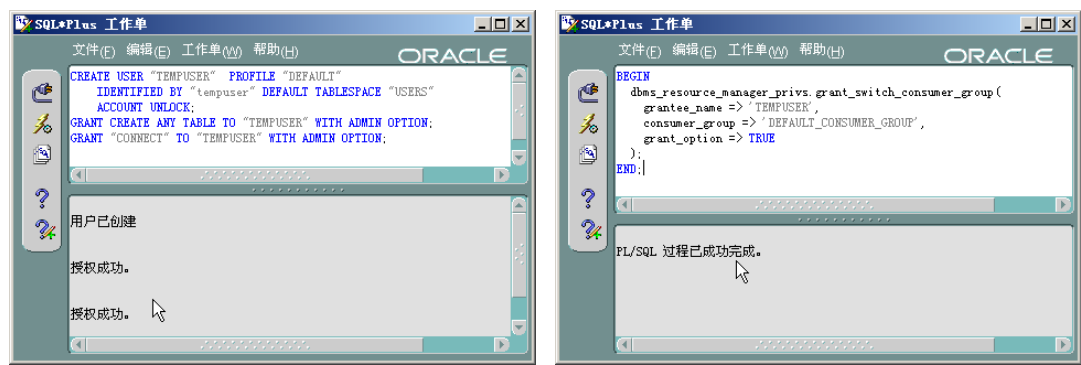


图 8.10 在【SQLPlus Worksheet】中创建用户第 1 步 图 8.11 在【SQLPlus Worksheet】中创建用户第 2 步
(5) 最后在【SQLPlus Worksheet】中执行下列代码，执行结果如图 8.12 所示。

```
-----
/*【使用者组】选项卡的配置，设置初始化资源使用者组*/
BEGIN
    dbms_resource_manager.set_initial_consumer_group(
        user => 'TEMPUSER',
        consumer_group => 'DEFAULT_CONSUMER_GROUP'
    );
END;
-----
```

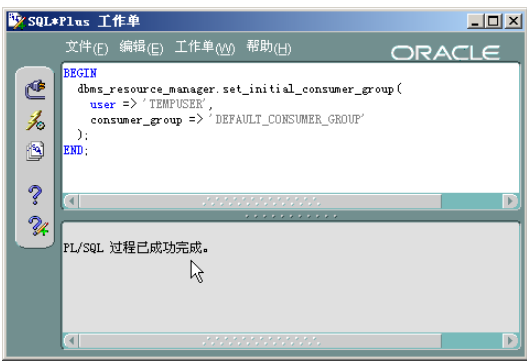


图 8.12 在【SQLPlus Worksheet】中创建用户第 3 步

8.2.4 创建用户中常见问题及解决方法

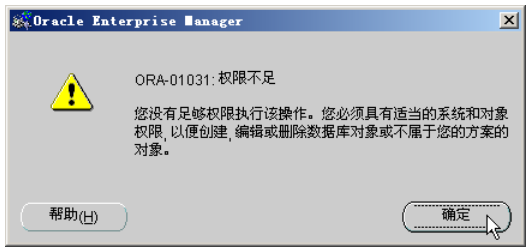


图 8.13 创建用户中权限不足的提示信息

8.2.5 用户的修改

(1) 如图 8.14 所示。

(2) 在出现的各选项卡中可以修改用户的各种配置参数。对应用户的修改的 SQL 语句为“ALTER USER”。

实例 1：将用户账号的状态设置为“锁定”的 SQL 代码如下。

```
ALTER USER "TEMPUSER" ACCOUNT LOCK
```

【参见光盘文件】：第 8 章\locktempuser.sql。

实例 2：修改用户的验证口令为“TEMP”的 SQL 代码如下。

```
ALTER USER "TEMPUSER" IDENTIFIED BY "temp"
```

【参见光盘文件】：第 8 章\passwordtempuser.sql。



图 8.14 选择修改用户

8.2.6 用户的删除

(1) 如图 8.15 所示的【用户删除确认】界面。



图 8.15 【用户删除确认】界面

(2) 上述过程对应的 SQL 代码如下。

```
DROP USER TEMPUSER CASCADE
```

【参见光盘文件】：第 8 章\droptempuser.sql。

8.3 角色的管理

8.3.1 Oracle 9i 预定义的角色

表 8.2 Oracle 9i 预定义的角色

角色名称	说明
CONNECT	数据库连接角色，用于连接数据库，具有创建簇、数据库链接、序列、同义词、表和视图，以及修改会话的权利
DBA	数据库管理员角色，具有所有使用 ADMIN 选项创建的系统权限，可以将系统权限授予其他用户或角色
DELETE_CATALOG_ROLE	删除目录角色，可以删除或重建数据字典
EXECUTE_CATALOG_ROLE	执行目录角色，能够执行所有系统包
EXP_FULL_DATABASE	能够使用导出程序执行数据库的完全和增量导出
IMP_FULL_DATABASE	能够使用导入程序执行数据库的完全导入
RESOURCE	可以创建簇、表、序列以及 PL/SQL 编程用方案对象，包括过程、程序包、触发器等
SELECT_CATALOG_ROLE	查询数据字典表或视图

8.3.2 在【企业管理器】中创建角色

(1) 如图 8.16 所示。

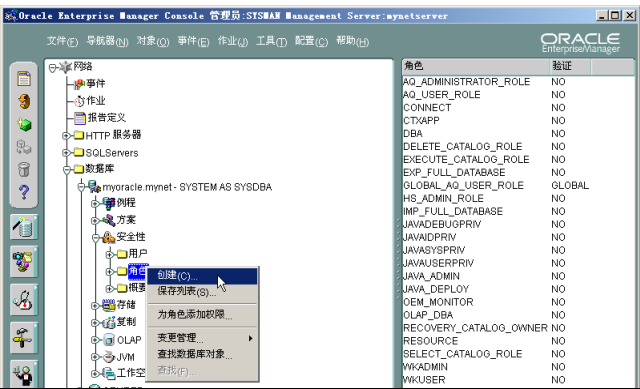


图 8.16 选择创建角色

(2) 出现如图 8.17 所示的创建角色的【一般信息】选项卡。

(3) 图 8.18 所示为创建角色的【角色】选项卡。用于为多个角色分配子角色。

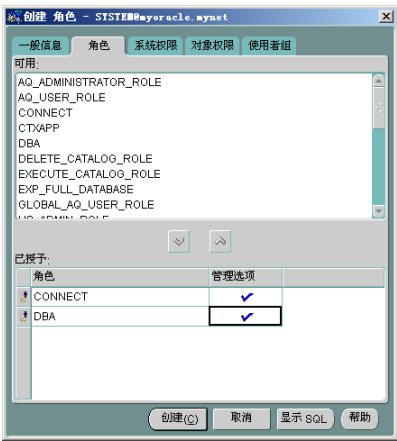


图 8.17 创建角色的【一般信息】选项卡 图 8.18 创建角色的【角色】选项卡
(4) 图 8.19 所示为创建角色的【系统权限】选项卡。
(5) 如图 8.20 所示为创建角色的【对象权限】选项卡。

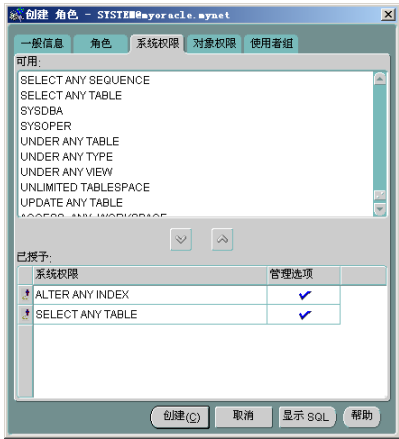


图 8.19 创建角色的【系统权限】选项卡 图 8.20 创建角色的【对象权限】选项卡
(6) 图 8.21 所示为创建角色的【使用者组】选项卡。
(7) 成功创建角色后出现如图 8.22 所示界面。



图 8.21 创建角色的【使用者组】选项卡



图 8.22 【成功创建角色】界面

(8) 上述过程创建角色的 SQL 代码如下。

```
CREATE ROLE "TEMPROLE"  
  IDENTIFIED BY "temprole";  
GRANT ALTER ANY INDEX TO "TEMPROLE" WITH ADMIN OPTION;  
GRANT SELECT ANY TABLE TO "TEMPROLE" WITH ADMIN OPTION;  
GRANT "CONNECT" TO "TEMPROLE" WITH ADMIN OPTION;  
GRANT "DBA" TO "TEMPROLE" WITH ADMIN OPTION;  
BEGIN  
  dbms_resource_manager_privs.grant_switch_consumer_group(  
    grantee_name => 'TEMPROLE',  
    consumer_group => 'DEFAULT_CONSUMER_GROUP',  
    grant_option => FALSE  
  );  
END;
```

【参见光盘文件】：第 8 章\createrole.sql。

8.3.3 在【SQLPlus Worksheet】中创建角色

(1) 在【SQLPlus Worksheet】中直接执行 createrole.sql 文件将完成角色的创建，执行结果如图 8.23 所示。

(2) 表明已经成功创建角色 TEMPROLE。

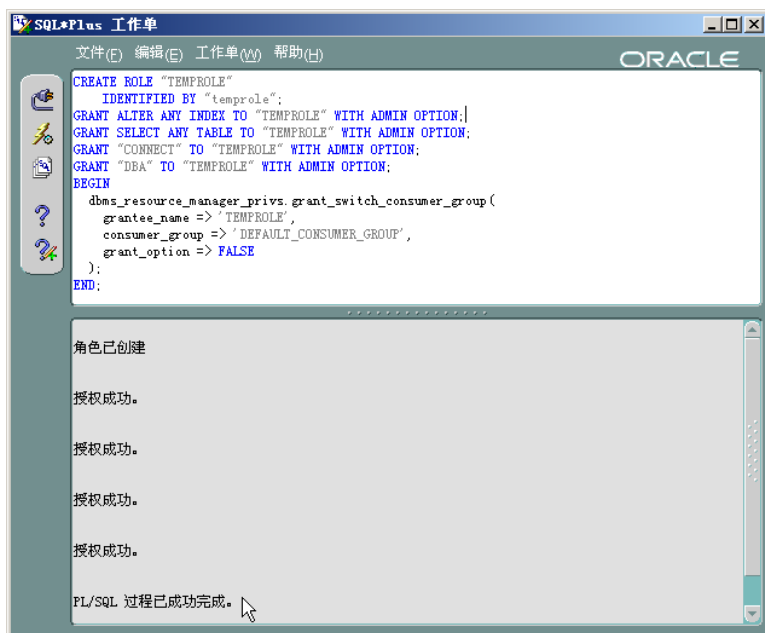


图 8.23 在【SQLPlus Worksheet】中创建角色

8.3.4 角色的修改

(1) 如图 8.24 所示。

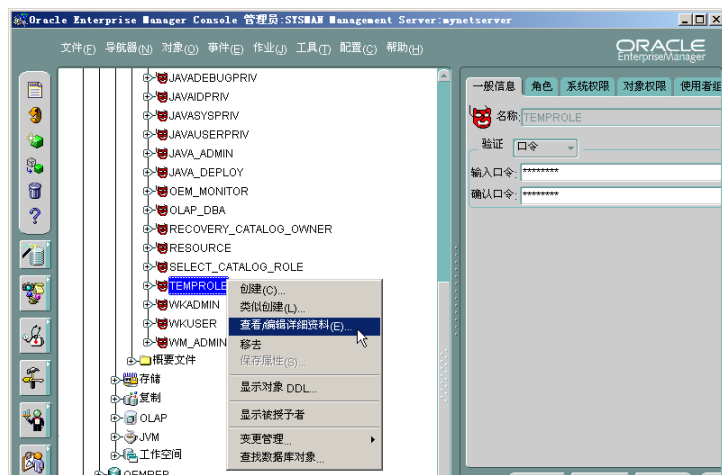


图 8.24 选择修改角色

(2) 在出现的各选项卡中可以修改角色的各种配置参数，对应角色的修改的 SQL 语句为“ALTER ROLE”或者“REVOKE”。

实例 1：将角色的验证方式更改为“外部”的 SQL 代码如下。

```
ALTER ROLE "TEMPROLE" IDENTIFIED EXTERNALLY;
```

【参见光盘文件】：第 8 章\alterrole.sql。

实例 2：将角色的系统权限“DBA”删除的 SQL 代码如下。

```
REVOKE "DBA" FROM "TEMPROLE";
```

【参见光盘文件】：第 8 章\revokerole.sql。

8.3.5 角色的删除

(1) 如图 8.25 所示【角色删除确认】界面。

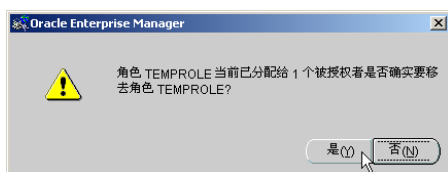


图 8.25 【角色删除确认】界面

(2) 上述过程对应的 SQL 代码如下。

```
DROP ROLE TEMPROLE;
```

【参见光盘文件】：第 8 章\droprole.sql。

8.4 概要文件的管理

8.4.1 在【企业管理器】中创建概要文件

(1) 如图 8.26 所示。

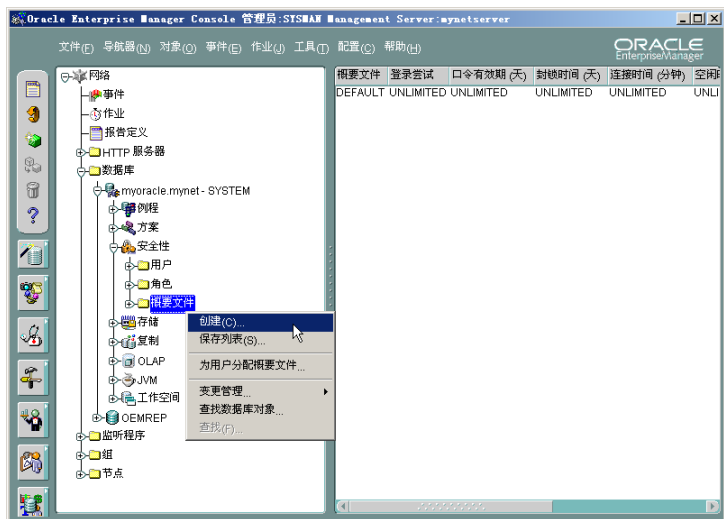


图 8.26 选择创建概要文件

(2) 出现如图 8.27 所示的创建概要文件的【一般信息】选项卡。



图 8.27 创建概要文件的【一般信息】选项卡

(3) 图 8.28 所示为创建概要文件的【口令】选项卡。



图 8.28 创建概要文件的【口令】选项卡

(4) 出现如图 8.29 所示界面。



图 8.29 【成功创建概要文件】界面

(5) 按照上述配置创建概要文件的 SQL 代码如下。

```
CREATE PROFILE "TEMPPROFILE"  
  /*【一般信息】选项卡对应的配置参数*/  
  LIMIT CPU_PER_SESSION 1000  
  CPU_PER_CALL 1000  
  CONNECT_TIME 30  
  IDLE_TIME DEFAULT  
  SESSIONS_PER_USER 10  
  LOGICAL_READS_PER_SESSION 1000  
  LOGICAL_READS_PER_CALL 1000  
  PRIVATE_SGA 16K  
  COMPOSITE_LIMIT 1000000  
  /*【口令】选项卡对应的配置参数*/  
  FAILED_LOGIN_ATTEMPTS 3  
  PASSWORD_LOCK_TIME 5  
  PASSWORD_GRACE_TIME 60  
  PASSWORD_LIFE_TIME 30  
  PASSWORD_REUSE_MAX DEFAULT  
  PASSWORD_REUSE_TIME 30  
  PASSWORD_VERIFY_FUNCTION DEFAULT
```

【参见光盘文件】：第 8 章\createprofile.sql。

8.4.2 在【SQLPlus Worksheet】中创建概要文件

在【SQLPlus Worksheet】中直接执行 createprofile.sql 文件将完成概要文件的创建，执行结果如图 8.30 所示。表明已经成功创建概要文件 TEMPPROFILE。

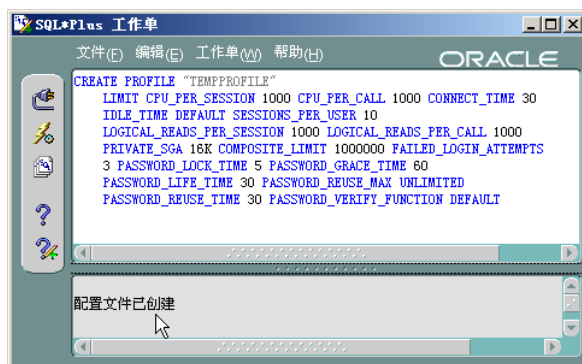


图 8.30 在【SQLPlus Worksheet】中创建概要文件

8.4.3 概要文件的修改

(1) 如图 8.31 所示。

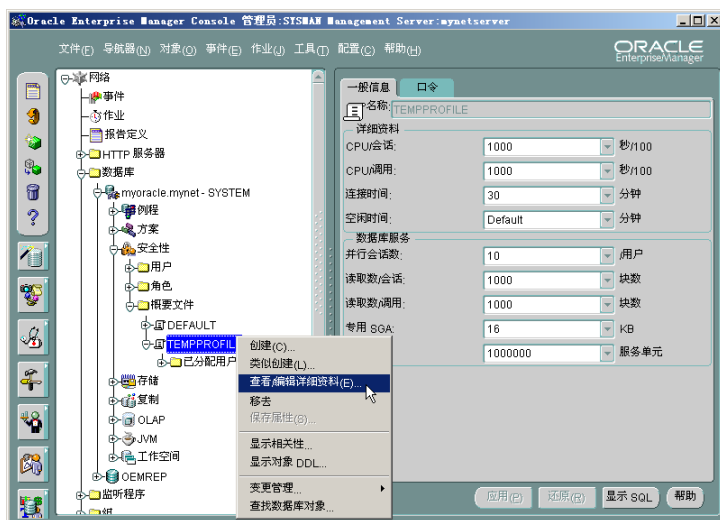


图 8.31 选择修改概要文件

(2) 在出现的编辑概要文件的【一般信息】和【口令】选项卡中可以修改概要文件的配置参数，对应修改概要文件的 SQL 语句为“ALTER PROFILE”。

实例：将 CPU/会话参数设置为 6000 的 SQL 代码如下。

```
ALTER PROFILE "TEMPPROFILE" LIMIT CPU_PER_SESSION 6000
```

【参见光盘文件】：第 8 章\alterprofile.sql。

8.4.4 将概要文件分配给用户

(1) 如图 8.32 的【分配概要文件】界面。



图 8.32 【分配概要文件】界面

(2) 上述过程对应的 SQL 代码如下。

```
ALTER USER TEMPUSER PROFILE TEMPPROFILE;
```

【参见光盘文件】：第 8 章\alteruserprofile.sql。

8.4.5 概要文件的删除

(1) 如图 8.33 所示的【概要文件删除确认】界面。

(2) 删除概要文件的 SQL 代码如下。

```
DROP PROFILE TEMPPROFILE
```

【参见光盘文件】：第 8 章\dropprofile.sql。



图 8.33 【删除概要文件确认】界面

8.5 审计

8.5.1 审计的作用

1. 审查可疑的活动
2. 监视和收集关于指定数据库活动的的数据

8.5.2 审计的类型

1. 语句审计 (STATEMENT AUDITING)
2. 权限审计 (PRIVILEGE AUDITING)
3. 对象审计 (OBJECT AUDITING)

8.5.3 审计的信息

AUD\$表记录的审计信息包括。

- ☐ SESSIONID: 会话的数字 ID。
- ☐ ENTRYID: 审计信息项的 ID。
- ☐ STATEMENT: 每个执行的命令的数字 ID。
- ☐ TIMESTAP#: 设计信息生成的日期和时间。
- ☐ USERID: 被审计的用户使用的 Oracle 用户 ID。
- ☐ USERHOST: 被审计的用户使用的数据库例程的数字 ID。
- ☐ TERMINAL: 被审计的用户的操作系统终端描述字。
- ☐ ACTION#: 被审计的操作的标识。
- ☐ RETURNCODE: 每个被审计的命令执行后的返回代码, 若为 0, 表明操作成功。
- ☐ OBJ\$CREATOR: 被一个操作影响到的对象的创建者 (对操作审计)。
- ☐ OBJ\$NAME: 被一个操作影响到的对象的名称 (对操作审计)。
- ☐ AUTH\$PRIVILEGES: 使用的系统权限。
- ☐ AUTH\$GRANTEE: 使用的对象权限。
- ☐ NEW\$OWNER: 在列 NEW_NAME 中命名的对象的所有者。
- ☐ NEW\$NAME: 在列 NEW_NAME 中命名的对象的名称。
- ☐ SESSACTIONS: 会话小结的字符串, 记录了不同操作的成功和失败的信息。
- ☐ SES\$TID: 会话的事务 ID。
- ☐ LOGOFF\$LREAD: 在会话中执行的逻辑读个数。
- ☐ LOGOFF\$PREAD: 在会话中执行的物理读个数。
- ☐ LOGOFF\$LWRITE: 在会话中执行的逻辑写个数。
- ☐ LOGOFF\$DEAD: 在会话中检测到的死锁个数。
- ☐ LOGOFF\$TIME: 用户退出系统的日期和时间。
- ☐ COMMENT\$TEXT: 对设计信息项的文本注释。
- ☐ CLIENTID: 客户机 ID。
- ☐ SPARE1: 备用。

- ❑ SPARE2: 备用。
- ❑ OBJ\$LABEL: 与对象关联的标签。
- ❑ SES\$LABEL: 与会话关联的标签。
- ❑ PRIV\$USED: 执行操作的系统权限。
- ❑ SESSIONCPU: 会话占用的 CPU 时间。

8.5.4 审计的启动

如图 8.34 所示的编辑数据库配置的【所有参数】选项卡。

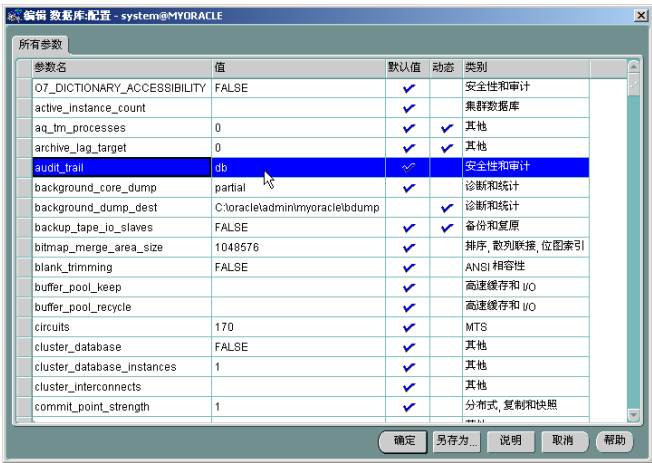


图 8.34 编辑数据库配置的【所有参数】选项卡

8.5.5 审计的实例

(1) 以 SYSTEM 用户登录【SQLPlus Worksheet】，执行如下 SQL 代码，执行结果如图 8.35 所示。

AUDIT SESSION;

【参见光盘文件】：第 8 章\auditsession.sql。

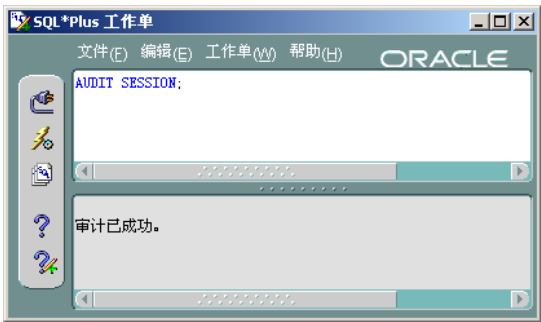


图 8.35 审计数据库的断开及连接操作

(2) 以 SCOTT 用户登录另外一个【SQLPlus Worksheet】。

(3) 查询 AUD\$表的内容，主要的审计信息如下。

```
-----  
SESSIONID: 518  
ENTRYID: 1  
STATEMENT: 1  
TIMESTAMP#: 13-二月 -2003 11:28:24 AM  
USERID: SCOTT  
TERMINAL: MYNETSERVER  
ACTION#: 100  
RETURNCODE: 0  
COMMENT$TEXT: Authenticated by: DATABASE; Client address:  
                ADDRESS=(PROTOCOL=tcp) (HOST=128.0.0.1) (PORT=1088)  
SPARE1: MYNETSERVER\Administrator  
PRIV$USED: 5  
-----
```

8.6 习题

- (1) Oracle 9i 的安全性体系包含哪些内容？
- (2) 什么是 Oracle 9i 的系统安全性机制和数据安全性机制？
- (3) Oracle 9i 默认的用户及口令是什么？各有什么身份？
- (4) 试通过实验熟悉创建用户、删除和修改用户的方法。
- (5) 角色有什么作用？
- (6) 试通过实验熟悉创建角色、删除和修改角色的方法。
- (7) 概要文件有什么作用？
- (8) 试通过实验熟悉创建概要文件、删除和修改概要文件的方法。
- (9) 审计有什么作用？
- (10) 试通过实验熟悉审计的使用。

第9章 编程进阶——PL/SQL

本章通过实例的方式教会读者掌握 PL/SQL 的编程基本要素，从而完成复杂的管理任务。

9.1 节从实例出发，分析了完整的 PL/SQL 程序的结构。

9.2 节介绍 PL/SQL 程序的基本语法要素，包括常量、基本数据类型、复合数据类型、表达式和函数。

9.3 节介绍了条件控制和循环控制的流程控制结构。

9.4 节介绍了事务处理的 commit、rollback 和 savepoint 等 3 个命令的使用。

9.5 节介绍了如何定义、打开和提取游标的数据，如何使用主要的属性。

9.6 节介绍无参数过程和带参数过程的创建、查询、修改和使用方法。

9.7 节介绍了序列的创建和使用方法。

9.8 节介绍如何定义异常。

9.9 节通过一个综合实例介绍如何编写 PL/SQL 程序。

9.1 PL/SQL 程序结构

9.1.1 什么是 PL/SQL 程序

前面第 4 章学习的标准化的 SQL 语言对数据库进行各种操作，每次只能执行一条语句，语句以英文的分号“;”为结束标识，这样使用起来很不方便，同时效率较低，这是因为 Oracle 数据库系统不像 VB、VC 这样的程序设计语言，它侧重于后台数据库的管理，因此提供的编程能力较弱，而结构化编程语言对数据库的支持能力又较弱，如果一些稍微复杂点的管理任务都要借助编程语言来实现的话，这对管理员来讲是很大的负担。

正是在这种需求的驱使下，从 Oracle 6 开始，Oracle 公司在标准 SQL 语言的基础上发展了自己的 PL/SQL（Procedural Language/SQL，过程化 SQL 语言）语言，将变量、控制结构、过程和函数等结构化程序设计的要素引入了 SQL 语言中，这样就能够编制比较复杂的 SQL 程序了，利用 PL/SQL 语言编写的程序也称为 PL/SQL 程序块。

PL/SQL 程序块的主要特点如下。

- ☐ 具有模块化的结构。
- ☐ 使用过程化语言控制结构。
- ☐ 能够进行错误处理。



PL/SQL 程序块只能在【SQL Plus】、【SQLPlus Worksheet】等工具支持下以解释型方式执行，不能编译成可执行文件，脱离支撑环境执行。

9.1.2 PL/SQL 实例分析

下面将为前面建立的 tempuser 用户建立一个名为 testtable 的数据表。

在该表中有 recordnumber 整数字段和 currentdate 时间型字段，编制一个 PL/SQL 程序完成向该表中自动输入 100 个记录，要求 recordnumber 字段从 1 到 100，currentdate 字段为当前系统时间。

(1) 前面建立的 tempuser 用户默认的表空间为 USERS，因此，要想使用该用户能够使用表空间建立数据方案对象，必须首先给其赋予名为“RESOURCE”的角色。

(2) 以 system 用户、SYSDBA 身份登录数据库后，在【企业管理器】中按照修改用户的步骤进行操作，直到出现如图 9.1 所示的编辑用户的【角色】选项卡。

在【可用】下拉列表框中选择“RESOURCE”，单击  按钮将其添加到【已授予】列表框中。【默认值】单元格被选中，单击  按钮。

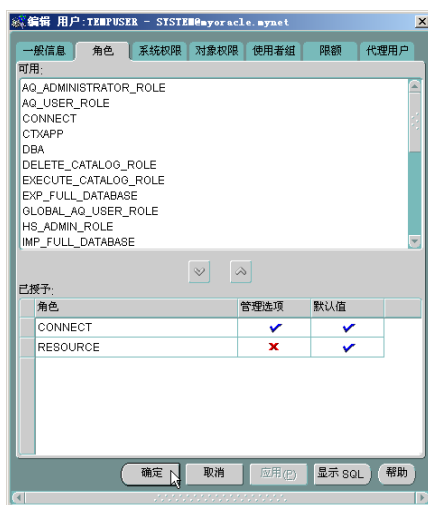


图 9.1 编辑用户的【角色】选项卡

(3) 读者也可以在【SQLPlus Worksheet】中直接执行如下 SQL 代码完成上述操作。

```
GRANT "RESOURCE" TO "TEMPUSER";
ALTER USER "TEMPUSER" DEFAULT ROLE ALL
```

【配套程序位置】：第 9 章\grantrole.sql。

(4) 按照创建数据表的操作步骤进行，直到出现如图 9.2 所示的创建表的【一般信息】选项卡。

在【名称】文本框中输入“testtable”。

在【方案】下拉列表框中选择“tempuser”。

在【表空间】下拉列表框中选择“users”。

选择【表】/【标准】单选钮。

选择【定义列】单选钮。

在【表列定义区】中输入两个数据列的定义。

完成设置后单击 **创建(C)** 按钮。

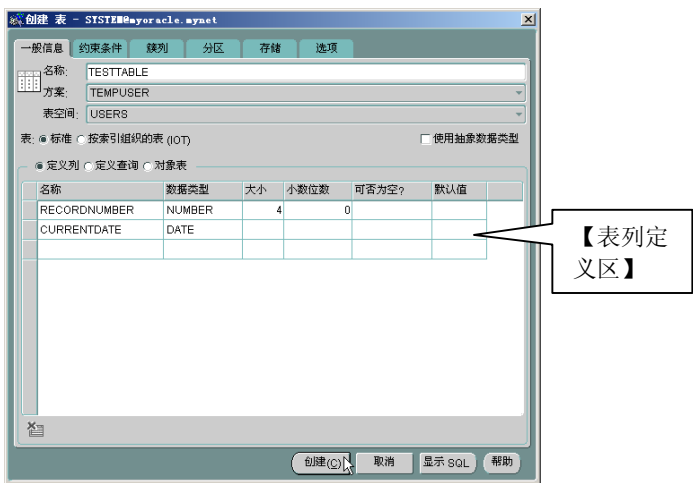


图 9.2 创建表的【一般信息】选项卡

(5) 读者也可以在【SQLPlus Worksheet】中直接执行如下 SQL 代码完成上述操作。

```
CREATE TABLE "TEMPUSER"."TESTTABLE" ("RECORDNUMBER" NUMBER(4) NOT
    NULL, "CURRENTDATE" DATE NOT NULL)
    TABLESPACE "USERS"
```

【配套程序位置】：第 9 章\createtesttable.sql。

(6) 以 tempuser 用户身份登录【SQLPlus Worksheet】，执行下列 SQL 代码完成向数据表 tempuser.testtable 中输入 100 个记录的功能。执行结果如图 9.3 所示。

```
set serveroutput on
declare
    maxrecords constant int:=100;
    i int :=1;
begin
    for i in 1..maxrecords loop
        insert into tempuser.testtable(recordnumber,currentdate)
        values(i,sysdate);
    end loop;
    dbms_output.put_line('成功录入数据!');
commit;
end;
```

【配套程序位置】：第 9 章\inserttesttable.sql。

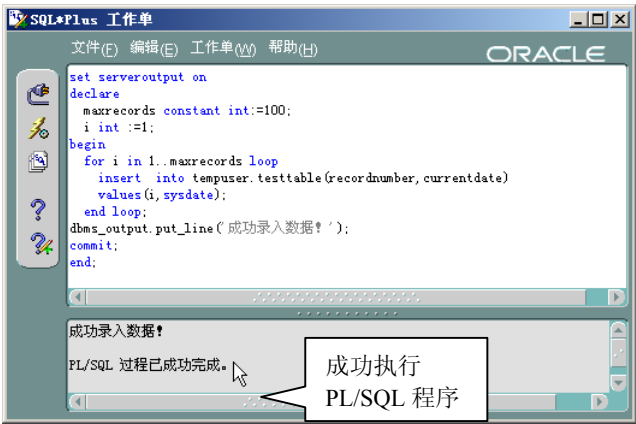


图 9.3 成功执行 PL/SQL 程序

(7) 在【SQLPlus Worksheet】中执行下列语句，查询插入的数据，结果如图 9.4 所示。

select * from tempuser.testtable;

【配套程序位置】：第 9 章\selecttesttable.sql。



图 9.4 查询执行 PL/SQL 程序后的结果

对完成上述数据插入过程的 PL/SQL 程序的分析如表 9.1 所示。

表 9.1 PL/SQL 实例代码分析

程序代码	说明
set serveroutput on	允许服务器输出
declare	定义部分标识
maxrecords constant int:=100;	定义 maxrecords 为整型常量 100
i int :=1;	定义 i 为整型值变量，初值为 1
Begin	执行部分标识

续表	
for i in 1..maxrecords loop	i 从 1 循环到 maxrecords
Insert into tempuser.testtable(recordnumber,currentdate) values (i,sysdate);	向数据表中插入数据
end loop;	结束循环
dbms_output.put_line('成功录入数据! ');	显示成功录入数据信息
commit;	提交结果
end;	结束执行



表中的 sysdate 为系统时间函数；dbms_output 为系统默认的程序包，put_line 为包中定义的方法，功能是输出信息；在 Oracle 中，所有对数据库数据的更改并没有直接操作数据库，而是放在叫工作区的内存里，只有在 commit 语句执行后，才发生永久更改。

9.1.3 PL/SQL 程序结构

结合上述实例进行分析，完整的 PL/SQL 程序结构可以分为 3 个部分。

1. 定义部分

以 Declare 为标识，在该部分中定义程序中要使用的常量、变量、游标和例外处理名称，PL/SQL 程序中使用的所有定义必须在该部分集中定义，而在高级语言里变量可以在程序执行过程中定义。

2. 执行部分

以 begin 为开始标识，以 end 为结束标识。该部分是每个 PL/SQL 程序所必备的，包含了对数据库的操作语句和各种流程控制语句。

3. 异常处理部分

该部分包含在执行部分里面，以 exception 为标识，对程序执行中产生的异常情况进行处理。一个完整的 PL/SQL 程序的总体结构如图 9.5 所示。

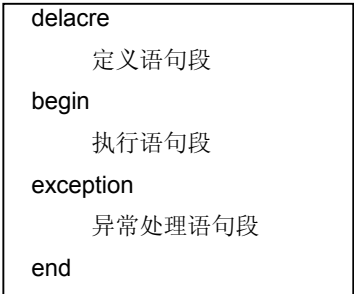


图 9.5 PL/SQL 程序的总体结构

有的程序比较简单，往往省略异常处理部分。下面开始介绍 PL/SQL 的一些基本语法要素。

9.2 基本语法要素

9.2.1 常量

1. 定义常量的语法格式

常量名 constant 类型标识符 [not null]:=值;



常量，包括后面的变量名都必须以字母开头，不能有空格，不能超过 30 个字符长度，同时不能和保留字同名，常（变）量名称不区分大小写，在字母后面可以带数字或特殊字符。括号内的 not null 为可选参数，若选用，表明该常（变）量不能为空值。

2. 实例

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 pi 的数字型常量，长度为 9。

执行结果如图 9.6 所示。

```
-----  
declare  
    pi constant number(9):=3.1415926;  
begin  
    commit;  
end;  
-----
```

【配套程序位置】：第 9 章\constantdefine.sql。

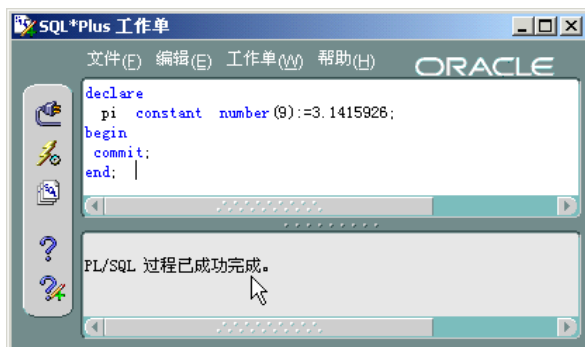


图 9.6 定义常量

9.2.2 基本数据类型变量

1. 基本数据类型

PL/SQL 中常用的基本数据类型如表 9.2 所示。

表 9.2 常见的数据基本类型

类型标识符	说明
Number	数字型
Int	整数型
Pls_integer	整数型，产生溢出时出现错误
Binary_integer	整数型，表示带符号的整数
Char	定长字符型，最大 255 个字符
Varchar2	变长字符型，最大 2000 个字符
Long	变长字符型，最长 2GB
Date	日期型
Boolean	布尔型（TRUE、FALSE、NULL 三者取一）



在 PL/SQL 中使用的数据类型和 Oracle 数据库中使用的数据类型，有的含义是完全一致的，有的是有不同的含义的。

2. 基本数据类型变量的定义方法

变量名 类型标识符 [not null]:=值;

3. 实例

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 age 的数字型变量，长度为 3，初始值为 26。执行结果如图 9.7 所示。

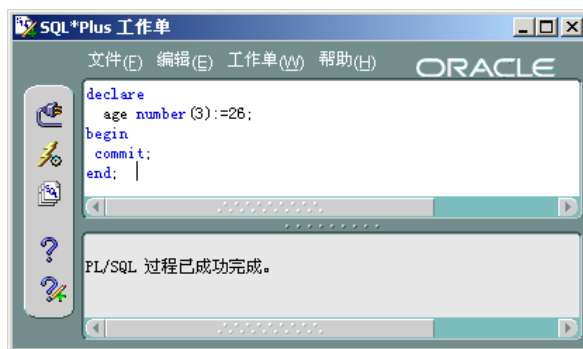


图 9.7 定义变量

```
declare
    age number(3):=26;
begin
    commit;
end;
```

【配套程序位置】：第 9 章\basicdatatypedefine.sql。

9.2.3 复合数据类型变量

下面介绍常见的几种复合数据类型变量的定义。

1. 使用%type 定义变量

为了让 PL/SQL 中变量的类型和数据表中的字段的数据类型一致，Oracle 9i 提供了%type 定义方法。这样当数据表的字段类型修改后，PL/SQL 程序中相应变量的类型也自动修改。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 mydate 的变量，其类型和 tempuser.testtable 数据表中的 currentdate 字段类型是一致的。

执行结果如图 9.8 所示。

```
Declare
    mydate tempuser.testtable.currentdate%type;
begin
    commit;
end;
```

【配套程序位置】：第 9 章\typedefine.sql。



图 9.8 用%type 定义复合变量

2. 定义记录类型变量

很多结构化程序设计语言都提供了记录类型的数据类型，在 PL/SQL 中，也支持将多个基本数据类型捆绑在一起的记录数据类型。

下面的程序代码定义了名为 `myrecord` 的记录类型，该记录类型由整数型的 `myrecordnumber` 和日期型的 `mycurrentdate` 基本类型变量组成，`srecord` 是该类型的变量，引用记录型变量的方法是“记录变量名.基本类型变量名”。

程序的执行部分从 `tempuser.testtable` 数据表中提取 `recordnumber` 字段为 68 的记录的内容，存放在 `srecord` 复合变量里，然后输出 `srecord.mycurrentdate` 的值，实际上就是数据表中相应记录的 `currentdate` 的值。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，执行结果如图 9.9 所示。

```
-----
set serveroutput on
declare
    type myrecord is record(
        myrecordnumber int,
        mycurrentdate date);
    srecord myrecord;
begin
    select * into srecord from tempuser.testtable where recordnumber=68;
    dbms_output.put_line(srecord.mycurrentdate);
end;
-----
```

【配套程序位置】：第 9 章\recordtypedefine.sql。



在 PL/SQL 程序中，`select` 语句总是和 `into` 配合使用，`into` 子句后面就是要被赋值的变量。



图 9.9 定义记录型复合变量

3. 使用%rowtype 定义变量

使用%type 可以使变量获得字段的数据类型，使用%rowtype 可以使变量获得整个记录的数据类型。比较两者定义的不同：变量名 数据表.列名%type，变量名 数据表%rowtype。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 mytable 的复合类型变量，与 testtable 数据表结构相同，执行结果如图 9.10 所示。

```

-----
Declare
    mytable testtable%rowtype;
begin
    select * into mytable
    from tempuser.testtable
    where recordnumber=88;
    dbms_output.put_line(mytable.currentdate);
end;
-----

```

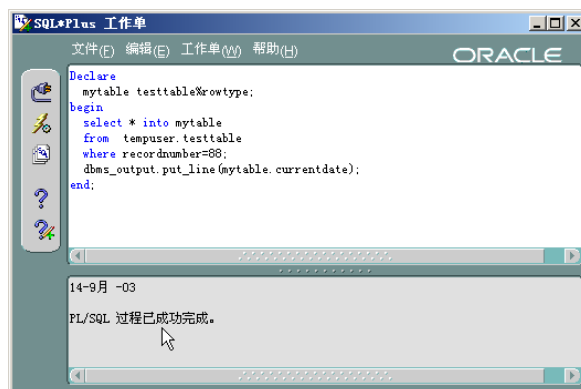


图 9.10 使用%rowtype 定义变量

【配套程序位置】：第 9 章\rowtypedefine.sql。

4. 定义一维表类型变量

表类型变量和数据表是有区别的，定义表类型变量的语法如下：

```

-----
type 表类型 is table of 类型 index by binary_integer;
表变量名 表类型;
-----

```

类型可以是前面的类型定义，index by binary_integer 子句代表以符号整数为索引，这样访问表类型变量中的数据方法就是“表变量名(索引符号整数)”。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 tabletype1 和 tabletype2 的两个一维表类型，相当于一维数组。table1 和 table2 分别是两种表类型变量。

执行结果如图 9.11 所示。

```

-----
Declare
    type tabletype1 is table of varchar2(4) index by binary_integer;
    type tabletype2 is table of tempuser.testtable.recordnumber%type index by
binary_integer;
    table1 tabletype1;
    table2 tabletype2;
begin
    table1(1):='大学';
    table1(2):='大专';
    table2(1):=88;
    table2(2):=55;
    dbms_output.put_line(table1(1)||table2(1));
    dbms_output.put_line(table1(2)||table2(2));
end;
-----

```

【配套程序位置】：第 9 章\tabletypedefine1.sql。

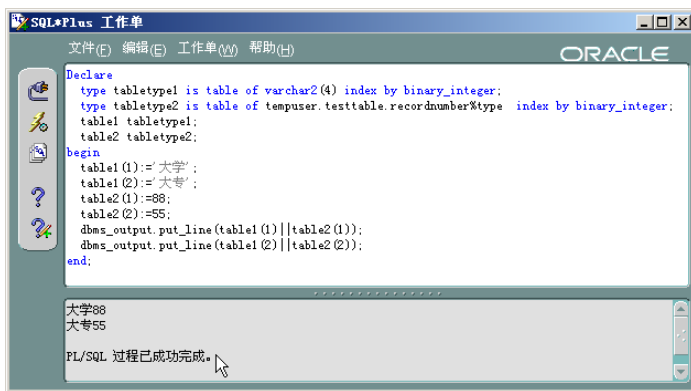


图 9.11 定义一维表类型变量



“||”是连接字符串的运算符。

5. 定义多维表类型变量

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 tabletype1 的多维表类型，相当于多维数组，table1 是多维表类型变量，将数据表 tempuser.testtable 中 recordnumber 为 60 的记录提取出来存放在 table1 中并显示。执行结果如图 9.12 所示。

```

-----
Declare

```

```

type tabletype1 is table of testtable%rowtype index by binary_integer;
table1 tabletype1;

begin
    select * into table1(60)
    from tempuser.testtable
    where recordnumber=60;
    dbms_output.put_line(table1(60).recordnumber||table1(60).currentdate);
end;

```

【配套程序位置】：第 9 章\tabletypedefine2.sql。

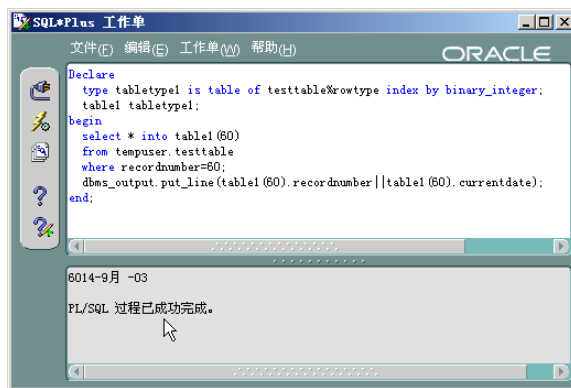


图 9.12 定义多维表类型变量



在定义好的表类型变量里，可以使用 count、delete、first、last、next、exists 和 prior 等属性进行操作，使用方法为“表变量名.属性”，返回的是数字。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 tabletype1 的一维表类型，table1 是一维表类型变量，变量中插入 3 个数据，综合使用了表变量属性。执行结果如图 9.13 所示。

```

set serveroutput on

Declare
    type tabletype1 is table of varchar2(9) index by binary_integer;
    table1 tabletype1;

begin
    table1(1):='成都市';
    table1(2):='北京市';
    table1(3):='青岛市';
    dbms_output.put_line('总记录数: '||to_char(table1.count));
    dbms_output.put_line('第一条记录: '||table1.first);
    dbms_output.put_line('最后条记录: '||table1.last);

```

```

dbms_output.put_line(' 第二条的前一条记录: '||table1.prior(2));
dbms_output.put_line(' 第二条的后一条记录: '||table1.next(2));
end;

```

【配套程序位置】: 第9章\tabletypedefine3.sql。

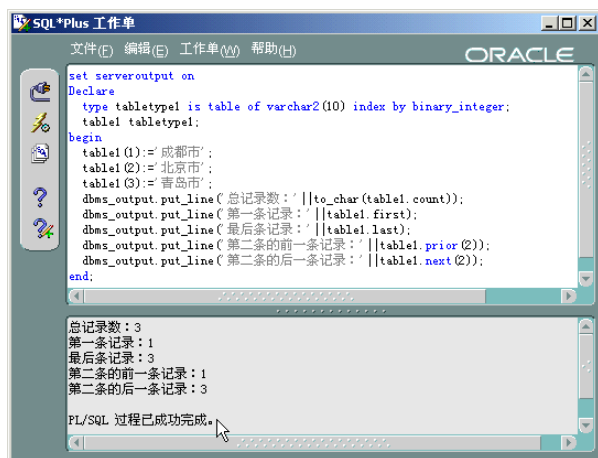


图 9.13 使用表类型变量的属性

9.2.4 表达式

变量、常量经常需要组成各种表达式来进行运算，下面介绍在 PL/SQL 中常见表达式的运算规则。

1. 数值表达式

PL/SQL 程序中的数值表达式是由数值型常数、变量、函数和算术运算符组成的，可以使用的算术运算符包括+（加法）、-（减法）、*（乘法）、/（除法）和**（乘方）等。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义了名为 result 的整数型变量，计算的是 $10+3*4-20+5**2$ 的值，理论结果应该是 27。执行结果如图 9.14 所示。

```

set serveroutput on
Declare
    result integer;
begin
    result:=10+3*4-20+5**2;
    dbms_output.put_line(' 运算结果是: '||to_char(result));
end;

```

【配套程序位置】: 第9章\datacompute.sql。

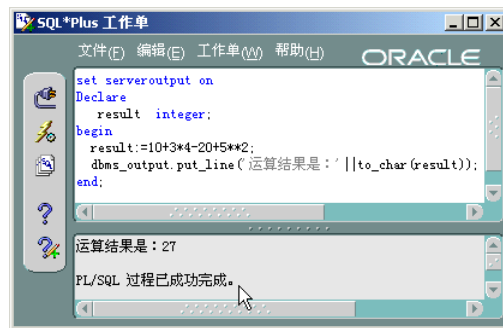


图 9.14 数值表达式运算



dbms_output.put_line 函数输出只能是字符串，因此利用 to_char 函数将数值型结果转换为字符型。

2. 字符表达式

字符表达式由字符型常数、变量、函数和字符运算符组成，唯一可以使用的字符运算符就是连接运算符“||”。

3. 关系表达式

关系表达式由字符表达式或数值表达式与关系运算符组成，可以使用的关系运算符包括以下 9 种。

- ☐ < 小于
- ☐ > 大于
- ☐ = 等于（不是赋值运算符:=）
- ☐ like 类似于
- ☐ in 在.....之中
- ☐ <= 小于等于
- ☐ >= 大于等于
- ☐ != 不等于
- ☐ between 在.....之间



关系型表达式运算符两边的表达式的数据类型必须一致。

4. 逻辑表达式

逻辑表达式由逻辑常数、变量、函数和逻辑运算符组成，常见的逻辑运算符包括以下 3 种。

- ☐ NOT: 逻辑非
- ☐ OR: 逻辑或
- ☐ AND: 逻辑与

运算的优先次序为 NOT、AND 和 OR。

9.2.5 函数

PL/SQL 程序中提供了很多函数供扩展功能，除了标准 SQL 语言的函数可以使用外，常见的数据类型转换函数有以下 3 个。

- ❑ To_char: 将其他类型数据转换为字符型。
- ❑ To_date: 将其他类型数据转换为日期型。
- ❑ To_number: 将其他类型数据转换为数值型。

以上介绍了 PL/SQL 中最基本的语法要素，下面介绍体现 PL/SQL 过程化编程思想的流程控制语句。

9.3 流程控制

PL/SQL 程序中的流程控制语句借鉴了许多高级语言的流程控制思想，但又有自己的特点。

9.3.1 条件控制

下面通过实例介绍条件控制语句的使用。

1. if..then..end if 条件控制

采用 if..then..end if 条件控制的语法结构如图 9.15 所示。

```
if 条件 then
    语句段;
end if;
```

图 9.15 if..then..end if 条件控制语法结构

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序判断两个整数变量的大小。执行结果如图 9.16 所示。

```
-----
set serveroutput on
declare
    number1 integer:=90;
    number2 integer:=60;
begin
    if number1>=number2 then
```

```

        dbms_output.put_line(' number1大于等于number2' );
    end if;
end;

```

【配套程序位置】：第 9 章\conditioncontrol1.sql。



图 9.16 if..then..end if 条件控制语句

2. if..then..else..end if 条件控制

采用 if..then..else..end if 条件控制的语法结构如图 9.17 所示。

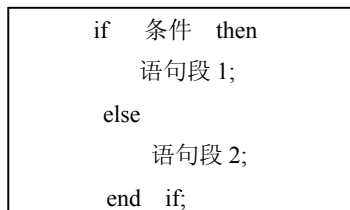


图 9.17 if..then..else..end if 条件控制语法结构

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序判断两个整数变量的大小，输出不同的结果。执行结果如图 9.18 所示。

```

set serveroutput on
declare
    number1 integer:=80;
    number2 integer:=90;
begin
    if number1>number2 then
        dbms_output.put_line(' number1大于等于number2' );
    else
        dbms_output.put_line(' number1小于number2' );
    end if;
end;

```


【配套程序位置】：第9章\conditioncontrol2.sql。

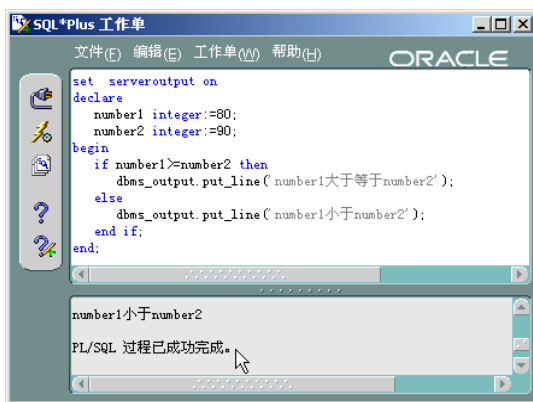


图 9.18 if..then..else..end if 条件控制语句

3. if 嵌套条件控制

采用 if 嵌套条件控制的语法结构如图 9.19 所示。

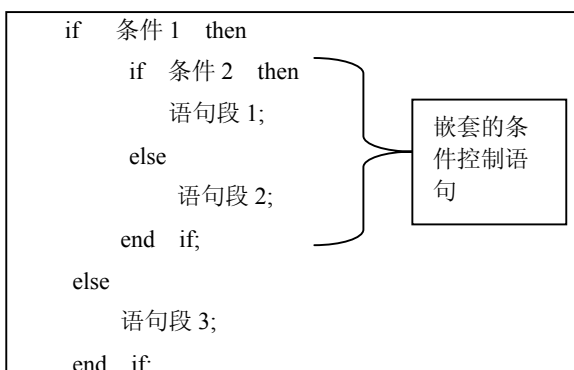


图 9.19 if 嵌套条件控制语法结构

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序判断两个整数变量的大小，输出不同的结果。

执行结果如图 9.20 所示。

```

set serveroutput on
declare
  number1 integer:=80;
  number2 integer:=90;
begin
  if number1<=number2 then
    if number1=number2 then
      dbms_output.put_line(' number1等于number2');
    end if;
  end if;
end;

```

```

else
    dbms_output.put_line(' number1小于number2');
end if;
else
    dbms_output.put_line(' number1大于number2');
end if;
end;

```

【配套程序位置】：第 9 章\conditioncontrol3.sql。



图 9.20 if 嵌套条件控制语句

9.3.2 循环控制

循环结构是按照一定逻辑条件执行一组命令，PL/SQL 中有 4 种基本循环结构，在它们基础上又可以演变出许多嵌套循环控制，这里介绍最基本的循环控制语句。

1. loop..exit..end loop 循环控制

采用 loop..exit..end loop 循环控制的语法结构如图 9.21 所示。

```

loop
    循环语句段;
    if 条件语句 then
        exit;
    else
        退出循环的处理语句段;
    end if;
end loop;

```

图 9.21 loop..exit..end loop 循环控制语法结构

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序将 number1 变量每次加 1，一直到等于 number2 为止，统计输出循环次数。

```

-----
set serveroutput on
declare
    number1 integer:=80;
    number2 integer:=90;
    i integer:=0;
begin
    loop
        number1:=number1+1;
        if number1=number2 then
            exit;
        else
            i:=i+1;
        end if;
    end loop;
    dbms_output.put_line('共循环次数:' || to_char(i));
end;
-----

```

执行结果如图 9.22 所示。



图 9.22 loop..exit..end loop 循环控制语句

【配套程序位置】：第 9 章\loopcontrol1.sql。

2. loop..exit..when..end loop 循环控制

采用 loop..exit..when..end loop 循环控制的语法结构与图 9.21 所示结构类似。

exit when 实际上就相当于

if 条件 then

exit;

end if;

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序将 number1 变量每次加 1，一直到等于 number2 为止，统计输出循环次数。

```
-----
set serveroutput on
declare
    number1 integer:=80;
    number2 integer:=90;
    i integer:=0;
begin
    loop
        number1:=number1+1;
        i:=i+1;
        exit when number1=number2;
    end loop;
    dbms_output.put_line('共循环次数:' || to_char(i));
end;
```

执行结果如图 9.23 所示。



图 9.23 loop..exit..when..end loop 循环控制语句

【配套程序位置】：第 9 章\loopcontrol2.sql。



when 循环控制结束条件比采用 if 的条件控制结束循环次数多 1 次。

3. while..loop..end loop 循环控制

采用 loop..exit..when..end loop 循环控制的语法如下。

while 条件 loop

 执行语句段;

end loop;

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序将 number1 变量每次加 1，一直到等于 number2 为止，统计输出循环次数。

```
-----
set serveroutput on
declare
    number1 integer:=80;
    number2 integer:=90;
    i integer:=0;
begin
    while number1<number2 loop
        number1:=number1+1;
        i:=i+1;
    end loop;
    dbms_output.put_line('共循环次数:' || to_char(i));
end;
-----
```

执行结果如图 9.24 所示。



图 9.24 while..loop..end loop 循环控制语句

【配套程序位置】：第9章\whilecontrol.sql。

4. for..in..loop..end 循环控制

采用 for..in..loop..end 循环控制的语法如下。

for 循环变量 in [reverse] 循环下界..循环上界 loop
循环处理语句段;

end loop;

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序,该程序通过循环变量 I 来控制 number1 增加次数,输出结果。执行结果如图 9.25 所示。

```
-----  
set serveroutput on  
declare  
    number1 integer:=80;  
    number2 integer:=90;  
    i integer:=0;  
begin  
    for i in 1..10 loop  
        number1:=number1+1;  
    end loop;  
    dbms_output.put_line('number1的值:' || to_char(number1));  
end;  
-----
```

【配套程序位置】: 第 9 章\forcontrol.sql。



图 9.25 for..in..loop..end 循环控制语句

9.4 事务处理

事务是 Oracle 9i 中进行数据库操作的基本单位,在 PL/SQL 程序中,可以使用 3 个事务处理控制命令。

9.4.1 commit 命令

commit 是事务提交命令。在 Oracle 9i 数据库中，为了保证数据的一致性，在内存中将为每个客户机建立工作区，客户机对数据库进行操作处理的事务都在工作区内完成，只有在输入 **commit** 命令后，工作区内的修改内容才写入到数据库上，称为物理写入，这样可以保证在任意的客户机没有物理提交修改以前，别的客户机读取的后台数据库中的数据是完整的、一致的，如图 9.26 所示。

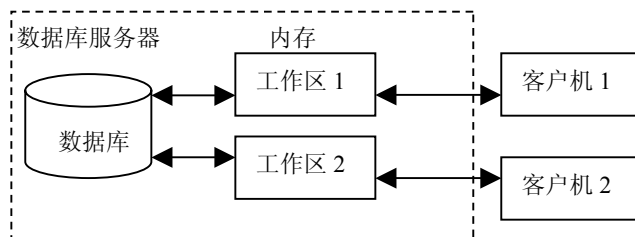


图 9.26 commit 命令示意图

在【SQLPlus Worksheet】中可以执行下列 PL/SQL 程序打开自动提交功能。这样每次执行 PL/SQL 程序都会自动进行事务提交。执行结果如图 9.27 所示。

```
-----
set auto on;
-----
```

【配套程序位置】：第 9 章\setautocommiton.sql。



图 9.27 打开自动提交功能

相应的，取消自动提交功能的 PL/SQL 程序如下。

```
-----
set auto off;
-----
```

【配套程序位置】：第 9 章\setautocommitoff.sql。

9.4.2 rollback 命令

rollback 是事务回滚命令，在尚未提交 **commit** 命令之前，如果发现 **delete**、**insert** 和 **update**

等操作需要恢复的话，可以使用 `rollback` 命令回滚到上次 `commit` 时的状态。

下面以 `delete` 命令为例演示如何回滚。

(1) 首先关闭自动提交功能。

(2) 在【SQLPlus Worksheet】中执行下列 SQL 程序，查看 `scott.emp` 数据表中的所有数据，执行结果如图 9.28 所示。

```
-----  
select * from scott.emp;  
-----
```

【配套程序位置】：第 9 章\selectemp.sql。



图 9.28 查看 `scott.emp` 表数据

(3) 在【SQLPlus Worksheet】中执行下列 SQL 程序，删除 `scott.emp` 数据表中的所有数据。执行结果如图 9.29 所示。

```
-----  
delete from scott.emp;  
-----
```

【配套程序位置】：第 9 章\deleteemp.sql。



图 9.29 删除 `scott.emp` 表数据

(4) 在【SQLPlus Worksheet】中执行下列 SQL 程序，查询删除数据表 `scott.emp` 后的结果。执行结果如图 9.30 所示。

```
-----  
select * from scott.emp;  
-----
```


【配套程序位置】：第9章\selectemp.sql。



图 9.30 查询删除 scott.emp 表数据后的结果

(5) 在【SQLPlus Worksheet】中执行以下 SQL 程序，完成事务的回滚。执行结果如图 9.31 所示。

```
-----
rollback;
-----
```

【配套程序位置】：第9章\rollbackemp.sql。



图 9.31 执行 rollback 命令

(5) 在【SQLPlus Worksheet】中执行 selectemp.sql 文件，执行结果如图 9.28 所示，表明成功回滚了事务。

9.4.3 savepoint 命令

savepoint 是保存点命令。事务通常由数条命令组成，可以将每个事务划分成若干个部分进行保存，这样每次可以回滚每个保存点，而不必回滚整个事务。语法格式如下。

创建保存点：savepoint 保存点名；

回滚保存点：rollback to 保存点名；

下面介绍如何使用保存点。

(1) 在【SQLPlus Worksheet】中执行以下 SQL 程序，程序完成向 scott.emp 数据表中插入一条记录。执行结果如图 9.32 所示。

```
-----
insert into scott.emp(empno,ename,sal) values(9000,'wang',2500);
-----
```

【配套程序位置】：第9章\insertemp.sql。



图 9.32 向 scott.emp 数据表中插入数据

(2) 在【SQLPlus Worksheet】中执行下列 SQL 代码，完成创建保存点的操作。执行结果如图 9.33 所示。

`savepoint insertpoint;`

【配套程序位置】: 第 9 章\createsavepoint.sql。

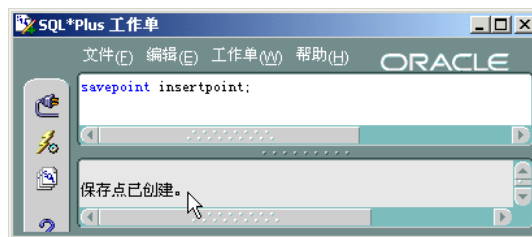


图 9.33 创建保存点

(3) 在【SQLPlus Worksheet】中可以执行其他的 SQL 语句。

(4) 在【SQLPlus Worksheet】中执行下列 SQL 代码，程序完成回滚到指定的保存点“insertpoint”。执行结果如图 9.34 所示。

`rollback to insertpoint;`

【配套程序位置】: 第 9 章\rollbacksavpoint.sql。

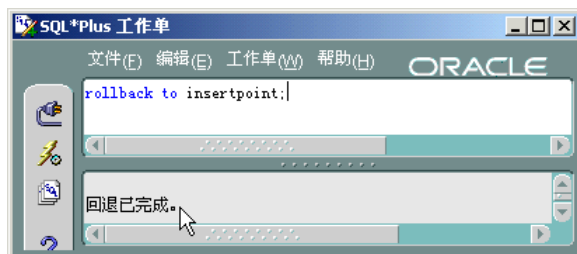


图 9.34 回滚保存点

(5) 可以执行 selectemp.sql 文件，查询回滚前后的数据。

9.5 游标

游标是从数据表中提取出来的数据，以临时表的形式存放在内存中，在游标中有一个数据指针，在初始状态下指向的是首记录，利用 `fetch` 语句可以移动该指针，从而对游标中的数据进行各种操作，然后将操作结果写回数据表中。

9.5.1 定义游标

游标作为一种数据类型，首先必须进行定义，其语法如下。

`cursor 游标名 is select 语句;`

`cursor` 是定义游标的关键词，`select` 是建立游标的数据表查询命令。

以 `scott` 用户连接数据库，在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义 `tempsal` 为与 `scott.emps` 数据表中的 `sal` 字段类型相同的变量，`mycursor` 为从 `scott.emp` 数据表中提取的 `sal` 大于 `tempsal` 的数据构成的游标。

执行结果如图 9.35 所示。

```
-----  
set serveroutput on  
declare  
    tempsal scott.emp.sal%type;  
    cursor mycursor is  
        select * from scott.emp  
        where sal>tempsal;  
begin  
    tempsal:=800;  
    open mycursor;  
end;  
-----
```

【配套程序位置】：第 9 章\cursordefine.sql。



图 9.35 定义游标

9.5.2 打开游标

要使用创建好的游标，接下来要打开游标，语法结构如下：

open 游标名;

打开游标的过程有以下两个步骤：

- (1) 将符合条件的记录送入内存。
- (2) 将指针指向第一条记录。

9.5.3 提取游标数据

要提取游标中的数据，使用 **fetch** 命令，语法形式如下。

fetch 游标名 **into** 变量名 1, 变量名 2,.....;

或

fetch 游标名 **into** 记录型变量名;

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序定义 **cursorrecord** 变量是游标 **mycursor** 的记录行变量，在游标 **mycursor** 的结果中找到 **sal** 字段大于 800 的第一个记录，显示 **deptno** 字段的内容。

执行结果如图 9.36 所示。



图 9.36 提取游标数据

```
set serveroutput on
declare
  tempsal scott.emp.sal%type;
  cursor mycursor is
    select * from scott.emp
    where sal>tempsal;
  cursorrecord mycursor%rowtype;
begin
  tempsal:=800;
```

```

open mycursor;
fetch mycursor into cursorrecord;
dbms_output.put_line(to_char(cursorrecord.deptno));
end;

```

【配套程序位置】: 第9章\cursorfetch.sql。

9.5.4 关闭游标

使用完游标后，要关闭游标，使用 close 命令，语法形式如下：

```
close 游标名;
```

9.5.5 游标的属性

游标提供的一些属性可以帮助编写 PL/SQL 程序，游标属性的使用方法为：游标名[属性]，例如 mycursor%isopen，主要的游标属性如下。

1. %isopen 属性

该属性功能是测试游标是否打开，如果没有打开游标就使用 fetch 语句将提示错误。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序利用%isopen 属性判断游标是否打开。执行结果如图 9.37 所示。

```

-----
set serveroutput on
declare
    tempsal scott.emp.sal%type;
    cursor mycursor is
        select * from scott.emp
        where sal>tempsal;
    cursorrecord mycursor%rowtype;
begin
    tempsal:=800;
    if mycursor%isopen then
        fetch mycursor into cursorrecord;
        dbms_output.put_line(to_char(cursorrecord.deptno));
    else
        dbms_output.put_line('游标没有打开!');
    end if;
end;

```

【配套程序位置】: 第9章\isopenattribute.sql。



图 9.37 使用%isopen 属性

2. %found 属性

该属性功能是测试前一个 fetch 语句是否有值，有值将返回 true，否则为 false。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序。该程序利用%found 属性判断游标是否有数据。

执行结果如图 9.38 所示。

```

-----
set serveroutput on
declare
  tempsal scott.emp.sal%type;
  cursor mycursor is
    select * from scott.emp
    where sal>tempsal;
  cursorrecord mycursor%rowtype;
begin
  tempsal:=800;
  open mycursor;
  fetch mycursor into cursorrecord;
  if mycursor%found then
    dbms_output.put_line(to_char(cursorrecord.deptno));
  else
    dbms_output.put_line('没有数据!');
  end if;
end;
-----

```

【配套程序位置】: 第 9 章\ foundattribute.sql。



图 9.38 使用%found 属性

3. %notfound 属性

该属性是%found 属性的反逻辑，常被用于退出循环。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序。该程序利用%notfound 属性判断游标是否没有数据。

执行结果如图 9.39 所示。

```

-----
set serveroutput on
declare
    tempsal scott.emp.sal%type;
    cursor mycursor is
        select * from scott.emp
        where sal>tempsal;
    cursorrecord mycursor%rowtype;
begin
    tempsal:=800;
    open mycursor;
    fetch mycursor into cursorrecord;
    if mycursor%notfound then
        dbms_output.put_line(to_char(cursorrecord.deptno));
    else
        dbms_output.put_line('发现数据!');
    end if;
end;
-----

```

【配套程序位置】：第9章\notfoundattribute.sql。



图 9.39 使用%notfound 属性

4. %rowcount 属性

该属性用于返回游标的数据行数。

在 SQLPlus Worksheet 的【代码编辑区】执行下列 PL/SQL 程序，该程序利用%rowcount 属性判断游标数据行数。

执行结果如图 9.40 所示。

```

-----
Set serveroutput on
declare
  tempsal scott.emp.sal%type;
  cursor mycursor is
    select * from scott.emp
    where sal>tempsal;
  cursorrecord mycursor%rowtype;
begin
  tempsal:=800;
  open mycursor;
  fetch mycursor into cursorrecord;
  dbms_output.put_line(to_char(mycursor%rowcount));
end;
-----

```

【配套程序位置】：第 9 章\rowcountattribute.sql。



若返回值为 0，表明游标已经打开，但没有提取出数据。



图 9.40 使用%rowcount 属性

9.6 过程

要想利用 PL/SQL 程序完成比较完整的数据库任务，需要进一步学习一些高级设计要素的内容。前面编写执行的 PL/SQL 程序，共同的特点是没有名称，只能存储为文件，然后通过执行文件的方式执行，因此称为无名块。与此对应的是在 PL/SQL 中也引入了高级程序设计的一些概念，其中最重要的就是过程。

过程就是高级程序设计语言中的模块的概念，将一些内部联系的命令组成一个个过程，通过参数在过程之间传递数据是模块化设计思想的重要内容。

9.6.1 创建过程

1. 过程的语法结构

完整的过程结构如下：

```

create or replace procedure 过程名 as
    声明语句段;
begin
    执行语句段;
exception
    异常处理语句段;
end;

```

2. 过程的特点

过程是有名称的程序块，as 关键词代替了无名块的 declare。

3. 创建过程实例

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序将创建名为 tempprocedure 的过程，create 是创建过程的标识符，replace 表示若同名过程存在将覆盖原过程。该过程定义了一个变量，其类型和 testtable 数据表中的 currentdate 字段类型相同，都是日期型，将数据表中的 recordnumber 字段为 88 的 currentdate 字段内容送入变量中，然后输出结果。

```
-----
set serveroutput on
create or replace procedure tempuser.tempprocedure as
    tempdate    tempuser.testtable.currentdate%type;
begin
    select  currentdate
    into    tempdate
    from    testtable
    where   recordnumber=88;
    dbms_output.put_line(to_char(tempdate));
end;
-----
```

【配套程序位置】：第 9 章\createprocedure.sql。

执行结果如图 9.41 所示。



图 9.41 创建过程



创建过程并不会直接输出结果，只是和创建其他数据库对象一样，是一个数据定义命令。

9.6.2 查询过程

登录【企业管理器】，在【管理目标导航树】里选择【网络】/【数据库】/【myoracle.mynet】/【方案】/【过程】/【TEMPUSER】选项，出现如图 9.42 所示的创建好的过程。

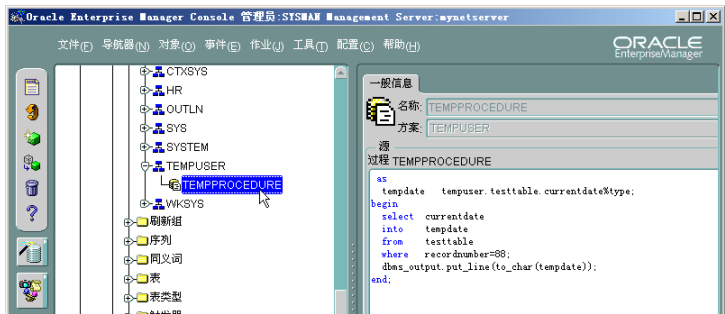


图 9.42 创建好的过程

9.6.3 修改过程

- (1) 在【SQLPlus Worksheet】的【菜单栏】选择【文件】/【打开】菜单命令，将创建过程的 createprocedure.sql 文件调出进行修改，修改完毕后重新执行创建过程。
- (2) 在【企业管理器】里选中要修改的过程，用鼠标右键单击，在出现的快捷菜单里选择【查看/编辑详细资料】选项，如图 9.43 所示。
- (3) 出现如图 9.44 所示的编辑过程的【一般信息】选项卡。在【文本编辑区】可以编辑该过程，单击 **确定** 按钮将更新该过程，单击 **编译** 按钮将编译该过程的 PL/SQL 源代码，使该过程可以在数据库中存储和执行。

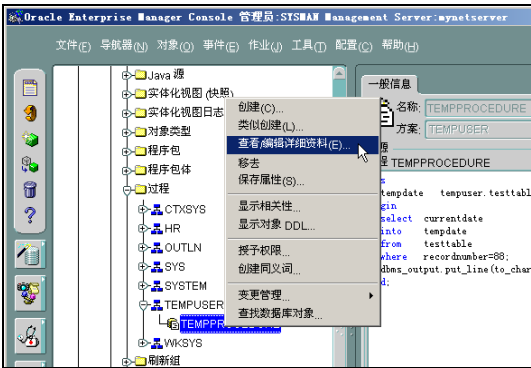


图 9.43 选择编辑过程

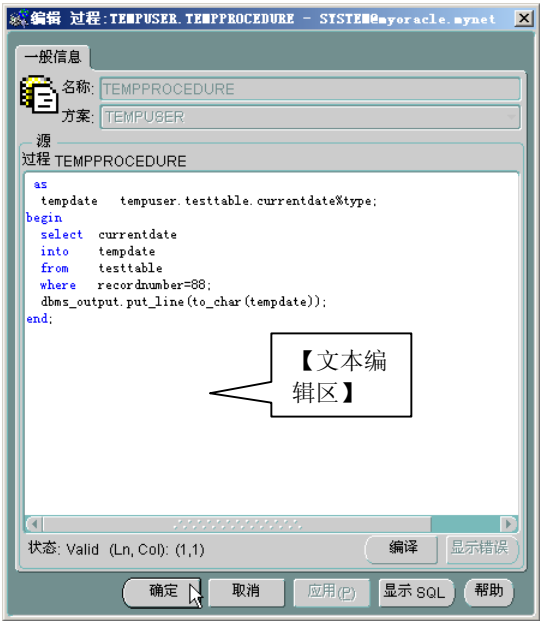


图 9.44 编辑过程的【一般信息】选项卡

9.6.4 执行过程

要执行创建的过程，必须通过主程序来调用过程。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，执行结果如图 9.45 所示。

```

set serveroutput on
begin
    tempprocedure;
end;

```

【配套程序位置】：第 9 章\executeprocedure.sql。

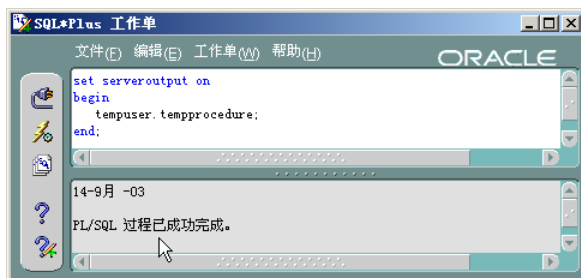


图 9.45 过程执行结果



在 Oracle 中，创建好的过程可以被任何程序调用。

9.6.5 带参数的过程

前面介绍的过程没有参数，主程序和过程没有数据的传递，下面介绍带参数的过程的设计和使用。

1. 参数类型

在 PL/SQL 过程中，可以有 3 种类型的参数。

- ☐ in 参数：读入参数，主程序向过程传递参数值。
- ☐ out 参数：读出参数，过程向主程序传递参数值。
- ☐ in out 参数：双向参数，过程与主程序双向交流数据。

2. 定义带参数的过程

在下面的 PL/SQL 程序代码中，将创建三个调用参数。

- ☐ tempdeptno: 类型为 in，与 scott.dept.deptno 的类型一致，为数值型。
- ☐ tempdname: 类型为 out，与 scott.dept.dname 的类型一致，为字符型。
- ☐ temploc: 类型为 in out，与 scott.dept.loc 类型一致，为字符型。

创建两个过程内参数。

- ☐ loc1: 与 scott.dept.loc 的类型一致，为字符型。
- ☐ dname1: 与 scott.dept.dname 的类型一致，为字符型。

该带参数的过程的功能是从数据表 scott.dept 中寻找 deptno 字段等于 tempdeptno 调用参

数值的 `dname` 和 `loc` 字段，和其他字符组合，送给两个出口参数。

以 `system` 用户名、`sysdba` 身份登录【SQLPlus Worksheet】，执行下列 PL/SQL 程序，执行结果如图 9.46 所示。

```
-----
Set serveroutput on
create or replace procedure scott.tempprocedure(
    tempdeptno in scott.dept.deptno%type,
    tempdname out scott.dept.dname%type,
    temploc in out scott.dept.loc%type)as
    loc1 scott.dept.loc%type;
    dname1 scott.dept.dname%type;
begin
    select loc into loc1
    from scott.dept
    where deptno=tempdeptno;
    select dname into dname1
    from scott.dept
    where deptno=tempdeptno;
    temploc:='地址:' || loc1;
    tempdname:='姓名' || dname1;
end;
-----
```

【配套程序位置】：第9章\createscottprocedure.sql。

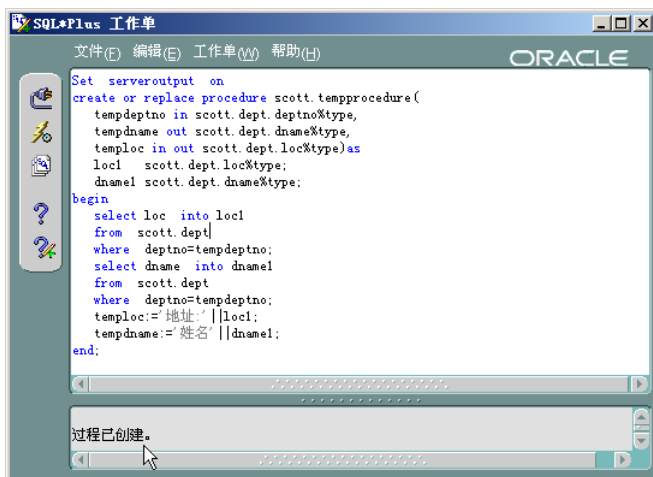


图 9.46 创建带参数的过程



调用参数分割用“,”号。

3. 使用带参数的过程

在主程序中的实际参数和过程中的形式参数的传递有很多种办法，这里推荐读者采用一对应的办法，按对应的位置传递参数。要求实际参数和形式参数在数据类型和位置排列上做到完全一致。

在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，该程序调用带参数的过程 scott.tempprocedure，实际参数为 (10,','')。

执行结果如图 9.47 所示。

```
-----  
set serveroutput on  
declare  
    myno scott.dept.deptno%type;  
    mydname scott.dept.dname%type;  
    myloc scott.dept.loc%type;  
begin  
    myno:=10;  
    mydname:='';  
    myloc:='';  
    scott.tempprocedure(myno,mydname,myloc);  
    dbms_output.put_line(myno);  
    dbms_output.put_line(mydname);  
    dbms_output.put_line(myloc);  
end;
```

读者可以尝试改变这 3 个参数，测试带参数过程

【配套程序位置】：第 9 章\executescottprocedure.sql。



图 9.47 使用带参数过程

读者可以尝试改变参数值，然后测试过程执行结果。

9.7 序列

序列是 Oracle 9i 提供的用于按照设定的规则自动产生数据的方案对象。在某些数据表的结构中，有些字段需要这种特性。比如，对于某个学生数据表的学号关键字段，用户可以希望在录入数据时，能够自动在上一个记录的学号字段上自动加 1 等。由于 Oracle 9i 提供的 16 种基本数据类型并没有这样的功能，可以通过序列方案对象来实现。

9.7.1 序列的创建

下面介绍在【企业管理器】中如何创建序列。

(1) 在【企业管理器】中选择【myoracle.mynet】/【方案】/【序列】选项，单击鼠标右键，在出现的快捷菜单里选择【创建】选项，如图 9.48 所示。

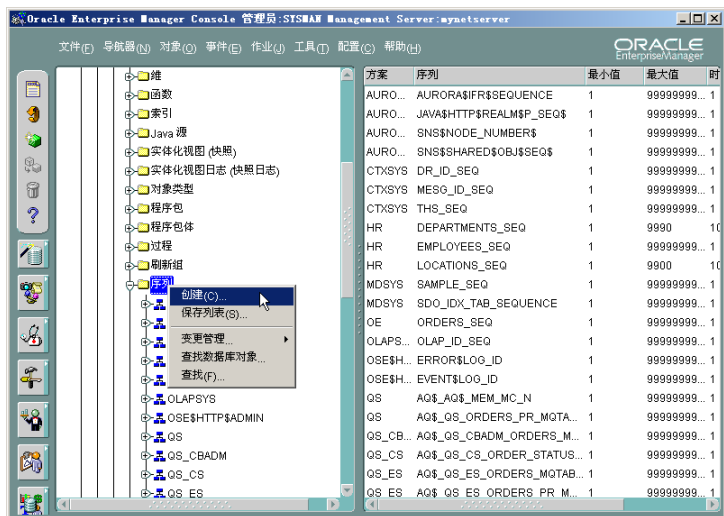


图 9.48 选择创建序列

(2) 出现如图 9.49 所示的创建序列的【一般信息】选项卡。

在【名称】文本框里输入待定义的序列的名称“TEMPSEQUENCE”。

在【方案】下拉列表框里选择序列所属的用户名“SCOTT”。

序列【类型】参数有两个选项。若选择【升序】单选钮，则表示将创建从初始值向最大值递增的序列，这是创建序列时的默认设置；若选择【降序】单选钮，则表示将创建从初始值向最小值递减的序列。

对【值】可以进行设置的参数如下。

□ 在【最小值】文本框里设置序列允许的最小值。创建序列时该字段最初为空。如果单击【创建】按钮时该字段为空，则对升序序列使用默认值 1，而对降序序列使用默认值 -1026。

□ 在【最大值】文本框里设置序列允许的最大值。创建序列时该字段最初为空。如果单击【创建】按钮后该字段为空，则将对升序序列使用默认值 1027，而对降序序列使用默认

值-1。

□ 在【时间间隔】文本框里设置递增序列递增的间隔数值（升序序列）或递减序列递减的间隔数值（降序序列）。创建序列时该字段最初为空，如果单击【创建】按钮后该字段为空，将使用默认值 1，该字段只能为正整数。


□ 在【初始值】文本框里设置序列的起始值。如果单击【创建】按钮后该字段为空，对升序序列将使用该序列默认的最小值，对降序序列将使用该序列默认的最大值。

对【选项】可以设置的参数如下。

□ 若选择【循环值】复选框，则表示指定在达到序列最小值或最大值之后，序列应继续生成值。对升序序列来说，在达到最大值后将生成最小值。对降序序列来说，在达到最小值后将生成最大值。如果未选择该复选框，序列将在达到最小值或最大值后停止生成任何值。默认情况下是未选择状态。

□ 若选择【排序值】复选框，则指定序列号要按请求次序生成，默认情况下是未选择状态。

□ 在【高速缓存】中设置由数据库预分配并存储的值的数目参数。若选择【默认值】单选按钮，则表示将设置默认值为 20，默认情况下选择此选项；若选择【无高速缓存】单选按钮，则表示指定不预分配序列值；若选择【大小】单选按钮，则表示在文本框里输入可接受的值，最小值为 2，对循环序列来说，该值必须小于循环中值的个数。如果序列能够生成的值数的上限小于高速缓存大小，则高速缓存大小将自动改换为该上限数。

完成设置后单击  按钮。


(3) 成功创建序列后，出现如图 9.50 所示界面。单击  按钮。



图 9.49 创建序列的【一般信息】选项卡 图 9.50 【成功创建序列】界面

(3) 读者也可以在【SQLPlus Worksheet】中执行下列 SQL 程序创建序列。

```
CREATE SEQUENCE "SCOTT"."TEMPSEQUENCE"
  INCREMENT BY 1 START WITH 1
  MAXVALUE 1.0E28 MINVALUE 1
  NOCYCLE CACHE 20 NOORDER
```


【配套程序位置】：第9章\createsequence.sql。

9.7.2 序列的使用

下面介绍在向数据表中插入数据时如何使用序列。

(1) 首先为实例建立一个数据表“SCOTT.SEQUENCE_TABLE”，为简化起见，该数据表仅包含一个类型为“NUMBER”的数据列“NO”。

在如图 9.51 所示的创建表的【一般信息】选项卡中进行如下设置。

在【名称】文本框中输入“SEQUENCE_TABLE”。

在【方案】下拉列表框中选择“SCOTT”。

在【表空间】下拉列表框中选择“USERS”。


在【名称】单元格中输入“NO”，在【数据类型】下拉列表框单元格中选择“NUMBER”。完成设置后单击  按钮。



图 9.51 创建表的【一般信息】选项卡

(2) 读者也可以在【SQLPlus Worksheet】中执行下列 SQL 代码创建数据表“SCOTT.SEQUENCE_TABLE”。

```
CREATE TABLE "SCOTT"."SEQUENCE_TABLE" ("NO" NUMBER(10) NOT NULL)
TABLESPACE "USERS"
```

【配套程序位置】：第9章\createsequencetable.sql。

(3) 在插入新的记录时，使用刚创建的“TEMPSEQUENCE”序列来自动产生“NO”数据列的值。在【SQLPlus Worksheet】里执行下面的 SQL 代码，执行的结果如图 9.52 所示。

```
INSERT INTO SCOTT.SEQUENCE_TABLE(NO)
VALUES (SCOTT.TEMPSEQUENCE.NEXTVAL);
```

【配套程序位置】：第 9 章\insertsequencetable.sql。

“SCOTT.TEMPSEQUENCE.NEXTVAL” 表分配下一个惟一的、可用的序列号。

执行“SCOTT.TEMPSEQUENCE.NEXTVAL”后，可以使用“SCOTT.TEMPSEQUENCE.CURRVAL”来标识上一个已经存储的序列值。

(4) 在【SQLPlus Worksheet】中可以执行查询数据表“SCOTT.SEQUENCE_TABLE”数据的语句。执行结果如图 9.53 所示，表明序列“SCOTT.SEQUENCE”产生的值已经成功录入数据表中。

```
select * from scott.sequence_table;
```

【配套程序位置】：第 9 章\selectsequencetable.sql。

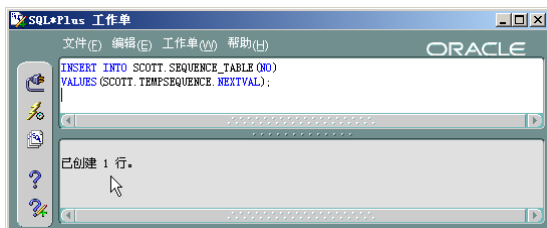


图 9.52 利用序列产生数据

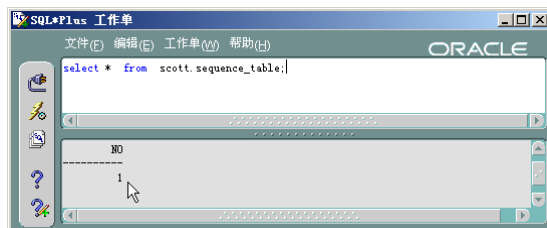


图 9.53 查询序列数据的结果

9.7.3 序列的删除

下面介绍在【企业管理器】中如何删除序列。

(1) 在创建好的序列“SEQUENCE1”上用单击鼠标右键，在出现的快捷菜单里选择【移去】选项，如图 9.54 所示。

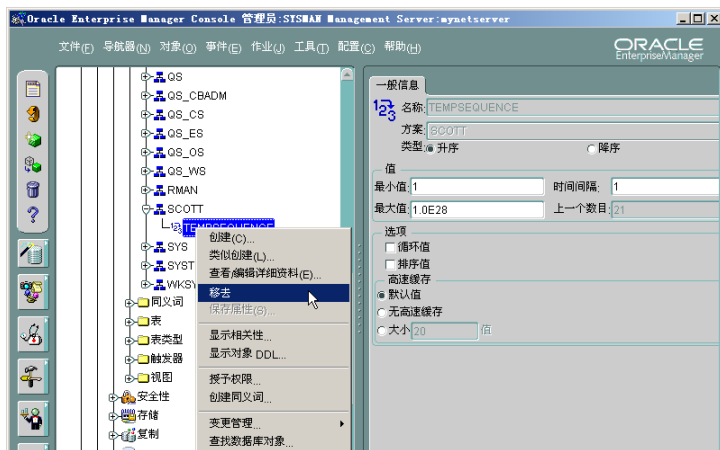


图 9.54 选择删除序列


(2) 出现如图 9.55 所示的【删除序列确认】界面，单击  按钮。



图 9.55 【删除序列确认】界面

9.8 异常处理

在设计 PL/SQL 程序时，经常会发生这样或那样的错误，异常处理就是针对错误进行处理的程序段，Oracle 9i 中的异常处理分为系统预定义异常处理和自定义异常处理两部分。

9.8.1 系统预定义异常处理

系统预定义异常处理是针对 PL/SQL 程序编译、执行过程中发生的问题进行处理的程序。下列代码为正确代码，在【SQLPlus Worksheet】中能够顺利执行。

```
-----
set serveroutput on
declare
    tempno integer:=90;
begin
    tempno:=tempno+1;
end;
-----
```

【配套程序位置】：第 9 章\correctplsql.sql。

下列代码为错误代码，在【SQLPlus Worksheet】中的执行结果如图 9.56 所示。

```
-----
set serveroutput on
declare
    tempno integer:=100;
begin
    tempno=tempno+1;
end;
-----
```

正确代码应该是：tempno:=tempno+1;

【配套程序位置】：第 9 章\wrongplsql.sql。

由于代码有错误，因此将激活系统预定义的异常处理，并得出如下提示信息。



图 9.56 调用系统异常处理

Oracle 9i 提供了很多异常处理，读者可以尝试修改可以正常运行的程序，并执行修改后的程序，就可以发现调用了哪些异常处理，下面着重介绍如何自定义异常处理。

9.8.2 自定义异常处理

1. 定义异常处理

定义异常处理的语法如下：

```
declare
    异常名 exception;
```

2. 触发异常处理

触发异常处理的语法如下：

```
raise 异常名;
```

3. 处理异常

触发异常处理后，可以定义异常处理部分，语法如下：

```
Exception
When 异常名 1 then
    异常处理语句段 1;
When 异常名 2 then
    异常处理语句段 2;
```

4. 实例

下面的 PL/SQL 程序包含了完整的异常处理定义、触发、处理的过程。定义名为 salaryerror 的异常，在 scott.emp 数据表中查找 empno=7566 的记录，将其值放入变量 tempsal 中，判断 tempsal 值若不在 900 和 2600 之间，说明该员工的薪水有问题，将激活异常处理，提示信息。

在【SQLPlus Worksheet】中执行下列 PL/SQL 代码，执行结果如图 9.57 所示。

```

set serveroutput on
declare
    salaryerror exception;
    tempsal scott.emp.sal%type;
begin
    select sal into tempsal
    from scott.emp
    where empno=7566;
    if tempsal<900 or tempsal>2600 then
        raise salaryerror;
    end if;
exception
    when salaryerror then
        dbms_output.put_line('薪水超出范围');
end;

```

【配套程序位置】：第9章\exceptiondefine.sql。



图 9.57 自定义异常

9.9 综合实例

本节将在前面学习的 PL/SQL 基础上分析一个较复杂的实例，以教会读者编写完整的 PL/SQL 程序的方法。

9.9.1 实例设计

1. 功能设计

某高校开发的研究生招生系统，要求设计 PL/SQL 程序对考生的成绩数据进行处理，处理的逻辑是根据每门专业课的最低分数线和总分的最低分数线自动将考生归类为录取考生、调剂考生和落选考生。

为此设计两个数据表，graduate 数据表存放考生成绩，result 数据表存放处理结果，PL/SQL 程序完成的功能就是将 graduate 数据表中的数据逐行扫描，根据分数线进行判断，计算出各科总分，在 result 数据表中将标志字段自动添加上“录取”或“落选”。

2. 数据表设计

Graduate 数据表结构如表 9.3 所示。

表 9.3 graduate 数据表结构

字段名称	类型	宽度	小数位数	说明
BH	NUMBER	9		考生编号
XM	VARCHAR2	9		考生姓名
LB	VARCHAR2	9		报考类别
YINGYU	NUMBER	4	1	英语成绩
ZHENGZHI	NUMBER	4	1	政治成绩
ZHUANYE1	NUMBER	4	1	专业课 1
ZHUANYE2	NUMBER	4	1	专业课 2
ZHUANYE3	NUMBER	4	1	专业课 3

result 数据表结构如表 9.4 所示。

表 9.4 result 数据表结构

字段名称	类型	宽度	小数位数	说明
BH	NUMBER	9		考生编号
XM	VARCHAR2	9		考生姓名
LB	VARCHAR2	9		报考类别
YINGYU	NUMBER	4	1	英语成绩
ZHENGZHI	NUMBER	4	1	政治成绩
ZHUANYE1	NUMBER	4	1	专业课 1
ZHUANYE2	NUMBER	4	1	专业课 2
ZHUANYE3	NUMBER	4	1	专业课 3

续表				
字段名称	类型	宽度	小数位数	说明
TOTALSCORE	NUMBER	5	1	总分
FLAG	VARCHAR2	4		标志

9.9.2 创建数据表

为了简化起见，实例的两个数据表都建立在默认的 scott 用户下，这样读者可以完全模拟实例一致的环境进行学习。

1. 创建 graduate 数据表

在图 9.58 所示的创建表的【一般信息】选项卡中按照下列步骤进行配置。

在【名称】文本框中输入“GRADUATE”。

在【方案】下拉列表框中选择“SCOTT”。

在【表空间】下拉列表框中选择“USERS”。

在【表结构定义区】按照图所示进行设置。


完成设置后单击  按钮。



图 9.58 创建表的【一般信息】选项卡

读者也可以在【SQLPlus Worksheet】中执行下列 SQL 代码，直接创建数据表 SCOTT.GRADUATE。

```
CREATE TABLE "SCOTT"."GRADUATE" (  
    "BH"          NUMBER(10)          NOT NULL,  
    "XM"          VARCHAR2(10)        NOT NULL,  
    "LB"          VARCHAR2(10)        NOT NULL,
```

```

"YINGYU"    NUMBER(4, 1)    NOT NULL,
"ZHENGZHI"  NUMBER(4, 1)    NOT NULL,
"ZHUANYE1"  NUMBER(4, 1)    NOT NULL,
"ZHUANYE2"  NUMBER(4, 1)    NOT NULL,
"ZHUANYE3"  NUMBER(4, 1)    NOT NULL)

```

TABLESPACE "USERS"

【配套程序位置】：第 9 章\creategraduate.sql。

2. 创建 result 数据表

在图 9.59 所示的创建表的【一般信息】选项卡中按照下列步骤进行配置。

在【名称】文本框中输入“RESULT”。

在【方案】下拉列表框中选择“SCOTT”。

在【表空间】下拉列表框中选择“USERS”。

在【表结构定义区】按照图所示进行设置。


完成设置后单击  按钮。



图 9.59 创建表的【一般信息】选项卡

读者也可以在【SQLPlus Worksheet】中执行下列 SQL 代码，直接创建数据表 SCOTT.RESULT。

```

CREATE TABLE "SCOTT"."RESULT" (
    "BH"          NUMBER(10)      NOT NULL,
    "XM"          VARCHAR2(10)    NOT NULL,
    "LB"          VARCHAR2(10)    NOT NULL,
    "YINGYU"      NUMBER(4, 1)    NOT NULL,

```



```

"ZHENGZHI" NUMBER(4, 1) NOT NULL,
"ZHUANYE1" NUMBER(4, 1) NOT NULL,
"ZHUANYE2" NUMBER(4, 1) NOT NULL,
"ZHUANYE3" NUMBER(4, 1) NOT NULL,
"TOTALSCORE" NUMBER(5, 1) NOT NULL,
"FLAG" VARCHAR2(4) NOT NULL)
TABLESPACE "USERS"

```

【配套程序位置】：第 9 章\createresult.sql。

9.9.3 录入数据

利用【表数据编辑器】向 SCOTT.GRADUATE 数据表录入下列实验数据。如图 9.60 所示。

图 9.60 向 graduate 数据表录入测试数据

读者也可以在【SQLPlus Worksheet】中执行下列 SQL 代码，直接向数据表 SCOTT.GRADUATE 中录入测试数据。

```

INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3")
VALUES (2003080520, '张三丰', '硕士', 55, 56, 67, 78, 89);
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3")
VALUES (2003060555, '张翠山', '硕士', 66, 78, 78, 89, 92);
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3")
VALUES (2003056066, '张无忌', '硕士', 76, 67, 89, 90, 66);
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3")
VALUES (2003010989, '赵敏', '硕士', 45, 59, 74, 66, 56);
INSERT INTO "SCOTT"."GRADUATE"

```

```

("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3" )
VALUES (2003050677, '周芷若', '硕士', 77, 67, 72, 87, 66 );
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3" )
VALUES (2003869401, '小昭', '硕士', 56, 67, 56, 64, 34 );
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3" ) VALUES
(2003340987, '阿离', '硕士', 68, 93, 64, 80, 56 );
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3" )
VALUES (2003056709, '宋远桥', '硕士', 90, 68, 81, 61, 67 );
INSERT INTO "SCOTT"."GRADUATE"
("BH", "XM", "LB", "YINGYU", "ZHENGZHI", "ZHUANYE1", "ZHUANYE2", "ZHUANYE3" ) VALUES
(2003100894, '殷素素', '硕士', 69, 73, 62, 70, 75 );

```

【配套程序位置】：第 9 章\insertgraduate.sql。

9.9.4 程序设计

1. 创建处理过程 scott.graduateprocess

在【SQLPlus Worksheet】中执行下列 PL/SQL 代码，创建处理过程 scott.graduateprocess。执行结果如图 9.61 所示。



图 9.61 创建处理过程 scott.graduateprocess

```
/*定义过程参数*/
```

```
create or replace procedure scott.graduateprocess(
    tempzhengzhi in scott.graduate.zhengzhi%type,
```

```

    tempyingyu in scott.graduate.yingyu%type,
    tempzhuanyel in scott.graduate.zhuanyel%type,
    tempzhuanye2 in scott.graduate.zhuanye2%type,
    tempzhuanye3 in scott.graduate.zhuanye3%type,
    temptotalscore in scott.result.totalscore%type) as
/*定义graduaterecord为记录型变量，临时存放通过游标从graduate数据表中提取的记录*/
    graduaterecord scott.graduate%rowtype;
/*定义graduatetotalscore为数值型变量，统计总分*/
    graduatetotalscore scott.result.totalscore%type;
/*定义graduateflag为字符型变量，根据结果放入“落选”或“录取”，然后写入数据表result*/
    graduateflag varchar2(4);
/*定义游标graduatecursor，存放的是所有的graduate数据表中的记录*/
    cursor graduatecursor is
        select * from scott.graduate;
/*定义异常处理errormessage*/
    errormessage exception;
/*开始执行*/
begin
    /*打开游标*/
    open graduatecursor;
/*如果游标没有数据，激活异常处理*/
    if graduatecursor%notfound then
        raise errormessage;
    end if;
/*游标有数据，指针指向第一条记录，每执行fetch命令，就自动下移，循环执行到记录提取完毕为止
*/
    loop
        fetch graduatecursor into graduaterecord;
/*计算总分*/
        graduatetotalscore:=graduaterecord.yingyu+graduaterecord.zhengzhi+graduaterecord.zhuanye
1+graduaterecord.zhuanye2+graduaterecord.zhuanye3;
/*判断单科和总分是否满足录取要求，若满足，graduateflag变量值为“录取”，否则为“落选”*/
    if (graduaterecord.yingyu>=tempyingyu and
        graduaterecord.zhengzhi>=tempzhengzhi and
        graduaterecord.zhuanyel>=tempzhuanyel and
        graduaterecord.zhuanye2>=tempzhuanye2 and
        graduaterecord.zhuanye3>=tempzhuanye3 and
        graduatetotalscore>=temptotalscore) then
        graduateflag:='录取';
    else

```

```

        graduateflag:='落选';
    end if;
    /*当游标数据提取完毕后，退出循环*/
    exit when graduatecursor%notfound;
    /*向结果数据表result中插入处理后的数据*/
    insert into
scott.result(bh, xm, lb, zhengzhi, yingyu, zhuanyel, zhuanye2, zhuanye3, totalscore, flag)
values(graduaterecord.bh, graduaterecord.xm, graduaterecord.lb, graduaterecord.zhengzhi, graduate
record.yingyu, graduaterecord.zhuanyel, graduaterecord.zhuanye2, graduaterecord.zhuanye3, graduat
etotalscore, graduateflag);
    end loop;
    /*关闭游标*/
    close graduatecursor;
    /*提交结果*/
    commit;
    /*异常处理，提示错误信息*/
    exception
    when errormessage then
        dbms_output.put_line('无法打开数据表');
    /*程序执行结束*/
end;

```

【配套程序位置】：第9章\creategraduateprocess.sql。

2. 主程序 mainprocess 设计

主程序调用名为 graduateprocess 的过程来完成处理，代码设计如下：

```

-----
set serveroutput on
/*定义6个入口变量，分别对应graduate数据表中的专业课和总分分数线*/
declare
    score1 number(4,1);
    score2 number(4,1);
    score3 number(4,1);
    score4 number(4,1);
    score5 number(4,1);
    scoretotal number(5,1);
    /*将分数线赋值，在这里修改各值就代表不同的分数线*/
begin
    score1:=50;

```

```

score2:=56;
score3:=60;
score4:=62;
score5:=64;
scoretot:=325;
/*调用处理过程*/
scott.graduateprocess(score1, score2, score3, score4, score5, scoretot);
end;

```

【配套程序位置】：第9章\mainprocess.sql。

在上述过程设计中，综合使用了异常处理、游标、变量等 PL/SQL 程序设计要素，通过主程序向带参数执行的过程传递参数。

9.9.5 执行结果

1. 第一组执行结果

(1) 将 mainprocess.sql 文件稍做改动，在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，执行结果如图 9.62 所示。

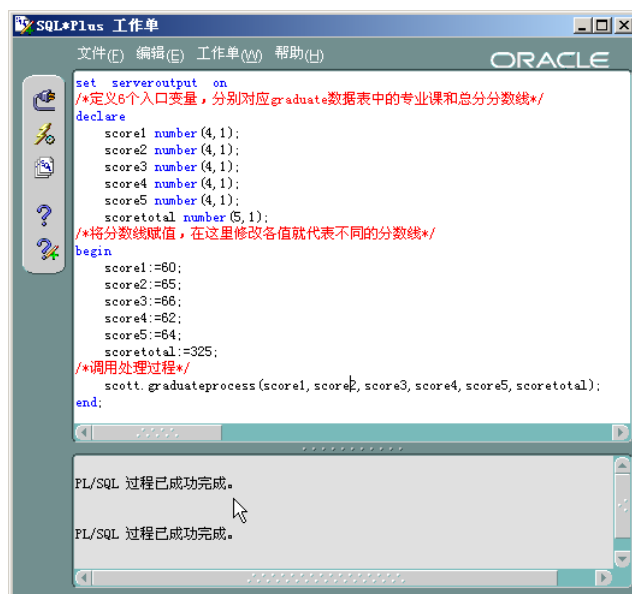


图 9.62 进行第一组测试

```

set serveroutput on
declare
  score1 number(4,1);

```

```

score2 number(4,1);
score3 number(4,1);
score4 number(4,1);
score5 number(4,1);
scoretototal number(5,1);

begin
score1:=60;
score2:=65;
score3:=66;
score4:=62;
score5:=64;
scoretototal:=325;


scott.graduateprocess(score1,score2,score3,score4,score5,scoretototal);

end;

```

【配套程序位置】：第 9 章\mainprocess1.sql。

(2) 利用【表数据编辑器】查看 scott.result 数据表的结果，如图 9.63 所示。



BH	XM	LB	YIN...	ZHE...	ZHU...	ZHU...	ZH...	TO...	FLAG
2003080520	张三丰	硕士	55	56	67	78	89	345	落选
2003060555	张翠山	硕士	66	78	78	89	92	403	录取
2003056066	张无忌	硕士	76	67	89	90	66	388	录取
2003010989	赵敏	硕士	45	59	74	66	56	300	落选
2003050677	周芷若	硕士	77	67	72	87	66	369	录取
2003869401	小昭	硕士	56	67	56	64	34	277	落选
2003340987	阿离	硕士	68	93	64	80	56	361	落选
2003056709	宋远桥	硕士	90	68	81	61	67	367	落选
2003100894	殷素素	硕士	69	73	62	70	75	349	落选

图 9.63 第一组参数执行结果

2. 第二组执行结果

(1) 将 mainprocess.sql 文件稍做改动，在【SQLPlus Worksheet】中执行下列 PL/SQL 程序，执行结果如图 9.64 所示。

```

set serveroutput on
declare
score1 number(4,1);
score2 number(4,1);
score3 number(4,1);
score4 number(4,1);
score5 number(4,1);

```

```

scoretotal number(5,1);
begin
    score1:=65;
    score2:=64;
    score3:=62;
    score4:=70;
    score5:=65;
    scoretotal:=335;
    scott.graduateprocess(score1,score2,score3,score4,score5,scoretotal);
end;

```

【配套程序位置】：第 9 章\mainprocess2.sql。

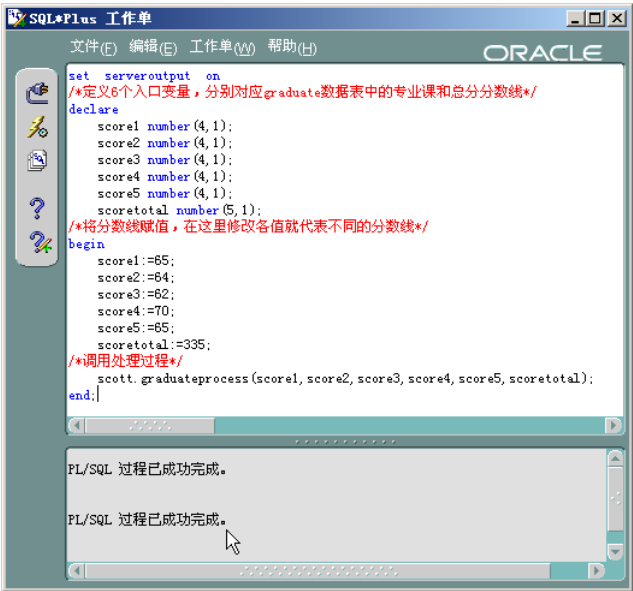


图 9.64 进行第二组测试

(2) 利用【表数据编辑器】查看 scott.result 数据表的结果，如图 9.65 所示。

BH	XM	LB	YIN...	ZH...	ZH...	ZH...	ZH...	TOT...	FLAG
2003080520	张三丰	硕士	55	56	67	78	89	345	落选
2003060555	张翠山	硕士	66	78	78	89	92	403	录取
2003056066	张无忌	硕士	76	67	89	90	66	388	录取
2003010989	赵敏	硕士	45	59	74	66	56	300	落选
2003050677	周芷若	硕士	77	67	72	87	66	369	录取
2003889401	小昭	硕士	56	67	56	64	34	277	落选
2003340987	阿离	硕士	68	93	64	80	56	361	落选
2003056709	宋远桥	硕士	90	68	81	61	67	367	落选
2003100894	殷素素	硕士	69	73	62	70	75	349	录取

图 9.65 第一组参数执行结果

综合运用 PL/SQL 的设计要素，就可以设计出很多复杂的处理程序，这也是 DBA 的一项重要任务。

9.10 习题

- (1) PL/SQL 程序的主要特点是什么？
- (2) 如何定义常量？
- (3) PL/SQL 中支持哪些基本的数据类型变量？试编写测试 PL/SQL 基本数据类型变量的实验代码。
- (4) 如何定义常用的复合类型数据变量？
- (5) PL/SQL 程序中有哪些常见的表达式？
- (6) 什么是过程？过程的特点是什么？
- (7) 什么是序列？如何使用序列？
- (8) 如何自定义异常处理？在 PL/SQL 程序中为什么要定义异常处理？
- (9) 试编写一个完整的 PL/SQL 程序，自行设计数据表结构。要求在 PL/SQL 程序中综合运用序列、过程、异常处理、表达式、函数、常量、变量、流程控制和事务处理的语句。写出实验结果和代码。

第 10 章 系统开发——VB+Oracle 9i

本章介绍如何利用中文版 Visual Basic 6.0（简称 VB）为前端开发工具，后台数据库为 Oracle 9i 进行应用系统的开发。

10.1 如何构建开发和运行的环境

10.1.1 如何选择前台开发工具

10.1.2 如何构建开发和运行环境

一个 VB+Oracle 9i 的数据库应用从逻辑上看有 3 个组成部分。如图 10.1 所示。

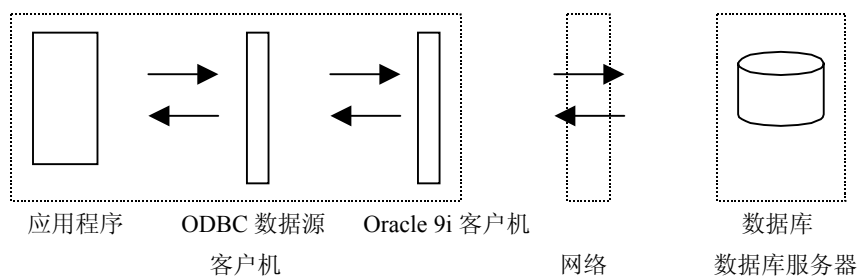


图 10.1 VB 数据库应用的结构

1. 构建开发和运行客户机

在开发客户机上进行以下工作。

- (1) 安装 VB。
- (2) 安装 Oracle 9i 客户机，最好是管理员，这样可以及时管理后台数据库服务器上的数据。

(3) 测试 Oracle 9i 的客户机是否能够正常连通数据库服务器。

在运行客户机上进行如下工作。

- (1) 安装用 VB 开发好的应用程序。
- (2) 安装 Oracle 9i 客户机，最好是连接，这样终端用户不能直接操作数据库。
- (3) 测试 Oracle 9i 的客户机是否能够正常连通数据库服务器。

2. 保证网络畅通

确保网络 TCP/IP 协议能够畅通。

3. 构建数据库服务器

2.1 节已经详细做了介绍。

10.1.3 环境构建实例

1.检查数据库服务器工作情况

2.安装 Oracle 9i 客户机

图 10.2 所示为 Oracle 9i 客户机安装步骤中的【安装类型】界面。

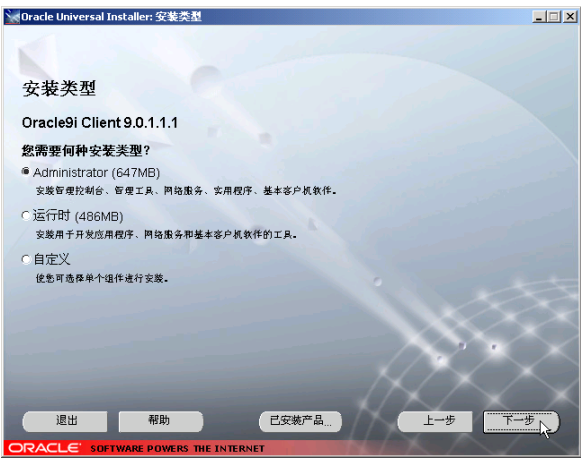


图 10.2 【安装类型】界面

3.安装 VB

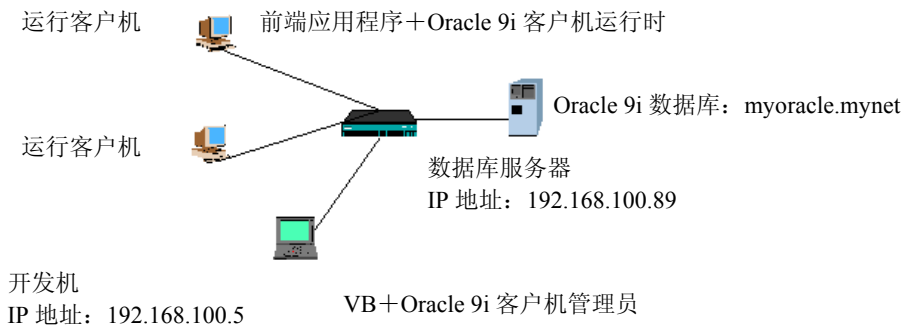


图 10.3 VB+Oracle 9i 数据库应用开发环境

10.2 VB 程序如何连接数据库

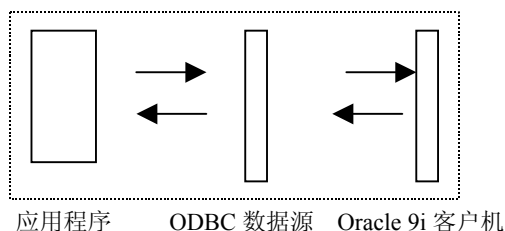


图 10.4 运行客户机的结构

10.2.1 ODBC 数据源

1. 什么是 ODBC

ODBC (Open Database Connectivity, 开放式数据库互联), 是目前国际上通用的数据库访问标准。

2. 什么是 ODBC 数据源

【ODBC 数据源】就是计算机上的 ODBC 配置和管理的工具, 利用这个工具, 用户就可以定制使用 ODBC 来连接数据库。

10.2.2 Oracle 9i 客户机

10.2.3 VB 程序连接数据库的过程分析

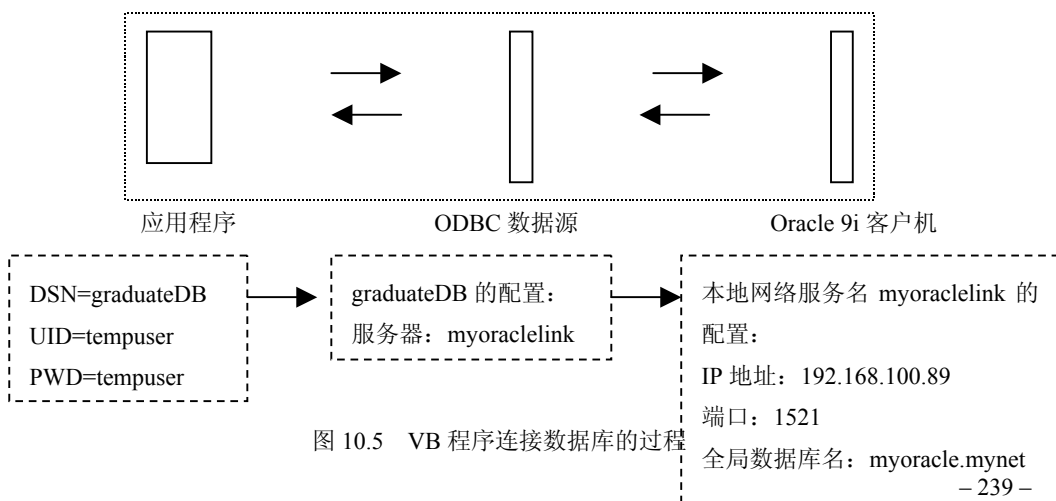


图 10.5 VB 程序连接数据库的过程

1. VB 程序指定 DSN 名称
2. 【ODBC 数据源】解析 DSN 名称
3. Oracle 9i 客户机解析数据库服务器名称
4. 数据库服务器验证 DSN 中的用户名和密码

10.2.4 如何建立 ODBC 数据源

- (1) 如图 10.6 所示的 ODBC 数据源管理器的【用户 DSN】选项卡。
- (2) 切换到如图 10.7 所示的 ODBC 数据源管理器的【系统 DSN】选项卡。



图 10.6 【用户 DSN】选项卡



图 10.7 【系统 DSN】选项卡

- (3) 出现如图 10.8 所示的【创建新数据源】界面。



图 10.8 【创建新数据源】界面

- (4) 出现如图 10.9 所示的【Microsoft ODBC for Oracle Setup】界面。



图 10.9 【Microsoft ODBC for Oracle Setup】界面

(5) 成功建立系统 DSN 名称后的【系统 DSN】选项卡如图 10.10 所示。

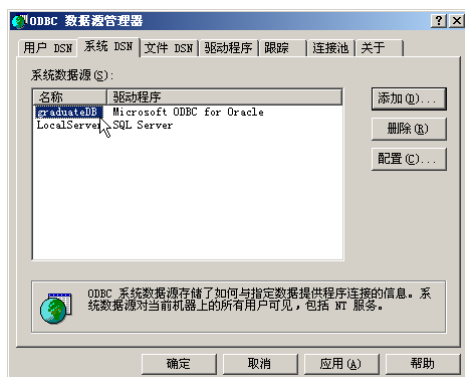


图 10.10 成功建立的 ODBC 数据源

10.3 VB 程序如何访问数据库

10.3.1 通过【Data】控件访问数据库

- (1) 启动 VB。
- (2) 出现如图 10.12 所示的新建工程的【新建】选项卡。

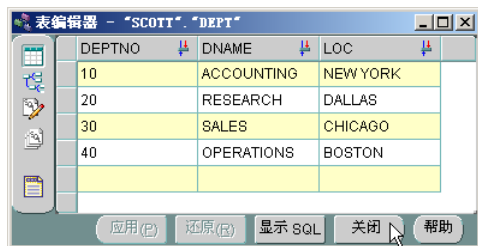


图 10.11 scott.dept 数据表的数据



图 10.12 新建工程的【新建】选项卡

(3) 出现如图 10.13 所示的 VB 主界面。

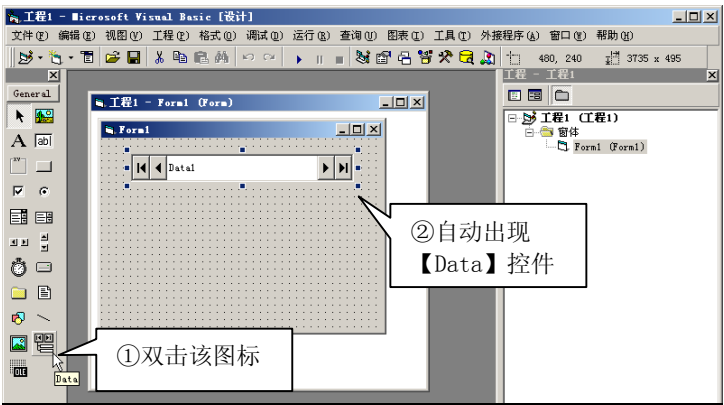


图 10.13 VB 主界面

(4) 如图 10.14 所示。



图 10.14 选择查看 data1 控件的属性窗口

(5) 出现如图 10.15 所示的【属性窗口】界面。

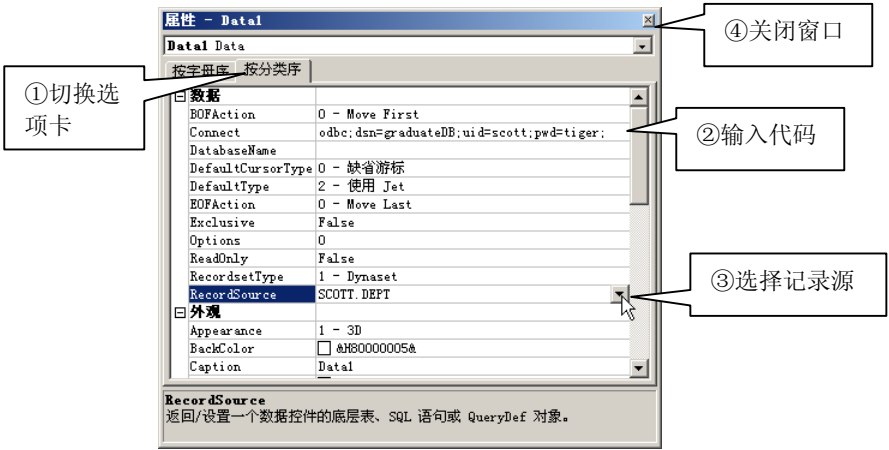


图 10.15 【Data】控件的【属性窗口】界面

odbc;dsn=graduateDB;uid=scott;pwd=tiger;

(6) 如图 10.16 所示。

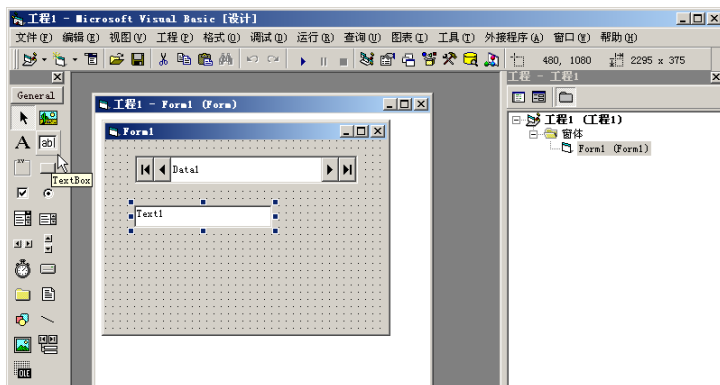


图 10.16 向窗体中添加文本框控件

(7) 选择【Form1】窗体上出现名为 Text1 的【TextBox】控件。

(8) 出现如图 10.17 所示的【TextBox】控件的属性窗口。

(9) 按照和 Text1 同样的步骤添加名为 Text2 的【TextBox】控件，其主要的设置为。

(10) 按照和 Text2 同样的步骤添加名为 Text3 的【TextBox】控件，其主要的设置为。

(11) 设计好的窗体 Form1 如图 10.18 所示。

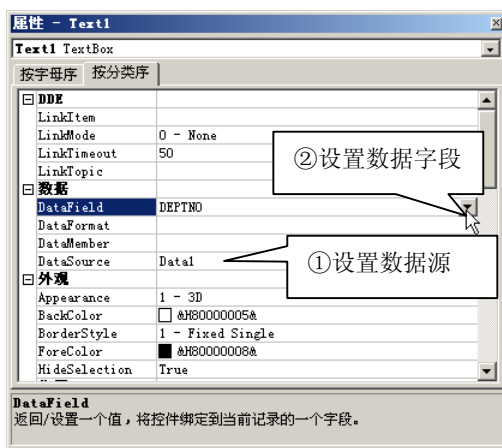


图 10.17 【TextBox】控件的【属性窗口】界面

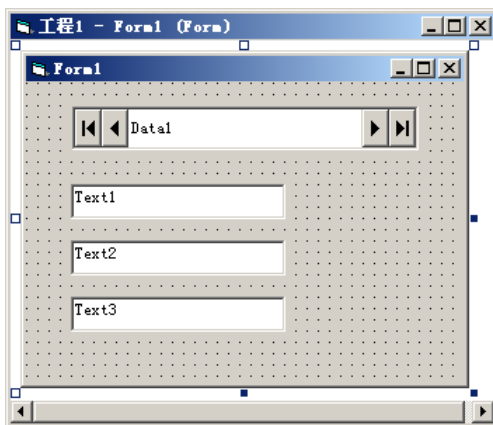



图 10.18 设计好的 Form1 窗体

(12) 在 VB 主界面的【快捷按钮区】单击运行快捷按钮, 如图 10.19 所示。

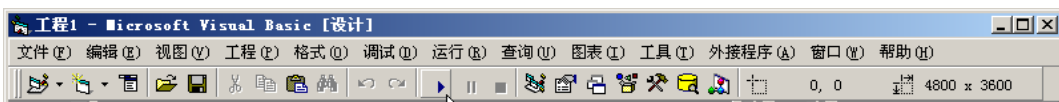


图 10.19 选择运行 Form1 窗体

(13) 运行结果如图 10.20 所示。

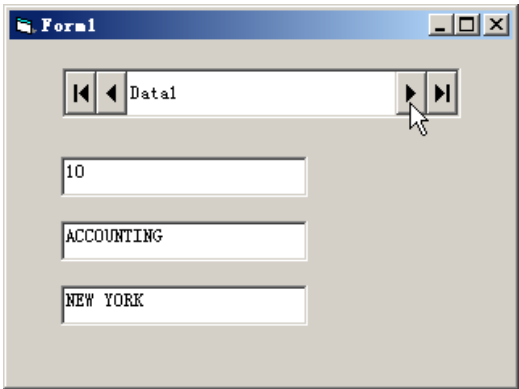


图 10.20 Form1 的运行结果

(14) 读者可以从本书的配套光盘上找到实例的配套程序。

【参见光盘文件】: 第 10 章\使用 Data 控件访问数据库.vbp, 窗体为 Form1.frm。

(15) 利用表格来表示上述过程中窗体的设置, 如表 10.1 所示。

表 10.1 Form1 窗体的设计

控件名称	类型	主要属性	属性值
Form1	【窗体】	名称	Form1
Data1	【Data】	Caption	Data1
		Connect	odbc;dsn=graduateDB;uid=scott;pwd=tiger;
		RecordSource	Scott.dept
Text1	【TextBox】	DataSource	Data1
		DataField	DEPTNO
		名称	Text1
Text2	【TextBox】	DataSource	Data1
		DataField	DNAME
		名称	Text2
Text3	【TextBox】	DataSource	Data1
		DataField	LOC
		名称	Text3

10.3.2 通过【ADODC】控件访问数据库

(1) 启动 VB, 建立新的工程。

(2) 如图 10.21 所示的 VB 主界面的【菜单栏】选择【工程】/【部件】选项。

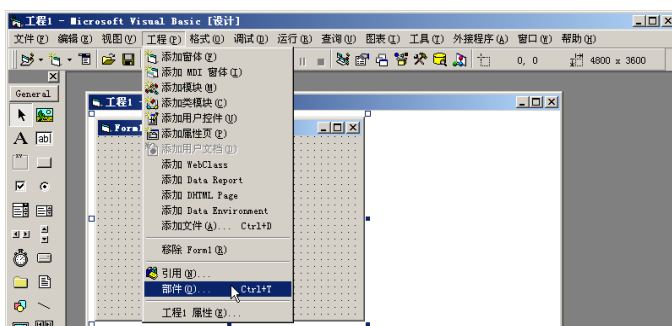


图 10.21 选择加载【ADODC】控件

(3) 出现如图 10.22 所示的部件的【控件】选项卡。

(4) 如图 10.23 所示。

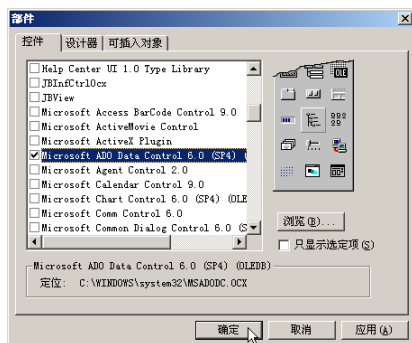


图 10.22 部件的【控件】选项卡

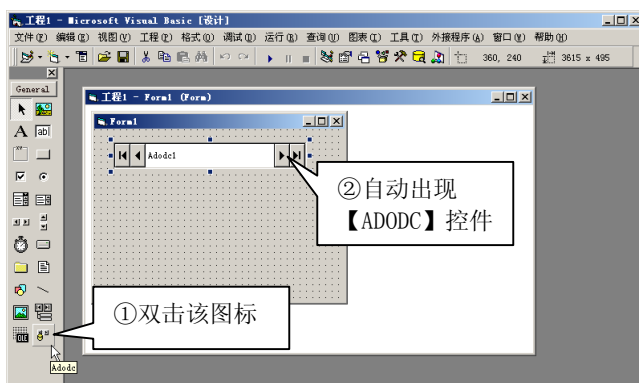


图 10.23 向窗体中添加【ADODC】控件

(5) 如图 10.24 所示。

(6) 出现如图 10.25 所示的属性页的【通用】选项卡。

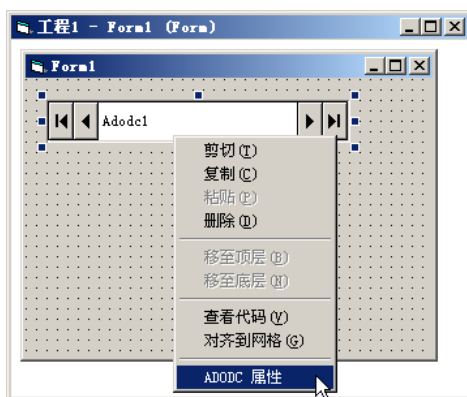


图 10.24 选择查看 adodc1 的属性窗口

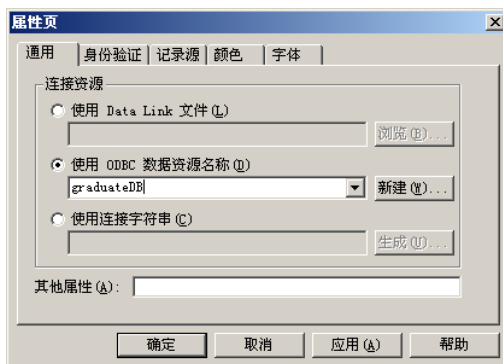


图 10.25 属性页的【通用】选项卡

(7) 切换到如图 10.26 所示的属性页的【身份验证】选项卡。

(8) 切换到如图 10.27 所示的属性页的【记录源】选项卡。

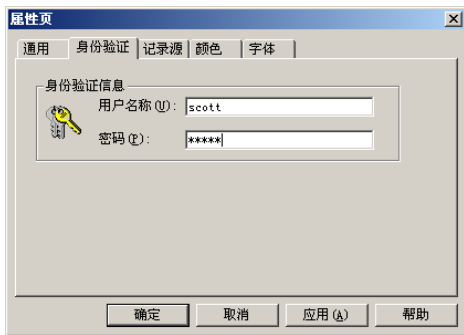


图 10.26 属性页的【身份验证】选项卡

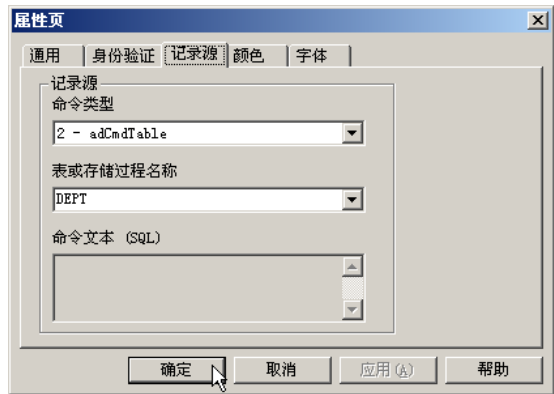


图 10.27 属性页的【记录源】选项卡

(9) 主要的属性设置如图 10.28 所示。

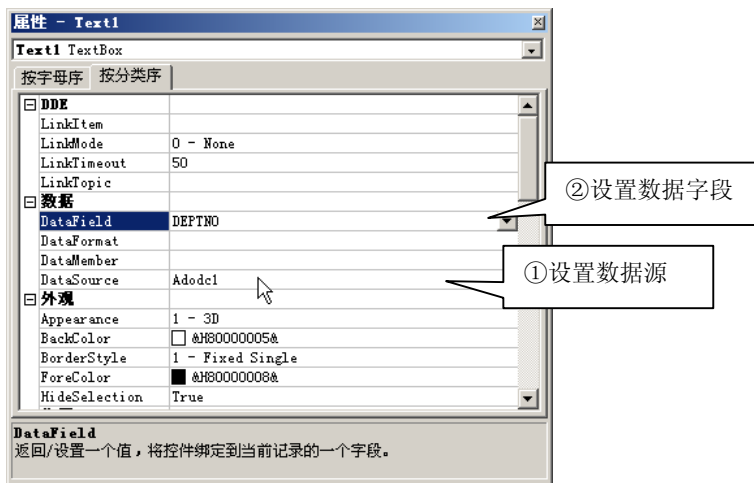


图 10.28 【TextBox】控件的【属性窗口】界面

(10) 向 Form1 窗体中添加一个名为 Text2 的【TextBox】控件。

(11) 向 Form1 窗体中添加一个名为 Text3 的【TextBox】控件。

(12) 读者可以从本书的配套光盘上找到实例的配套程序。

【参见光盘文件】：第 10 章\使用 ADODC 控件访问数据库.vbp，窗体为 Form2.frm。

(13) 利用表格来表示上述过程中窗体的设置，如表 10.2 所示。

表 10.2 Form2 窗体的设计

控件名称	类型	主要属性	属性值
Form2	【窗体】	名称	Form2
		Caption	Form2
Adodc1	【ADODC】	【通用】选项卡	使用 ODBC 数据源 graduateDB
		【身份验证】选项卡	【用户名】scott，【口令】tiger

		【记录源】选项卡	【命令类型】2-adCmdTable, 【表名】dept
Text1	【TextBox】	DataSource	Adodc1
		DataField	DEPTNO
		名称	Text1
Text2	【TextBox】	DataSource	Adodc1
		DataField	DNAME
		名称	Text2
Text3	【TextBox】	DataSource	Adodc1
		DataField	LOC
		名称	Text3

(14) 程序运行结果如图 10.29 所示。

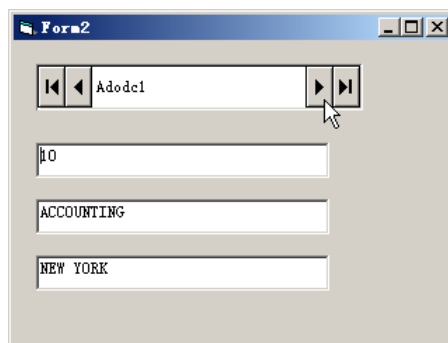


图 10.29 Form2 窗体的运行结果

10.3.3 通过【ADODB】编程对象访问数据库

1. 引用【ADODB】对象

要使用【ADODB】对象，必须首先进行引用。

- (1) 启动 VB，新建工程。
- (2) 如图 10.30 所示。
- (3) 出现如图 10.31 所示的【引用】界面。

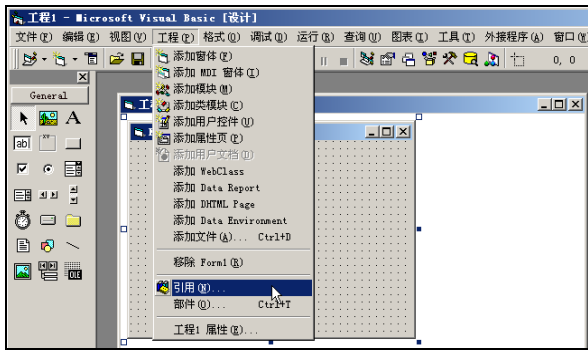


图 10.30 选择引用【ADODB】对象

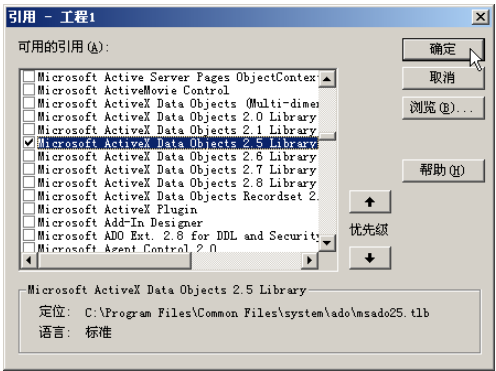


图 10.31 【引用】界面

2. 使用【ADODB】对象编程

(1) 设计如图 10.32 所示的窗体 Form3。



图 10.32 Form3 窗口的设计

利用表格来表示上述过程中窗体的设置，如表 10.3 所示。

表 10.3 Form3 窗体的设计

控件名称	类型	主要属性	属性值
Form3	【窗体】	名称	Form3
		Caption	使用 ADODB 访问数据库
Text1	【TextBox】	名称	Text1
Text2	【TextBox】	名称	Text2
Text3	【TextBox】	名称	Text3
CmdFirst	【CommandButton】	名称	CmdFirst
		Caption	首记录
CmdPrevious	【CommandButton】	名称	CmdPrevious
		Caption	上记录
CmdNext	【CommandButton】	名称	CmdNext

		Caption	下记录
CmdLast	【CommandButton】	名称	CmdLast
		Caption	末记录
CmdExit	【CommandButton】	名称	CmdExit
		Caption	退出

(2) 读者可以从本书的配套光盘上找到实例的配套程序。

【参见光盘文件】：第 10 章\使用 ADODB 对象编程访问数据库.vbp，窗体为 Form3.frm。

(3) 窗体 Form3 内所有的 VB 代码如下。

```

'强制性声明变量后才能使用
Option Explicit
'定义RS为ADODB对象的记录集
Dim Rs As New ADODB.Recordset
'定义conn为ADODB对象的连接
Private conn As ADODB.Connection
'窗体运行时初始化过程
Private Sub Form_Load()
    '定义ConnStr为连接字符串
    Dim ConnStr As String
    '定义Cmd为ADODB的命令字符串
    Dim Cmd As New ADODB.Command
    Set conn = New ADODB.Connection
    ConnStr = "DSN=graduateDB;uid=scott;pwd=tiger;"
    conn.ConnectionString = ConnStr
    '按照ConnStr的内容连接数据库
    conn.Open
    '查询数据表dept的内容
    With Cmd
        .ActiveConnection = conn
        .CommandType = adCmdTable
        .CommandText = "dept"
    End With
    '将数据表dept的内容送记录集RS
    With Rs
        .CursorLocation = adUseClient
        .CursorType = adOpenStatic
        .LockType = adLockPessimistic
        .Open Cmd
    End With
End Sub

```

```
End With
    '定义3个文本框的数据源为RS
Set Text1.DataSource = Rs
Set Text2.DataSource = Rs
Set Text3.DataSource = Rs
    '定义3个文本框显示的字段
Text1.DataField = "DEPTNO"
Text2.DataField = "DNAME"
Text3.DataField = "LOC"
End Sub

    '单击【首记录】按钮的执行过程
Private Sub CmdFirst_Click()
    Rs.MoveFirst
    Rs.Update
End Sub

    '单击【上记录】按钮的执行过程
Private Sub CmdPrevious_Click()
    Rs.MovePrevious
    If Rs.BOF Then
        Rs.MoveFirst
        Rs.Update
    End If
End Sub

    '单击【下记录】按钮的执行过程
Private Sub CmdNext_Click()
    Rs.MoveNext
    If Rs.EOF Then
        Rs.MoveLast
        Rs.Update
    End If
End Sub

    '单击【末记录】按钮的执行过程
Private Sub CmdLast_Click()
    Rs.MoveLast
    Rs.Update
End Sub

    '单击【退出】按钮的执行过程
Private Sub CmdExit_Click()
    conn.Close
    Set conn = Nothing
```

Unload Me
End Sub

(4) 窗体 Form3 的运行结果如图 10.33 所示。

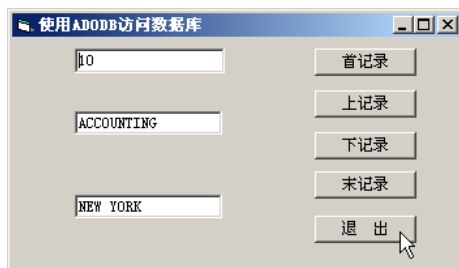


图 10.33 窗体 Form3 的运行结果

10.4 研究生管理信息系统的开发流程

10.4.1 系统需求分析

- ☐ 研究生个人信息管理
- ☐ 导师信息管理
- ☐ 专业信息管理

10.4.2 功能模块设计

针对上述需求，设计的功能模块如图 10.34 所示。

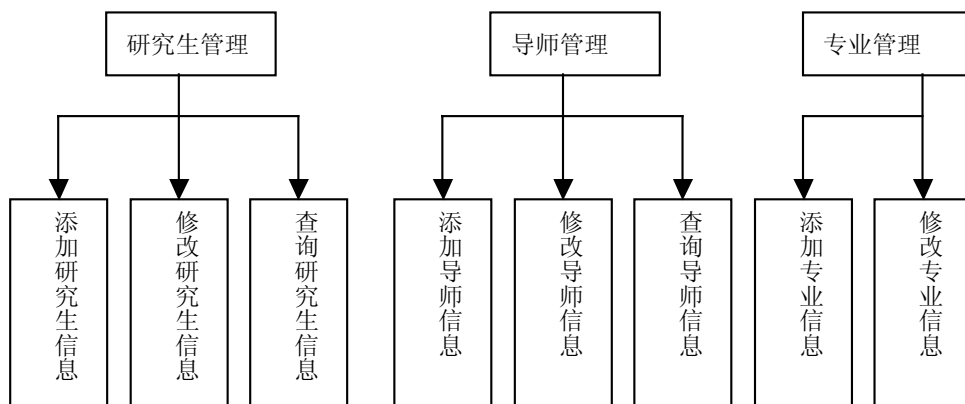


图 10.34 系统功能模块设计

10.4.3 数据表设计

1. 研究生信息数据表

数据表名称：graduate_info。数据表结构如表 10.4 所示。

表 10.4 graduate_info 数据表的结构

编号	字段名称	字段类型	字段宽度	说明
0	ID	NUMBER	10	学号
1	NAME	VARCHAR2	10	姓名
2	SEX	VARCHAR2	2	性别
3	TYPE	VARCHAR2	10	类别
4	BIRTHDAY	DATE		出生日期
5	CLASSNO	NUMBER	6	班级编号
6	MAJOR	VARCHAR2	30	专业
7	DIRECTOR	VARCHAR2	10	导师姓名
8	SOURCE	VARCHAR2	50	生源性质

2. 专业信息数据表

表 10.5 major_info 数据表的结构

编号	字段名称	字段类型	字段宽度	说明
0	ID	NUMBER	2	专业编号
1	NAME	VARCHAR2	30	专业名称

3. 导师信息数据表

表 10.6 director_info 数据表的结构

编号	字段名称	字段类型	字段宽度	说明
0	ID	NUMBER	10	学号
1	NAME	VARCHAR2	10	姓名
2	SEX	VARCHAR2	2	性别
3	ZHICHENG	VARCHAR2	20	职称
4	ZHIWU	VARCHAR2	60	职务
5	MAJOR	VARCHAR2	40	专业

10.4.4 索引和视图设计

- 1.研究生信息数据表的索引和视图设计
- 2.专业信息表的索引和视图设计
- 3.导师信息数据表的索引和视图设计

10.4.5 创建数据库用户

- (1) 在如图 10.35 所示的创建用户的【一般信息】选项卡中进行如下设置。
- (2) 切换到如图 10.36 所示的创建用户的【角色】选项卡。

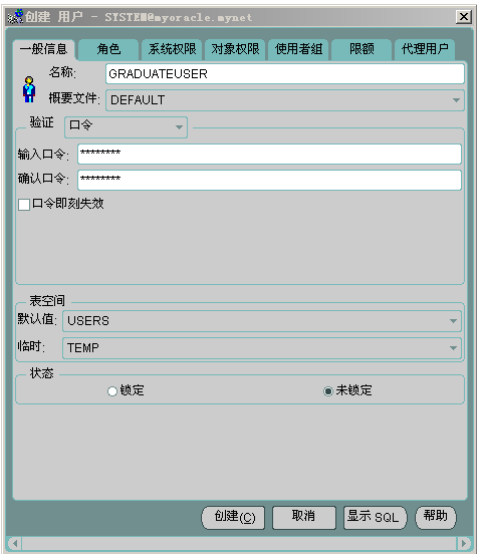


图 10.35 创建用户的【一般信息】选项卡

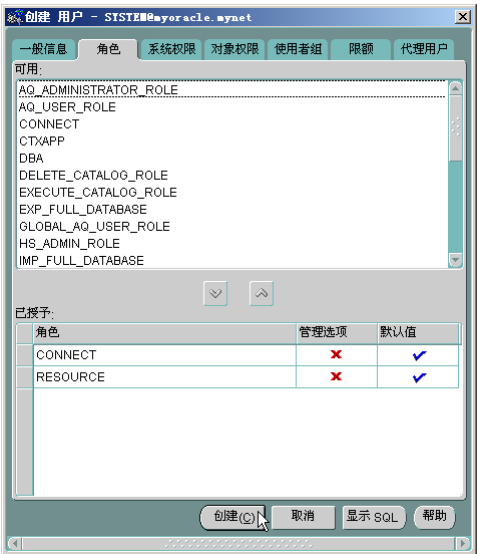


图 10.36 创建用户的【角色】选项卡

- (3) 成功创建上述用户的 SQL 代码如下。

```
CREATE USER "GRADUATEUSER" PROFILE "DEFAULT"  
IDENTIFIED BY "12345678" DEFAULT TABLESPACE "USERS"  
TEMPORARY TABLESPACE "TEMP"  
ACCOUNT UNLOCK;  
  
GRANT UNLIMITED TABLESPACE TO "GRADUATEUSER";  
GRANT "CONNECT" TO "GRADUATEUSER";  
GRANT "RESOURCE" TO "GRADUATEUSER";
```

【参见光盘文件】：第 10 章\creategraduateuser.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 creategraduateuser.sql 文件创建用户。

10.4.6 创建数据表

1. 创建研究生信息表

- (1) 在如图 10.37 所示的创建表的【一般信息】选项卡中进行如下操作。
- (2) 切换到如图 10.38 所示的创建表的【约束条件】选项卡。



图 10.37 创建表的【一般信息】选项卡

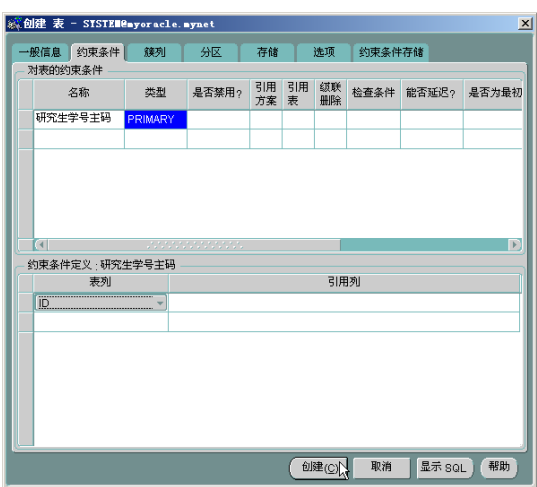


图 10.38 创建表的【约束条件】选项卡

(3) 上述创建数据表 graduateuser.graduate_info 的 SQL 代码如下。

```
CREATE TABLE "GRADUATEUSER"."GRADUATE_INFO" (  
    "ID"          NUMBER(10)      NOT NULL,  
    "NAME"        VARCHAR2(10)    NOT NULL,  
    "SEX"         VARCHAR2(2)      NOT NULL,  
    "TYPE"        VARCHAR2(10)    NOT NULL,  
    "BIRTHDAY"    DATE             NOT NULL,  
    "CLASSNO"     NUMBER(6)        NOT NULL,  
    "MAJOR"       VARCHAR2(30)     NOT NULL,  
    "DIRECTOR"    VARCHAR2(10)     NOT NULL,  
    "SOURCE"      VARCHAR2(50)     NOT NULL,  
    CONSTRAINT "研究生学号主码" PRIMARY KEY("ID"))  
TABLESPACE "USERS"
```

【参见光盘文件】：第 10 章\createtablegraduateinfo.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createtablegraduateinfo.sql 文件创建数据表 graduateuser.graduate_info。

2.创建专业信息表

- (1) 在如图 10.39 所示的创建表的【一般信息】选项卡中按照如图所示进行设置。
- (2) 切换到如图 10.40 所示的创建表的【约束条件】选项卡。



图 10.39 创建表的【一般信息】选项卡

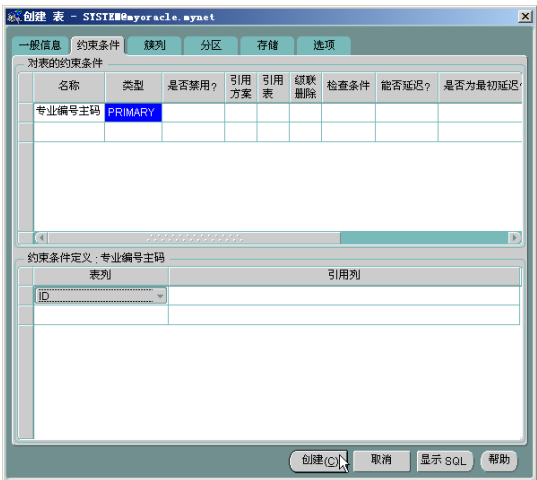


图 10.40 创建表的【约束条件】选项卡

- (3) 上述创建数据表 graduateuser.major_info 的 SQL 代码如下。

```
CREATE TABLE "GRADUATEUSER"."MAJOR_INFO" (  
    "ID"          NUMBER(2)      NOT NULL,  
    "NAME"        VARCHAR2(30)  NOT NULL,  
    CONSTRAINT "专业编号主码" PRIMARY KEY("ID"))  
TABLESPACE "USERS"
```

【参见光盘文件】：第 10 章\createtablemajorinfo.sql。

- (4) 读者可以直接在【SQLPlus Worksheet】中执行 createtablemajorinfo.sql 文件创建数据表 graduateuser.major_info。

3.创建导师信息表

- (1) 在如图 10.41 所示的创建表的【一般信息】选项卡中按照如图所示进行设置。
- (2) 切换到如图 10.42 所示的创建表的【约束条件】选项卡。



图 10.41 创建表的【一般信息】选项卡

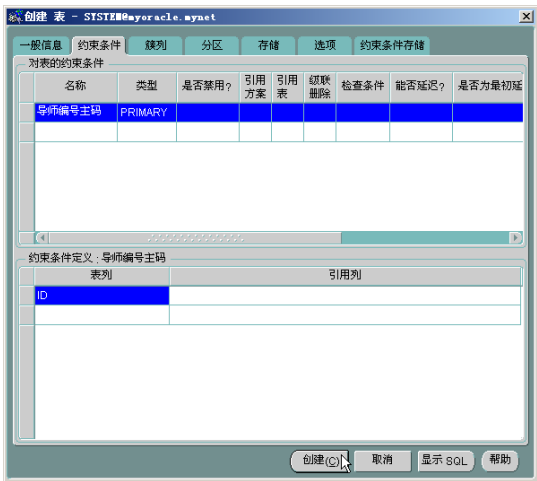


图 10.42 创建表的【约束条件】选项卡

(3) 上述创建数据表 graduateuser.director_info 的 SQL 代码如下。

```
CREATE TABLE "GRADUATEUSER"."DIRECTOR_INFO" (  
    "ID"          NUMBER(10)    NOT NULL,  
    "NAME"        VARCHAR2(10)  NOT NULL,  
    "SEX"         VARCHAR2(2)   NOT NULL,  
    "ZHICHENG"    VARCHAR2(20)  NOT NULL,  
    "ZHIWU"       VARCHAR2(20)  NOT NULL,  
    "MAJOR"       VARCHAR2(40)  NOT NULL,  
    CONSTRAINT "导师编号主码" PRIMARY KEY("ID"))  
TABLESPACE "USERS"
```

【参见光盘文件】：第 10 章\createtabledirectorinfo.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createtabledirectorinfo.sql 文件创建数据表 graduateuserdirector_info。

10.4.7 创建索引

1.为研究生信息表的 NAME 字段建立索引

- (1) 在如图 10.43 所示的创建索引的【一般信息】选项卡中按照图示进行操作。
- (2) 其他选项卡按照默认设置即可。



图 10.43 创建索引的【一般信息】选项卡

(3) 上述创建索引的 SQL 代码如下。

```
CREATE INDEX "GRADUATEUSER"."研究生姓名字段索引"
ON "GRADUATEUSER"."GRADUATE_INFO"("NAME")
TABLESPACE "INDX"
```

【参见光盘文件】：第 10 章\createindexgraduate.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createindexgraduate.sql 文件创建数据表 graduateuser.graduate_info 的索引“研究生姓名字段索引”。

2. 为导师信息表的 NAME 字段建立索引

(1) 在如图 10.44 所示的创建索引的【一般信息】选项卡中按照图示进行操作。

(2) 其他选项卡按照默认设置即可。

(3) 上述创建索引的 SQL 代码如下。

```
CREATE INDEX "GRADUATEUSER"."导师姓名字段索引"
ON "GRADUATEUSER"."DIRECTOR_INFO"("NAME")
TABLESPACE "INDX"
```

【参见光盘文件】：第 10 章\createindexdirector.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createindexdirector.sql 文件创建数据表 graduateuser.director_info 的索引“导师姓名字段索引”。



图 10.44 创建索引的【一般信息】选项卡

10.4.8 创建视图

1.为研究生信息表建立视图

(1) 在如图 10.45 所示的创建视图的【一般信息】选项卡中按照图示进行操作。

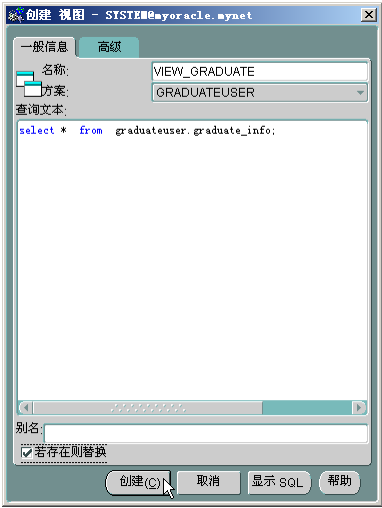


图 10.45 创建视图的【一般信息】选项卡

- (2) 其他选项卡按照默认设置即可。
- (3) 上述创建视图的 SQL 代码如下。

```
CREATE OR REPLACE VIEW "GRADUATEUSER"."VIEW_GRADUATE" AS
```

```
select *
from graduateuser.graduate_info WITH READ ONLY
```

【参见光盘文件】：第 10 章\createviewgraduate.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createviewgraduate.sql 文件创建数据表 graduateuser.graduate_info 的视图“VIEW_GRADUATE”。

2. 为专业信息表建立视图

(1) 在如图 10.46 所示的创建视图的【一般信息】选项卡中按照图示进行操作。

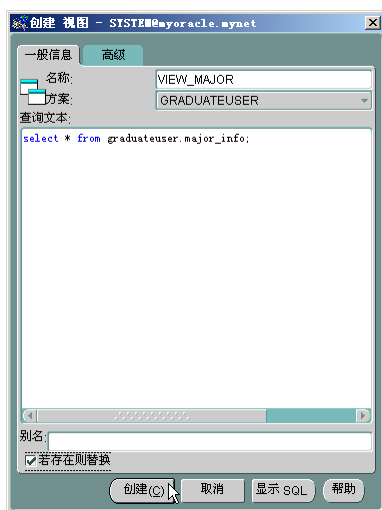


图 10.46 创建视图的【一般信息】选项卡

(2) 其他选项卡按照默认设置即可。

(3) 上述创建视图的 SQL 代码如下。

```
CREATE OR REPLACE VIEW "GRADUATEUSER"."VIEW_MAJOR" AS
select *
from graduateuser.major_info WITH READ ONLY
```

【参见光盘文件】：第 10 章\createviewmajor.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createviewmajor.sql 文件创建数据表 graduateuser.major_info 的视图“VIEW_MAJOR”。

2. 为导师信息表建立视图

(1) 在如图 10.47 所示的创建视图的【一般信息】选项卡中按照图示进行操作。

(2) 其他选项卡按照默认设置即可。

(3) 上述创建视图的 SQL 代码如下。

```

CREATE OR REPLACE VIEW "GRADUATEUSER"."VIEW_DIRECTOR" AS
select *
from graduateuser.director_info WITH READ ONLY

```

【参见光盘文件】：第 10 章\createviewdirector.sql。

(4) 读者可以直接在【SQLPlus Worksheet】中执行 createviewdirector.sql 文件创建数据表 graduateuser.director_info 的视图“VIEW_DIRECTOR”。

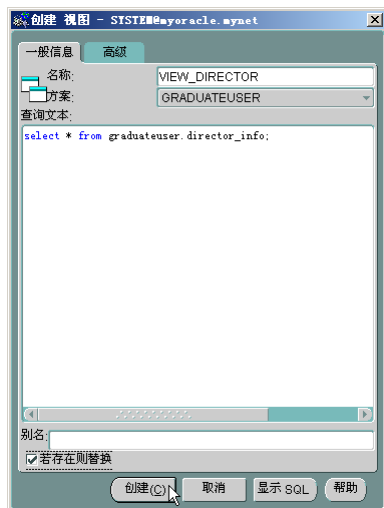


图 10.47 创建视图的【一般信息】选项卡

10.4.9 VB 工程总体框架

开发完成的研究生管理信息系统的总体框架如表 10.7 所示。

【参见光盘文件】：第 10 章\研究生管理信息系统.vbp。

表 10.7 研究生管理信息系统工程各成分

成分名称	成分类型	作用
Frmmain	窗体	主界面的主菜单调用各子窗体完成各项功能
FormInsertDirector	窗体	添加导师信息窗体
FormInsertGraduate	窗体	添加研究生信息窗体
FormInsertMajor	窗体	添加专业信息窗体
FormSelectDirector	窗体	查询导师信息窗体
FormSelectGraduate	窗体	查询研究生信息窗体
FormUpdateGraduate	窗体	修改研究生信息窗体

FormUpdateDirector	窗体	修改导师信息窗体
FormUpdateMajor	窗体	修改专业信息窗体
Module1	模块	全局性的公用函数

10.4.10 系统是如何运行的

- (1) 启动 VB，打开配套光盘的工程文件。
- (2) 如图 10.48 所示。

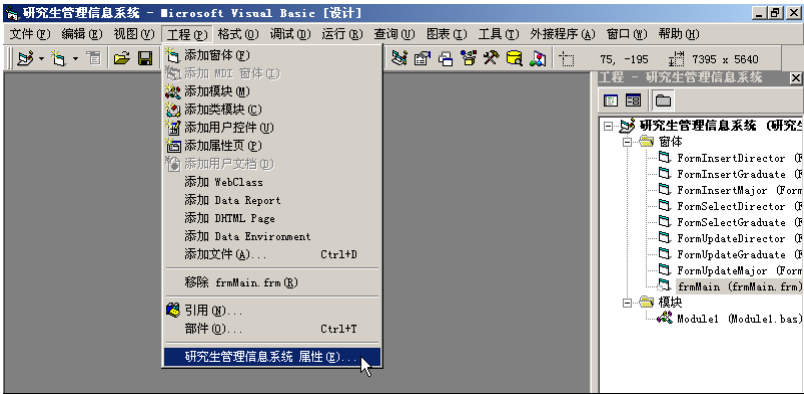


图 10.48 查看工程的属性

- (3) 出现如图 10.49 所示的工程属性的【通用】选项卡。
- (4) 打开模块 Module1，查看名为 Main 的过程的代码如下。说明 Main 过程将启动名为 frmMain 的主窗体。

```
Public fMainForm As frmMain
Sub Main()
    Set fMainForm = New frmMain
    fMainForm.Show
End Sub
```

显示主窗体

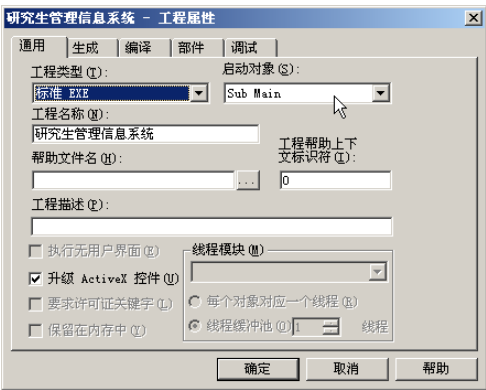


图 10.49 工程属性的【通用】选项卡

(5) 编辑名为 frmMain 的主窗体，如图 10.50 所示。

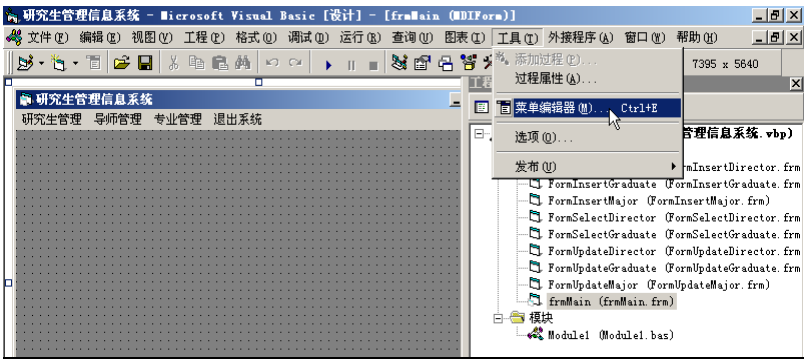


图 10.50 选择查看主窗体的菜单设计

(6) 出现如图 10.51 所示的【菜单编辑器】界面，可以查看主窗体的菜单设计。



图 10.51 【菜单编辑器】界面

工程中设计的菜单如表 10.8 所示。

表 10.8 主窗体设计的菜单

菜单名称	菜单分级	使用的变量	执行的操作
------	------	-------	-------

研究生管理	1	GraduateMenu	
添加研究生信息	2	GraduateInsertMenu	调用窗体 FormInsertGraduate
修改研究生信息	2	GraduateUpdateMenu	调用窗体 FormUpdateGraduate
查询研究生信息	2	GraduateSelectMenu	调用窗体 FormSelectGraduate
导师管理	1	DirectorMenu	
添加导师信息	2	DirectorInsertMenu	调用窗体 FormInsertDirector
修改导师信息	2	DirectorUpdateMenu	调用窗体 FormUpdateDirector
查询导师信息	2	DirectorSelectMenu	调用窗体 FormSelectDirector
专业管理	1	MajorMenu	
添加专业信息	2	MajorInsertMenu	调用窗体 FormInsertMajor
修改专业信息	2	MajorUpdateMenu	调用窗体 FormUpdateMajor
退出系统	1	ExitMenu	退出主窗体

(7) 为什么这些菜单可以调用这些子窗体或者执行退出主窗体的操作呢？在图 10.50 的主窗体中用鼠标双击，在出现的代码窗口中的主要代码如下。

```

-----
‘选择菜单项DirectorInsertMenu后将显示子窗体FormInsertDirector
Private Sub DirectorInsertMenu_Click()
    FormInsertDirector.Show
End Sub
‘选择菜单项DirectorSelectMenu 后将显示子窗体FormSelectDirector
Private Sub DirectorSelectMenu_Click()
    FormSelectDirector.Show
End Sub
‘选择菜单项DirectorUpdateMenu后将显示子窗体FormUpdateDirector
Private Sub DirectorUpdateMenu_Click()
    FormUpdateDirector.Show
End Sub
‘选择菜单项GraduateInsertMenu后将显示子窗体FormInsertGraduate
Private Sub GraduateInsertMenu_Click()
    FormInsertGraduate.Show
End Sub
‘选择菜单项GraduateSelectMenu后将显示子窗体FormSelectGraduate
Private Sub GraduateSelectMenu_Click()
    FormSelectGraduate.Show
End Sub
‘选择菜单项GraduateUpdateMenu后将显示子窗体FormUpdateGraduate
Private Sub GraduateUpdateMenu_Click()

```

```

FormUpdateGraduate.Show
End Sub
‘选择菜单项MajorInsertMenu后将显示子窗体FormInsertMajor
Private Sub MajorInsertMenu_Click()
    FormInsertMajor.Show
End Sub
‘选择菜单项MajorUpdateMenu后将显示子窗体FormUpdateMajor
Private Sub MajorUpdateMenu_Click()
    FormUpdateMajor.Show
End Sub
‘选择菜单项ExitMenu后将显示退出主窗体
Private Sub ExitMenu_Click()
    End
End Sub
-----

```

10.4.11 模块的设计

模块 **Module1** 的其他主要代码如下。

```

-----
‘定义了名为ConnectString的无参数函数，连接数据库
Public Function ConnectString() As String
    ConnectString = "DSN=graduateDB;UID=graduateuser;PWD=12345678"
End Function
‘定义了名为ExecuteSQL的带参数函数，参数SQL和MsgString都为字符串型。该函数用于执行SQL语句。
Public Function ExecuteSQL(ByVal SQL As String, MsgString As String) As ADODB.Recordset
    ‘conn为ADODB的connection连接对象
    Dim conn As ADODB.Connection
    ‘rest为ADODB的记录集对象
    Dim rest As ADODB.Recordset
    ‘若执行SQL错误转向ExecuteSQL_Error标记处
    On Error GoTo ExecuteSQL_Error
    Set conn = New ADODB.Connection
    ‘连接数据库
    conn.Open ConnectString
    Set rest = New ADODB.Recordset
    ‘执行SQL字符串语句
    rest.Open Trim$(SQL), conn, adOpenKeyset, adLockOptimistic
    Set ExecuteSQL = rest
    MsgString = "查询到" & rest.RecordCount & " 条记录 "

```

‘执行SQL完毕后关闭连接和记录集

ExecuteSQL_Exit:

Set rest = Nothing

Set conn = Nothing

Exit Function

‘执行SQL错误时的处理方法

ExecuteSQL_Error:

MsgString = "查询错误: " & _

Err.Description

Resume ExecuteSQL_Exit

End Function

‘定义了名为Checktxt的带参数函数，参数txt为字符串型。该函数用于检测txt字符串是否有内容。

Public Function Checktxt(txt As String) As Boolean

If Trim(txt) = "" Then

Checktxt = False

Else

Checktxt = True

End If

End Function

10.4.12 各子窗体的设计

1. 添加信息窗体的设计

打开名为 FormInsertGraduate 的子窗体，如图 10.52 所示。

窗体的设计如表 10.8 所示。

表 10.8 子窗体 FormInsertGraduate 的设计

图 10.52 子窗体 FormInsertGraduate

控件名称	类型	主要属性	属性值
FormInsertGraduate	【窗体】	名称	FormInsertGraduate
		Caption	添加研究生信息
		StartPosition	2-屏幕中心

Label1	【Label】	Caption	学号
Label2	【Label】	Caption	姓名
Label3	【Label】	Caption	性别
Label4	【Label】	Caption	类别
Label5	【Label】	Caption	(例如 2002-09-10)
Label6	【Label】	Caption	班级编号
Label7	【Label】	Caption	专业
Label8	【Label】	Caption	导师姓名
Label9	【Label】	Caption	生源性质
Label10	【Label】	Caption	出生日期
CmdCommit	【CommandButton】	名称	CmdCommit
		Caption	确定添加
CmdCancel	【CommandButton】	名称	CmdCancel
		Caption	取消添加
txtID	【TextBox】	名称	TxtID(用于向 graduate_info 数据表的 ID 字段输入数据)
TxtName	【TextBox】	名称	TxtName (用于向 graduate_info 数据表的 NAME 字段输入数据)
txtBirthday	【TextBox】	名称	TxtBirthday (用于向 graduate_info 数据表的 BIRTHDAY 字段输入数据)
TxtClassno	【TextBox】	名称	TxtClassno(用于向 graduate_info 数据表的 CLASSNO 字段输入数据)
TxtSource	【TextBox】	名称	TxtSource (用于向 graduate_info 数据表的 SOURCE 字段输入数据)
		Scrollbars	2-Vertical
ComboSex	【ComboBox】	名称	ComboSex(用于向 graduate_info 数据表的 SEX 字段输入数据)
		Style	0-DropDown Combo
		Text	请选择性别
ComboType	【ComboBox】	名称	ComboType(用于向 graduate_info 数据表的 TYPE 字段输入数据)
		Style	0-DropDown Combo
		Text	请选择类别
ComboMajor	【ComboBox】	名称	ComboMajor (用于向 graduate_info 数据表的 MAJOR 字段输入数据)

CombDirector	【ComboBox】	Style	0-DropDown Combo
		Text	请选择专业
		名称	CombDirector（用于向 graduate_info 数据表的 DIRECTOR 字段输入数据）
		Style	0-DropDown Combo
		Text	请选择导师姓名

窗体的程序代码分析如下。

‘窗体启动时的执行代码

```
Private Sub Form_Load()
```

‘定义窗体要使用的对象和变量

```
Dim TempRecordset As ADODB.Recordset
```

```
Dim txtSQL As String
```

```
Dim MsgText As String
```

```
Dim i As Integer
```

‘向【性别】下拉列表框中添加两个选项

```
comboSex.AddItem "男"
```

```
comboSex.AddItem "女"
```

‘向【类别】下拉列表框中添加3个选项

```
ComboType.AddItem "硕士"
```

```
ComboType.AddItem "博士"
```

```
ComboType.AddItem "博士后"
```

‘从导师信息表的视图view_director中取出导师姓名，添加到【导师姓名】下拉列表框中供选择

```
txtSQL = "select name from view_director"
```

```
Set TempRecordset = ExecuteSQL(txtSQL, MsgText)
```

```
For i = 1 To TempRecordset.RecordCount
```

```
    comboDirector.AddItem TempRecordset.Fields(0)
```

```
    TempRecordset.MoveNext
```

```
Next i
```

```
TempRecordset.Close
```

‘从专业信息表的视图view_major中取出专业姓名，添加到【专业】下拉列表框中供选择

```
txtSQL = "select name from view_major"
```

```
Set TempRecordset = ExecuteSQL(txtSQL, MsgText)
```

```
For i = 1 To TempRecordset.RecordCount
```

```
    ComboMajor.AddItem TempRecordset.Fields(0)
```

```
    TempRecordset.MoveNext
```

```
Next i
```

```
TempRecordset.Close
```

```
End Sub
```

‘单击按钮  后的执行代码

```
Private Sub CmdCommit_Click()  
    Dim TempRecordset As ADODB.Recordset  
    Dim txtSQL As String  
    Dim MsgText As String  
    ‘调用全局模块中的函数Checktxt检查【姓名】文本框中是否输入了内容  
    If Not Checktxt(txtID.Text) Then  
        MsgBox "请输入学号!", vbOKOnly + vbExclamation, "警告"  
        txtID.SetFocus  
        Exit Sub  
    End If  
    If Not Checktxt(txtName.Text) Then  
        MsgBox "请输入姓名!", vbOKOnly + vbExclamation, "警告"  
        txtName.SetFocus  
        Exit Sub  
    End If  
    ‘调用全局模块中的函数Checktxt检查【性别】下拉列表框中是否输入了内容  
    If Not Checktxt(comboSex.Text) Then  
        MsgBox "请选择性别!", vbOKOnly + vbExclamation, "警告"  
        comboSex.SetFocus  
        Exit Sub  
    End If  
    ‘调用全局模块中的函数Checktxt检查【类别】下拉列表框中是否输入了内容  
    If Not Checktxt(ComboType.Text) Then  
        MsgBox "请输入研究生类别!", vbOKOnly + vbExclamation, "警告"  
        ComboType.SetFocus  
        Exit Sub  
    End If  
    ‘调用全局模块中的函数Checktxt检查【出生日期】文本框中是否输入了内容  
    If Not Checktxt(txtBirthday.Text) Then  
        MsgBox "请输入出生日期!", vbOKOnly + vbExclamation, "警告"  
        txtBirthday.SetFocus  
        Exit Sub  
    End If  
    ‘调用全局模块中的函数Checktxt检查【班级编号】文本框中是否输入了内容  
    If Not Checktxt(txtClassno.Text) Then  
        MsgBox "请输入班级编号!", vbOKOnly + vbExclamation, "警告"  
        txtClassno.SetFocus  
        Exit Sub  
    End If
```


‘调用全局模块中的函数Checktxt检查【专业】下拉列表框中是否输入了内容

```
If Not Checktxt(ComboMajor.Text) Then
    MsgBox "请输入专业!", vbOKOnly + vbExclamation, "警告"
    ComboMajor.SetFocus
Exit Sub
```

End If

‘调用全局模块中的函数Checktxt检查【导师姓名】下拉列表框中是否输入了内容

```
If Not Checktxt(comboDirector.Text) Then
    MsgBox "请输入导师姓名!", vbOKOnly + vbExclamation, "警告"
    comboDirector.SetFocus
Exit Sub
```

End If

‘调用VB系统函数IsNumeric检查【学号】文本框中是否输入的数字学号

```
If Not IsNumeric(Trim(txtID.Text)) Then
    MsgBox "请输入数字学号!", vbOKOnly + vbExclamation, "警告"
Exit Sub
txtID.SetFocus
```

End If

‘上述输入信息无误后，查询研究生信息表中是否有重复学号的记录

```
txtSQL = "select * from graduate_Info where ID = '" & Trim(txtID.Text) & "'"
Set TempRecordset = ExecuteSQL(txtSQL, MsgText)
```

‘若发现研究生信息表中有重复学号的记录，提示重新输入数据

```
If TempRecordset.EOF = False Then
    MsgBox "学号重复，请重新输入!", vbOKOnly + vbExclamation, "警告"
    TempRecordset.Close
    txtID.SetFocus
```

Else

```
TempRecordset.Close
```

‘若没有发现研究生信息表中有重复学号的记录，检查【出生日期】文本框中输入的是否是日期型的数据

```
If Not IsDate(txtBirthday.Text) Then
    MsgBox "出生日期应输入日期格式 (yyyy-mm-dd)!", vbOKOnly + vbExclamation, "警告"
    txtBirthday.SetFocus
Else
    txtBirthday = Format(txtBirthday, "yyyy-mm-dd")
    If Not IsDate(txtBirthday.Text) Then
        MsgBox "出生日期应输入日期格式 (yyyy-mm-dd)!", vbOKOnly + vbExclamation, "警告"
        txtBirthday.SetFocus
    Else
        txtBirthday = Format(txtBirthday, "yyyy-mm-dd")
```

```
txtSQL = "select * from graduate_Info"
```

```
‘执行查询语句
```

```
Set TempRecordset = ExecuteSQL(txtSQL, MsgText)
```

‘给记录集赋值，每个数据表的字段对应了窗体中的每个输入数据，注意添加时要和数据表中的字段顺序一致。

```
TempRecordset.AddNew
```

```
TempRecordset.Fields(0) = Trim(txtID.Text)
```

```
TempRecordset.Fields(1) = Trim(txtName.Text)
```

```
TempRecordset.Fields(2) = Trim(comboSex.Text)
```

```
TempRecordset.Fields(3) = Trim(ComboType.Text)
```

```
TempRecordset.Fields(4) = Trim(txtBirthday.Text)
```

```
TempRecordset.Fields(5) = Trim(txtClassno.Text)
```

```
TempRecordset.Fields(6) = Trim(ComboMajor.Text)
```

```
TempRecordset.Fields(7) = Trim(comboDirector.Text)
```

```
TempRecordset.Fields(8) = Trim(txtSource.Text)
```

```
‘更新记录集，数据被插入到数据表中
```

```
TempRecordset.Update
```

```
MsgBox "添加研究生信息成功！", vbOKOnly + vbExclamation, "警告"
```

```
TempRecordset.Close
```

```
Unload Me
```

```
End If
```

```
End If
```

```
End Sub
```

```
End Sub
```

‘单击按钮  后的执行代码

```
Private Sub CmdCancel_Click()
```

```
‘释放窗体，返回到调用的主窗体
```

```
Unload Me
```

```
End Sub
```

2.修改信息窗体的设计

打开名为 FormUpdateGraduate 的子窗体，如图 10.53 所示。

图 10.53 子窗体 FormUpdateGraduate

窗体的设计如表 10.10 所示。

表 10.10 子窗体 FormUpdateGraduate 的设计

控件名称	类型	主要属性	属性值
FormUpdateGraduate	【窗体】	名称	FormUpdateGraduate
		Caption	修改研究生信息
		StartPosition	2-屏幕中心
Label1	【Label】	Caption	学号
Label2	【Label】	Caption	姓名
Label3	【Label】	Caption	性别
Label4	【Label】	Caption	类别
Label5	【Label】	Caption	班级编号
Label6	【Label】	Caption	专业
Label7	【Label】	Caption	导师姓名
Label8	【Label】	Caption	生源性质
Label9	【Label】	Caption	出生日期
TxtID	【TextBox】	名称	TxtID（无法修改，惟一编号）
TxtName	【TextBox】	名称	TxtName（NAME 字段）
TxtSource	【TextBox】	名称	TxtSource（SOURCE 字段）
		Scrollbars	2-Vertical
TxtBirthday	【TextBox】	名称	TxtBirthday（BIRTHDAY 字段）
TxtClassno	【TextBox】	名称	TxtClassno（CLASSNO 字段）
ComboSex	【ComboBox】	名称	ComboSex（SEX 字段）

		Style	0-DropDown Combo
ComboType	【ComboBox】	名称	ComboType (TYPE 字段)
		Style	0-DropDown Combo
ComboMajor	【ComboBox】	名称	ComboMajor (MAJOR 字段)
		Style	0-DropDown Combo
CombDirector	【ComboBox】	名称	ComboDirector (DIRECTOR 字段)
		Style	0-DropDown Combo
Frame1	【Frame】	名称	Frame1
Frame2	【Frame】	名称	Frame2
CmdFirst	【CommandButton】	名称	CmdFirst
		Caption	首记录
CmdPrevious	【CommandButton】	名称	CmdPrevious
		Caption	上记录
CmdNext	【CommandButton】	名称	CmdNext
		Caption	下记录
CmdFirst	【CommandButton】	名称	CmdFirst
		Caption	首记录
CmdLast	【CommandButton】	名称	CmdLast
		Caption	末记录
CmdEdit	【CommandButton】	名称	CmdEdit
		Caption	修改记录
CmdUpdate	【CommandButton】	名称	CmdUpdate
		Caption	更新数据
CmdCancel	【CommandButton】	名称	CmdCancel
		Caption	取消修改
CmdDelete	【CommandButton】	名称	CmdDelete
		Caption	删除记录
CmdExit	【CommandButton】	名称	CmdExit
		Caption	退出修改


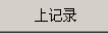

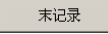
窗体的程序代码分析如下。

‘定义窗体级的变量，这样各模块可以使用


```
Dim TempRecordset As ADODB.Recordset
```

```
Dim TempBookmark As Variant
```

```


Dim Tempclean As Boolean
'启动窗体时执行的代码
Private Sub Form_Load()
    Dim txtSQL As String
    Dim MsgText As String
    '按照ID为顺序，从数据表中获取数据
    txtSQL = "select * from graduate_Info order by id"
    Set TempRecordset = ExecuteSQL(txtSQL, MsgText)
    '调用过程DisplayData，显示第一条记录
    TempRecordset.MoveFirst
    Call DisplayData
    mcbookmark = TempRecordset.Bookmark
    Tempclean = True
End Sub
'单击按钮  后的执行代码
Private Sub CmdFirst_Click()
    TempRecordset.MoveFirst
    Call DisplayData
End Sub
'单击按钮  后的执行代码
Private Sub CmdPrevious_Click()
    TempRecordset.MovePrevious
    If TempRecordset.BOF Then
        TempRecordset.MoveLast
    End If
    Call DisplayData
End Sub
'单击按钮  后的执行代码
Private Sub CmdNext_Click()
    TempRecordset.MoveNext
    If TempRecordset.EOF Then
        TempRecordset.MoveFirst
    End If
    Call DisplayData
End Sub
'单击按钮  后的执行代码
Private Sub CmdLast_Click()
    TempRecordset.MoveLast
    Call DisplayData
End Sub

```

‘单击按钮  后的执行代码

```
Private Sub CmdEdit_Click()  
    Tempclean = False  
    Frame1.Enabled = False  
    CmdFirst.Enabled = False  
    CmdPrevious.Enabled = False  
    CmdNext.Enabled = False  
    CmdLast.Enabled = False  
    txtID.Enabled = False  
    ‘ 激活其他编辑框  
    txtName.Enabled = True  
    comboSex.Enabled = True  
    comboType.Enabled = True  
    txtBirthday.Enabled = True  
    txtClassno.Enabled = True  
    ComboMajor.Enabled = True  
    comboDirector.Enabled = True  
    txtSource.Enabled = True  
    TempBookmark = TempRecordset.Bookmark
```

End Sub

‘单击按钮  后的执行代码

```
Private Sub CmdUpdate_Click()  
    Dim txtSQL As String  
    Dim MsgText As String  
    Dim TempRecordsetc As ADODB.Recordset  
    If Tempclean Then  
        MsgBox "请先选择修改研究生信息", vbOKOnly + vbExclamation, "警告"  
        Exit Sub  
    End If  
    If Not Checktxt(txtID.Text) Then  
        MsgBox "请输入学号!", vbOKOnly + vbExclamation, "警告"  
        txtID.SetFocus  
        Exit Sub  
    End If  
    If Not IsNumeric(Trim(txtID.Text)) Then  
        MsgBox "学号请输入数字!", vbOKOnly + vbExclamation, "警告"  
        txtID.SetFocus  
        Exit Sub  
    End If
```

```
If Not Checktxt(txtName.Text) Then
    MsgBox "请输入姓名!", vbOKOnly + vbExclamation, "警告"
    txtName.SetFocus
Exit Sub
End If

If Not Checktxt(comboSex.Text) Then
    MsgBox "请选择性别!", vbOKOnly + vbExclamation, "警告"
    comboSex.SetFocus
Exit Sub
End If

If Not Checktxt(comboType.Text) Then
    MsgBox "请选择类别!", vbOKOnly + vbExclamation, "警告"
    comboType.SetFocus
Exit Sub
End If

If Not Checktxt(txtBirthday.Text) Then
    MsgBox "请输入出生日期!", vbOKOnly + vbExclamation, "警告"
    txtBirthday.SetFocus
Exit Sub
End If

If Not Checktxt(txtClassno.Text) Then
    MsgBox "请选择班号!", vbOKOnly + vbExclamation, "警告"
    txtClassno.SetFocus
Exit Sub
End If

If Not Checktxt(ComboMajor.Text) Then
    MsgBox "请输入专业!", vbOKOnly + vbExclamation, "警告"
    ComboMajor.SetFocus
Exit Sub
End If



If Not Checktxt(comboDirector.Text) Then
    MsgBox "请输入导师姓名!", vbOKOnly + vbExclamation, "警告"
    comboDirector.SetFocus
Exit Sub
End If

If Not Checktxt(txtSource.Text) Then
    MsgBox "请输入生源性质!", vbOKOnly + vbExclamation, "警告"
    txtSource.SetFocus
Exit Sub
End If
```

```
TempRecordset.Delete
txtSQL = "select * from graduate_Info where ID = '" & Trim(txtID.Text) & "'"
Set TempRecordsetc = ExecuteSQL(txtSQL, MsgText)
If TempRecordsetc.EOF = False Then
    MsgBox "学号重复, 请重新输入!", vbOKOnly + vbExclamation, "警告"
    TempRecordsetc.Close
    txtID.SetFocus
Else
    TempRecordsetc.Close
    If Not IsDate(txtBirthday.Text) Then
        MsgBox "出生时间应输入日期格式 (yyyy-mm-dd)!", vbOKOnly + vbExclamation, "警告"
        txtBirthday.SetFocus
    Else
        txtBirthday = Format(txtBirthday, "yyyy-mm-dd")
        If Not IsDate(txtBirthday.Text) Then
            MsgBox "出生时间应输入日期格式 (yyyy-mm-dd)!", vbOKOnly + vbExclamation, "警告"
            txtBirthday.SetFocus
        Else
            txtBirthday = Format(txtBirthday, "yyyy-mm-dd")
            TempRecordset.AddNew
            TempRecordset.Fields(0) = Trim(txtID.Text)
            TempRecordset.Fields(1) = Trim(txtName.Text)
            TempRecordset.Fields(2) = Trim(comboSex.Text)
            TempRecordset.Fields(3) = Trim(comboType.Text)
            TempRecordset.Fields(4) = Trim(txtBirthday.Text)
            TempRecordset.Fields(5) = Trim(txtClassno.Text)
            TempRecordset.Fields(6) = Trim(ComboMajor.Text)
            TempRecordset.Fields(7) = Trim(comboDirector.Text)
            TempRecordset.Fields(8) = Trim(txtSource.Text)
            TempRecordset.Update
            MsgBox "修改研究生信息成功!", vbOKOnly + vbExclamation, "修改研究生信息"
            Call DisplayData
            Frame1.Enabled = True
            CmdFirst.Enabled = True
            CmdPrevious.Enabled = True
            CmdNext.Enabled = True
            CmdLast.Enabled = True
            txtID.Enabled = False
            txtName.Enabled = False
            comboSex.Enabled = False
```




```

        comboType.Enabled = False
        txtBirthday.Enabled = False
        txtClassno.Enabled = False
        ComboMajor.Enabled = False
        comboDirector.Enabled = False
        txtSource.Enabled = False
        Tempclean = True
    End If
End If
End If
End Sub
‘单击按钮  后的执行代码
Private Sub CmdCancel_Click()
    If Not Tempclean Then
        Frame1.Enabled = True
        CmdFirst.Enabled = True
        CmdPrevious.Enabled = True
        CmdNext.Enabled = True
        CmdLast.Enabled = True
        txtID.Enabled = False
        txtName.Enabled = False
        comboSex.Enabled = False
        comboType.Enabled = False
        txtBirthday.Enabled = False
        txtClassno.Enabled = False
        ComboMajor.Enabled = False
        comboDirector.Enabled = False
        txtSource.Enabled = False
        TempRecordset.Bookmark = TempBookmark
        Call DisplayData
    Else
        MsgBox "没有进行任何修改!", vbOKOnly + vbExclamation, "警告"
    End If
End Sub
‘单击按钮  后的执行代码
Private Sub CmdDelete_Click()
    TempBookmark = TempRecordset.Bookmark
    str2$ = MsgBox("是否删除当前记录?", vbOKCancel, "删除当前记录")
    If str2$ = vbOK Then
        TempRecordset.MoveNext
    End If
End Sub

```

```
        If TempRecordset.EOF Then
            TempRecordset.MoveFirst
            TempBookmark = TempRecordset.Bookmark
            TempRecordset.MoveLast
            TempRecordset.Delete
            TempRecordset.Bookmark = TempBookmark
            Call DisplayData
        Else
            TempBookmark = TempRecordset.Bookmark
            TempRecordset.MovePrevious
            TempRecordset.Delete
            TempRecordset.Bookmark = TempBookmark
            Call DisplayData
        End If
    Else
        TempRecordset.Bookmark = TempBookmark
        Call DisplayData
    End If
End Sub

‘单击按钮  后的执行代码
Private Sub CmdExit_Click()
    Unload Me
End Sub

‘显示记录内容的过程DisplayData的执行代码
Public Sub DisplayData()
    txtID.Text = TempRecordset.Fields(0)
    txtName.Text = TempRecordset.Fields(1)
    comboSex.Text = TempRecordset.Fields(2)
    comboType.Text = TempRecordset.Fields(3)
    txtBirthday.Text = Format(TempRecordset.Fields(4), "yyyy-mm-dd")
    txtClassno.Text = TempRecordset.Fields(5)
    ComboMajor.Text = TempRecordset.Fields(6)
    comboDirector.Text = TempRecordset.Fields(7)
    txtSource.Text = TempRecordset.Fields(8)
End Sub
```

3 查询信息窗体的设计

打开名为 FormSelectGraduate 的子窗体，如图 10.54 所示。窗体的设计如表 10.11 所示。

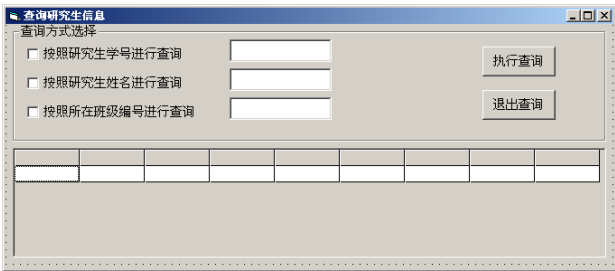


图 10.54 子窗体 FormSelectGraduate


表 10.11 子窗体 FormSelectGraduate 的设计

控件名称	类型	主要属性	属性值
FormSelectGraduate	【窗体】	名称	FormSelectGraduate
		Caption	查询研究生信息
		StartPosition	2-屏幕中心
Frame1	【Frame】	名称	Frame1
		Caption	查询方式选择
myflexgrid	【MSHFlexGrid】	名称	myflexgrid
Check1(0)	【CheckBox】	名称	Check1
		Caption	按照研究生学号进行查询
Check1(1)	【CheckBox】	名称	Check1
		Caption	按照研究生姓名进行查询
Check1(2)	【CheckBox】	名称	Check1
		Caption	按照所在班级编号进行查询
TxtID	【TextBox】	名称	TxtID
TxtName	【TextBox】	名称	TxtName
TxtClassno	【TextBox】	名称	TxtClassno
CmdSelect	【CommandButton】	名称	CmdSelect
		Caption	执行查询
CmdExit	【CommandButton】	名称	CmdExit
		Caption	退出查询

窗体的代码设计如下。

```
-----
'启动窗体时的执行代码
Private Sub Form_Load()
    '在窗体的myflexgrid表格上显示标题栏
```

```
With myflexgrid
    .CellAlignment = 4
    .TextMatrix(1, 0) = "学号"
    .TextMatrix(1, 1) = "姓名"
    .TextMatrix(1, 2) = "性别"
    .TextMatrix(1, 3) = "培养类别"
    .TextMatrix(1, 4) = "生日"
    .TextMatrix(1, 5) = "班级编号"
    .TextMatrix(1, 6) = "专业"
    .TextMatrix(1, 7) = "导师姓名"
    .TextMatrix(1, 8) = "生源性质"
End With
End Sub


‘单击  按钮后的执行代码
Private Sub CmdSelect_Click()
    Dim txtSQL As String
    Dim MsgText As String
    Dim Condition(3) As Boolean
    Dim TempRecordset As ADODB.Recordset
    ‘组合查询条件
    txtSQL = "select * from view_graduate where "
    If Check1(0).Value Then
        If Trim(txtID.Text) = "" Then
            sMsg = "学号不能为空"
            MsgBox sMsg, vbOKOnly + vbExclamation, "警告"
            txtID.SetFocus
            Exit Sub
        Else
            If Not IsNumeric(Trim(txtID.Text)) Then
                MsgBox "请输入数字学号!", vbOKOnly + vbExclamation, "警告"
                Exit Sub
            txtID.SetFocus
        End If
        Condition(0) = True
        txtSQL = txtSQL & "ID =' " & Trim(txtID.Text) & "' "
    End If
    End If
    If Check1(1).Value Then
        If Trim(txtName.Text) = "" Then
            sMsg = "姓名不能为空"
```

```

        MsgBox sMeg, vbOKOnly + vbExclamation, "警告"
        txtName.SetFocus
    Exit Sub
Else
    Condition(1) = True
    If Condition(0) Then
        txtSQL = txtSQL & "and Name =' " & txtName.Text & "' "
    Else
        txtSQL = txtSQL & "Name =' " & txtName.Text & "' "
    End If
End If
End If
If Check1(2).Value Then
    If Trim(txtClassno.Text) = "" Then
        sMeg = "班级编号不能为空"
        MsgBox sMeg, vbOKOnly + vbExclamation, "警告"
        txtClassno.SetFocus
    Exit Sub
Else
    Condition(2) = True
    If Condition(0) Or Condition(1) Then
        txtSQL = txtSQL & "and classno = ' " & txtClassno.Text & "' "
    Else
        txtSQL = txtSQL & "classno = ' " & txtClassno.Text & "' "
    End If
End If
End If
If Not (Condition(0) Or Condition(1) Or Condition(2)) Then
    MsgBox "请设置查询方式!", vbOKOnly + vbExclamation, "警告"
    Exit Sub
End If
txtSQL = txtSQL & " order by ID "
Set TempRecordset = ExecuteSQL(txtSQL, MsgText)
With myflexgrid
    .Rows = 2
    .CellAlignment = 4
    .TextMatrix(1, 0) = "学号"
    .TextMatrix(1, 1) = "姓名"
    .TextMatrix(1, 2) = "性别"
    .TextMatrix(1, 3) = "培养类型"

```

```

.TextMatrix(1, 4) = "生日"
.TextMatrix(1, 5) = "班级编号"
.TextMatrix(1, 6) = "专业"
.TextMatrix(1, 7) = "导师姓名"
.TextMatrix(1, 8) = "生源性质"
'显示查询结果
Do While Not TempRecordset.EOF
    .Rows = .Rows + 1
    .CellAlignment = 4
    .TextMatrix(.Rows - 1, 0) = TempRecordset.Fields(0)
    .TextMatrix(.Rows - 1, 1) = TempRecordset.Fields(1)
    .TextMatrix(.Rows - 1, 2) = TempRecordset.Fields(2)
    .TextMatrix(.Rows - 1, 3) = TempRecordset.Fields(3)
    .TextMatrix(.Rows - 1, 4) = Format(TempRecordset.Fields(4), "yyyy-mm-dd")
    .TextMatrix(.Rows - 1, 5) = TempRecordset.Fields(5)
    .TextMatrix(.Rows - 1, 6) = TempRecordset.Fields(6)
    .TextMatrix(.Rows - 1, 7) = TempRecordset.Fields(7)
    .TextMatrix(.Rows - 1, 8) = TempRecordset.Fields(8)
    TempRecordset.MoveNext
Loop
End With
TempRecordset.Close
End Sub
'单击  按钮后的执行代码
Private Sub CmdExit_Click()
    Unload Me
End Sub

```

10.5 如何执行实例程序

10.5.1 在服务器上的操作

(1) 以 system 用户、SYSDBA 身份登录【SQLPlus Worksheet】, 执行 creategraduateuser.sql 文件创建用户 graduateuser。

(2) 执行 createtablegraduateinfo.sql 文件创建数据表 graduateuser.graduate_info。

(3) 执行 createtabledirectorinfo.sql 文件创建数据表 graduateuser.director_info。

(4) 执行 createtablemajorinfo.sql 文件创建数据表 graduateuser.major_info。

(5) 执行 createindexgraduate.sql 文件创建数据表 graduateuser. graduate_info 的姓名字段的索引。

(6) 执行 createindexdirector.sql 文件创建数据表 graduateuser. director_info 的姓名字段的索引。

(7) 执行 createviewgraduate.sql 文件创建数据表 graduateuser. graduate_info 的视图。

(8) 执行 createviewdirector.sql 文件创建数据表 graduateuser. director_info 的视图。

(9) 执行 createviewmajor.sql 文件创建数据表 graduateuser. major_info 的视图。

10.5.2 在客户机上的操作

(1) 在客户机上调用 Oracle 9i 客户机的【网络配置助手】(Net Configuration Assistant) 配置一个能够连接数据库服务器的【本地网络服务名称】，笔者配置的名称为 myoraclelink。这个名称是可以任意取的。

(2) 在客户机的【控制面板】/【32 位 ODBC】中创建一个系统数据源名称，这个名称必须是 graduateDB。当然如果读者要采用自己的名称，可以在 VB 程序的模块 MODULE1 中进行适当的修改。修改的代码如下。

```
-----
'定义了名为ConnectionString的无参数函数，连接数据库
Public Function ConnectString() As String
    ConnectString = "DSN=graduateDB;UID=graduateuser;PWD=12345678"
End Function
-----
```

10.5.3 在客户机上运行程序

(1) 将配套光盘上的 Graduate.exe 文件任意复制到客户机上，执行该文件出现如图 10.55 所示界面。



图 10.55 研究生管理信息系统主界面

(2) 出现如图 10.56 所示的【添加专业信息】界面。

(3) 成功添加专业信息后出现如图 10.57 所示的提示界面。

(4) 按照同样的步骤依次添加专业信息数据，这样添加后的专业数据将用于研究生信息录入和导师信息录入窗体使用。

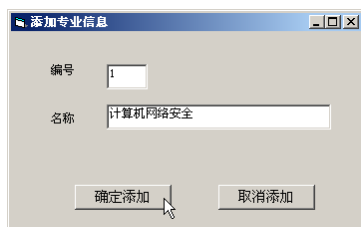


图 10.56 【添加专业信息】界面



图 10.57 成功添加专业信息

(5) 出现如图 10.58 所示的【添加导师信息】界面。

(6) 成功添加导师信息后出现如图 10.59 所示的界面。

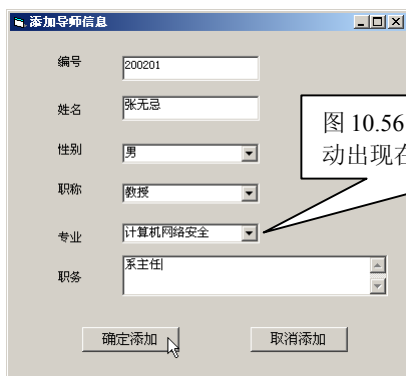


图 10.58 【添加导师信息】界面



图 10.59 成功添加导师信息

(7) 按照同样的步骤依次添加导师信息数据。

(8) 出现如图 10.60 所示的界面。

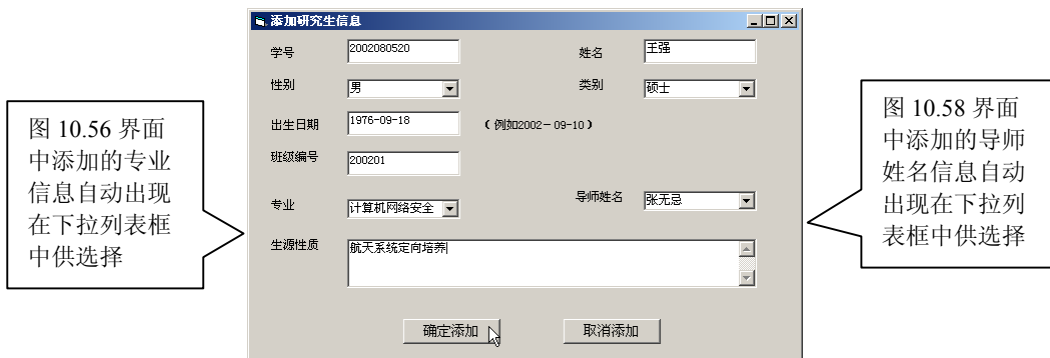


图 10.60 【添加研究生信息】界面

(9) 成功添加研究生信息后出现如图 10.61 所示界面，单击 **确定** 按钮。



图 10.61 成功添加研究生信息

(10) 出现如图 10.62 所示的【修改研究生信息】界面。

图 10.62 【修改研究生信息】界面

(11) 出现如图 10.63 所示的【查询研究生信息】界面。

学号	姓名	性别	培养类型	生日	班级编号	专业	导师姓名	生源性质
2002080520	王强	男	硕士	1976-09-18	200201	计算机网络安全	张无忌	航天系统定向培养
2002080521	赵敏	女	博士	1978-09-11	200201	计算机网络安全	王海洋	国家统招

图 10.63 【查询研究生信息】界面

(12) 在图 10.55 所示的系统主界面的【菜单栏】选择【退出系统】选项将退出系统。

(13) 其他操作读者可以自行学习，这里不再赘述。

10.6 习题

- (1) 在构建 VB+Oracle 9i 的开发和运行环境中，有什么主要的步骤？
- (2) ODBC 数据源和本地网络服务名是什么关系？
- (3) 通过试验熟悉 VB 程序中 3 种访问数据库的方法。
- (4) 完成一个完整的管理信息系统的开发。写出开发报告以及可以完整的运行程序。

第 11 章 存储管理——深入 Oracle 9i 核心

本章简要介绍存储管理的内容。

11.1 大脑——控制文件

11.1.1 控制文件的作用

11.1.2 控制文件的内容

- (1) 如图 11.1 所示。
- (2) 出现如图 11.2 所示的编辑控制文件的【一般信息】选项卡。
- (3) 图 11.3 所示为编辑控制文件的【记录文档段】选项卡。

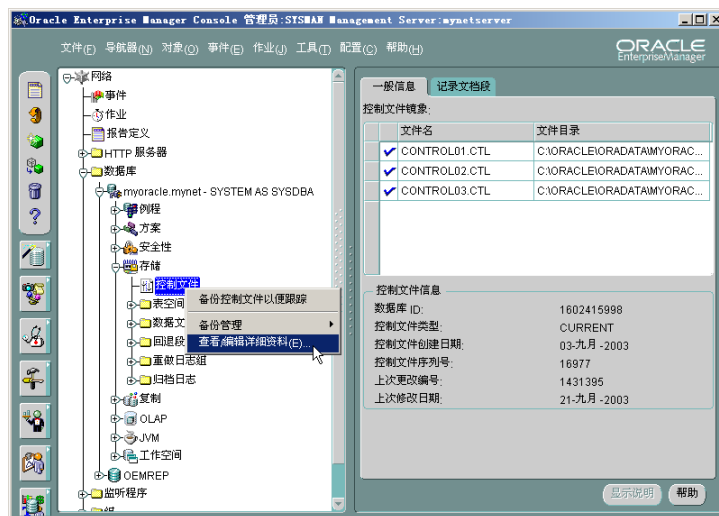


图 11.1 选择查看控制文件



图 11.2 编辑控制文件的【一般信息】选项卡 图 11.3 编辑控制文件的【记录文档】选项卡在【记录文档段】共记录了 8 类信息，各参数的含义如表 11.1 所示。

表 11.1 控制文件的【记录文档段】的内容

参数名称	参数含义
DATABASE	有关该数据库的信息
CKPT PROGRESS	可对每一个数据库例程的检查点进行跟踪记录的信息
REDO THREAD	显示分配给数据库重做日志的线程数
REDO LOG	显示当前分配的重做日志组数和数据库最大数量
DATAFILE	显示在数据库中创建的数据文件数和允许最大数量
FILENAME	显示存储在控制文件的文件名数量，包括数据文件、重做日志文件和控制文件
TABLESPACE	显示可以在数据库中创建的表空间数
LOG HISTORY	显示最大重做日志条目和当前分配的重做日志数目

11.2 文件夹——表空间

11.2.1 查询表空间的信息

- (1) 如图 11.4 所示。
- (2) 在【名称】单元格显示的是表空间的名称。
- (3) 在【类型】单元格显示的是表空间的类型，有 3 种类型。
- (4) 在【区管理】单元格显示的是表空间本地空间的管理方法。有两种类型。
- (5) 在【大小】单元格显示了表空间设置的大小。

(6) 在【已使用】单元格显示了表空间已经使用的空间大小。



图 11.4 选择查看表空间

11.2.2 创建表空间

(1) 如图 11.5 所示。

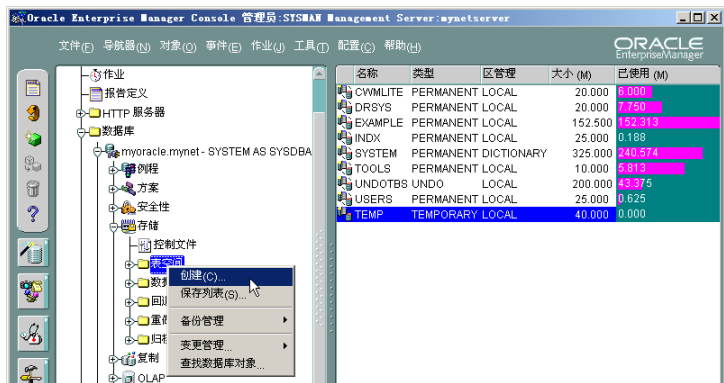


图 11.5 选择创建表空间

(2) 出现如图 11.6 所示的创建表空间的【一般信息】选项卡。

(3) 切换到如图 11.7 所示的创建表空间的【存储】选项卡。



图 11.6 创建表空间的【一般信息】选项卡

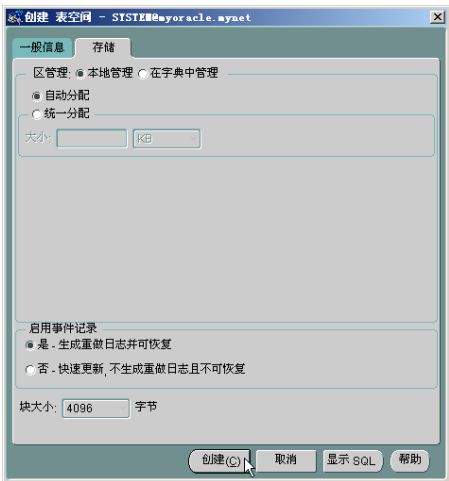


图 11.7 创建表空间的【存储】选项卡

(4) 成功创建表空间后出现如图 11.8 所示的界面，单击 **确定** 按钮。



图 11.8 成功创建表空间

(5) 上述创建表空间的 SQL 代码如下。

```
CREATE TABLESPACE "TEMPTABLESPACE"  
LOGGING  
DATAFILE 'C:\ORACLE\ORADATA\MYORACLE\TEMPTABLESPACE.ORA' SIZE  
5M REUSE EXTENT MANAGEMENT LOCAL
```

【参见光盘文件】：第 11 章\createtablespace.sql。

11.2.3 表空间脱机

1. 什么时候需要表空间脱机

- ❑ 让部分数据库不可用，而允许正常访问数据库的其他部分。
- ❑ 执行脱机表空间备份，尽管表空间处于联机或正在使用状态时也可进行备份。
- ❑ 使某个应用程序及其一组表在更新或维护该应用程序时暂时不可用。

2. 表空间脱机的 4 种方法

表 11.2 表空间脱机的 4 种方法

方法名称	执行操作
正常脱机	对表空间中所有数据文件（所有这些数据文件都必须是可用的）都将使用检查点
临时脱机	对表空间中所有联机数据文件都使用检查点
立即脱机	Oracle 不保证提供数据文件，而且不使用任何检查点
介质恢复脱机	该操作用于执行检查点恢复操作，可以将备份的表空间的数据文件进行复制，并用于归档日志文件

3. 脱机操作的步骤

- (1) 如图 11.9 所示。
- (2) 出现如图 11.10 所示的【脱机操作确认】界面，单击  按钮。

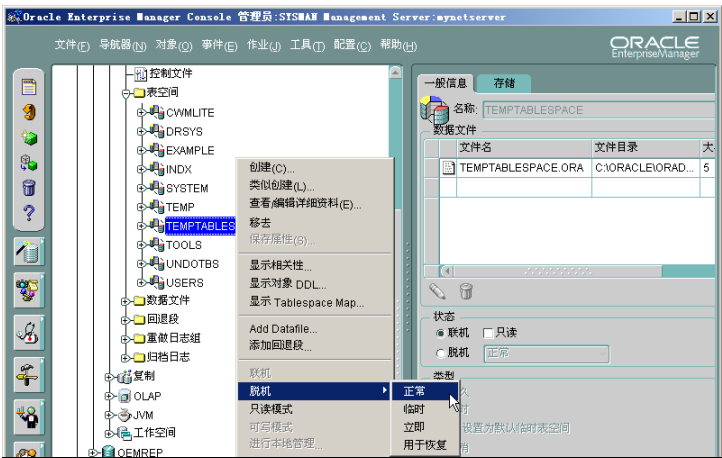


图 11.9 选择执行脱机操作



图 11.10 【脱机操作确认】界面

11.2.4 表空间联机

- (1) 如图 11.11 所示。
- (2) 出现如图 11.12 所示的【联机操作确认】界面。

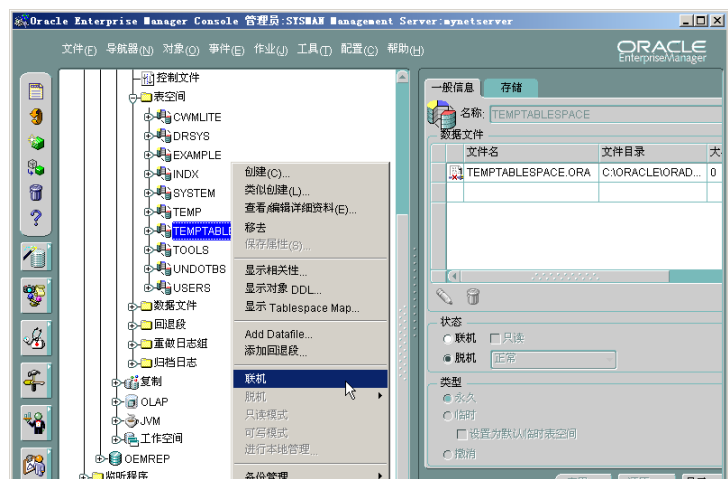


图 11.11 选择执行联机操作



图 11.12 【联机操作确认】界面

11.3 数据的仓库——数据文件

11.3.1 查询数据文件的信息

(1) 如图 11.13 所示。

名称	表空间	大小 (M)	已使用 (M)	占用率
C:\ORACLE\ORADATA\MYO... SYSTEM	SYSTEM	325.000	240.574	74.02
C:\ORACLE\ORADATA\MYO... UNDOTBS	UNDOTBS	200.000	43.375	21.69
C:\ORACLE\ORADATA\MYO... CWMLITE	CWMLITE	20.000	6.000	30.00
C:\ORACLE\ORADATA\MYO... DRSYS	DRSYS	20.000	7.750	38.75
C:\ORACLE\ORADATA\MYO... EXAMPLE	EXAMPLE	152.500	152.313	99.88
C:\ORACLE\ORADATA\MYO... INDX	INDX	25.000	0.188	0.75
C:\ORACLE\ORADATA\MYO... TOOLS	TOOLS	10.000	5.813	58.13
C:\ORACLE\ORADATA\MYO... USERS	USERS	25.000	0.625	2.50
C:\ORACLE\ORADATA\MYO... TEMP	TEMP	40.000	0.000	0.00

图 11.13 数据库已经建立的数据文件

- (2) 【名称】单元格显示的是数据文件的路径和名称。
- (3) 【表空间】单元格显示的是数据文件所在的表空间。
- (4) 【大小】单元格显示的是数据文件的空间大小。
- (5) 【已使用】单元格显示的是数据文件已经占用的空间大小。
- (6) 【占用率】单元格显示的是已经使用的空间占总空间的百分比。

11.3.2 建立数据文件

- (1) 如图 11.14 所示。
- (2) 出现如图 11.15 所示的创建数据文件的【一般信息】选项卡。

(3) 切换到如图 11.16 所示的创建数据文件的【存储】选项卡。

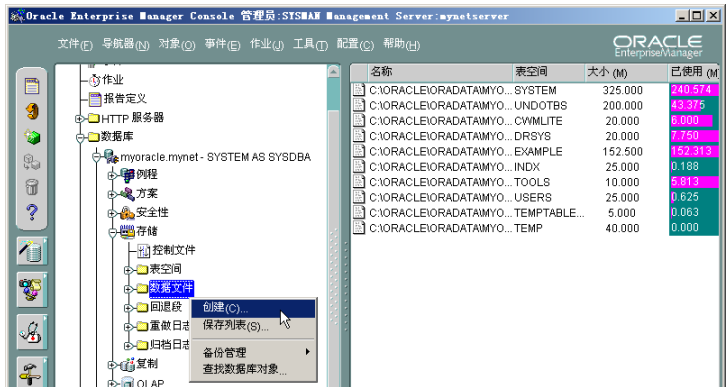


图 11.14 选择创建数据文件

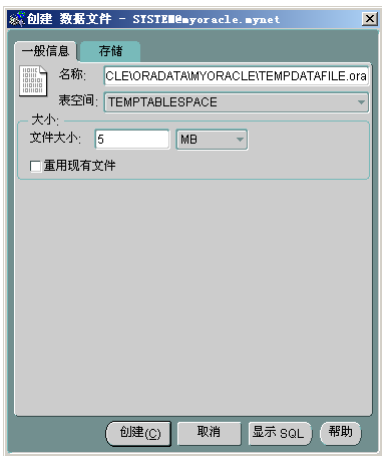


图 11.15 创建数据文件的【一般信息】选项卡

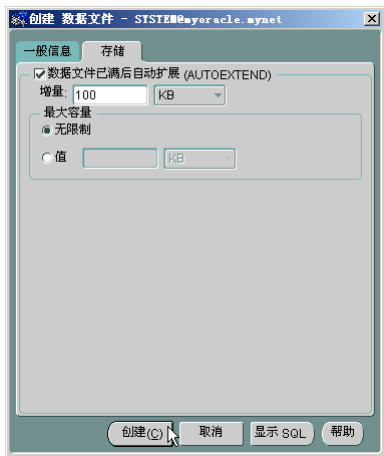


图 11.16 创建数据文件的【存储】选项卡

(4) 成功创建数据文件后出现如图 11.17 所示界面。



图 11.17 成功创建数据文件

(5) 上述创建数据文件的 SQL 代码如下。

```
ALTER TABLESPACE "TEMPTABLESPACE"  
ADD  
DATAFILE 'C:\ORACLE\ORADATA\MYORACLE\TEMPDATAFILE.ora' SIZE
```



```
5M AUTOEXTEND
ON NEXT 100K MAXSIZE UNLIMITED
```

【参见光盘文件】：第 11 章\createdatafile.sql。

11.3.3 数据文件脱机与脱机

- (1) 如图 11.18 所示。
- (2) 出现如图 11.19 所示界面。

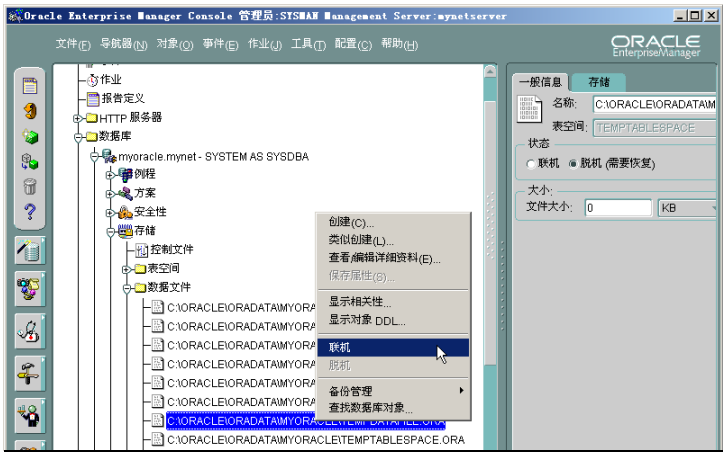


图 11.18 选择数据文件联机



图 11.19 【数据文件联机确认】界面

11.4 有备无患——回退段

表 11.3 段的类型

段的名称	段的作用
数据段	存放资料表或簇的资料的区的集合
索引段	存储索引数据的区的集合
回退段	存储要撤消的信息，有的书籍上也称为回滚段
临时段	当 SQL 语句需要临时空间时，将建立临时段。一旦执行完毕，临时段占用的空间将归还给系统

11.4.1 什么是回退段

回退段是一种特殊类型的数据段，记录着数据库被某个事务操作后的资料的原值，因此回退段里的资料可以用来对数据库进行恢复。

11.4.2 创建回退段

(1) 如图 11.20 所示。

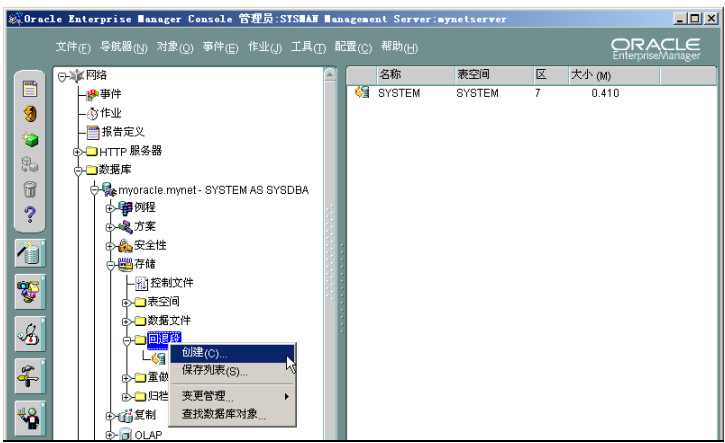


图 11.20 选择创建回退段

(2) 出现如图 11.21 所示的创建回退段的【一般信息】选项卡。

(3) 切换到如图 11.22 所示的创建回退段的【存储】选项卡。

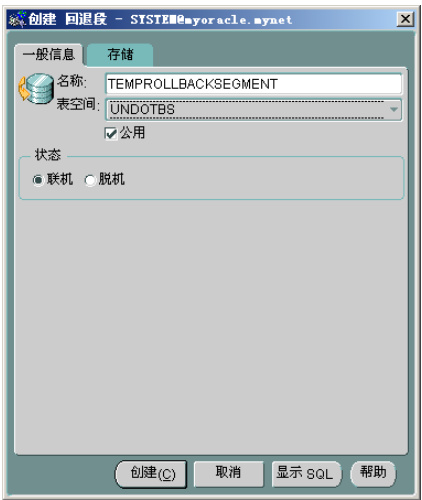


图 11.21 创建回退段的【一般信息】选项卡

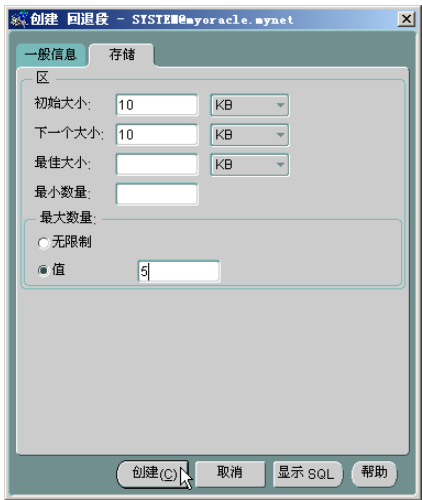


图 11.22 创建回退段的【存储】选项卡

(4) 出现如图 11.23 所示界面。



图 11.23 工作在自动撤消模式的数据库无法创建回退段

(5) 打开数据库的初始化文件 `init.ora`，其中有关回退段的设置参数如下。

```
# 系统管理的撤消和回退段
```

```
undo_management=AUTO
```

```
undo_tablespace=UNDOTBS
```

将上述代码更改为如下代码。

```
# 系统管理的撤消和回退段
```

```
undo_management=MANUAL
```

```
undo_tablespace=UNDOTBS
```

(6) 如图 11.24 所示。

(7) 成功创建回退段后的界面如图 11.25 所示。

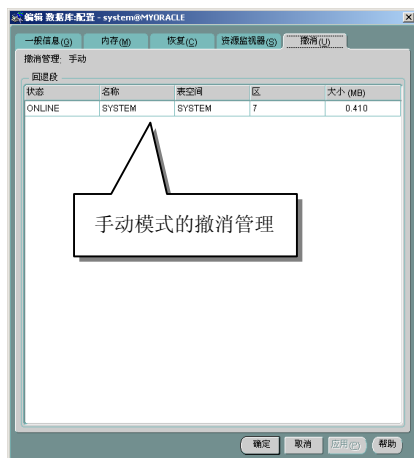


图 11.24 工作在手动撤消模式的数据库



图 11.25 成功创建回退段

(8) 上述创建回退段的 SQL 代码如下。

```
CREATE PUBLIC ROLLBACK SEGMENT "TEMPROLLBACKSEGMENT"
TABLESPACE "UNDOTBS"
STORAGE ( INITIAL 10K
           NEXT 10K
           MAXEXTENTS 5);
ALTER ROLLBACK SEGMENT "TEMPROLLBACKSEGMENT" ONLINE;
```

【参见光盘文件】：第 11 章\createrollbacksegment.sql。

11.5 黑匣子——重做日志组

11.5.1 重做日志组的工作原理

重做日志文件组记录了数据库的所有变化，其工作原理如图 11.26 所示。

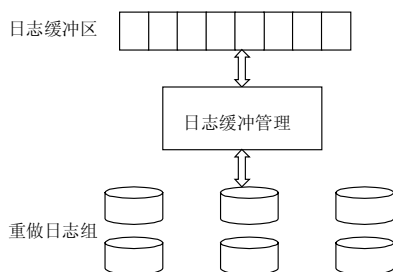


图 11.26 重做日志组的工作原理

11.5.1 查询默认的重做日志组信息

(1) 如图 11.27 所示。

状态	组	成员数	已归档	大小 (K)	序列	第一个更改编号
Inactive 1	1	1	No	102,400	59	1451729
Current 2	1	1	No	102,400	60	1472128
Inactive 3	1	1	No	102,400	58	1431394

图 11.27 默认的重做日志组

(2) 在【状态】单元格下显示了重做日志组的状态。有 4 种状态。

(3) 在【组】单元格显示的是组的序号。

(4) 在【成员数】单元格显示的是该组包含的日志文件数目。

(5) 在【已归档】单元格显示的是日志组是否已经归档。

(6) 在【大小】单元格显示的是日志组的大小。

(7) 在【序列】单元格显示的是日志组的序列号。

(8) 在【第一个更改编号】单元格显示的是日志组记录的系统更改编号，该编号用于进行恢复。

11.5.2 创建重做日志组

(1) 如图 11.28 所示。

(2) 出现如图 11.29 所示的创建重做日志组的【一般信息】选项卡。

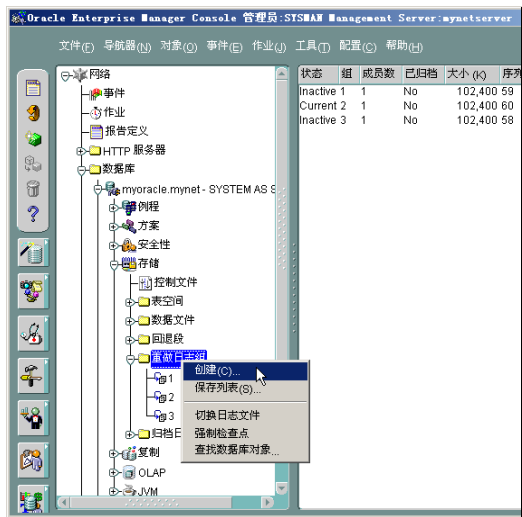


图 11.28 选择创建重做日志组

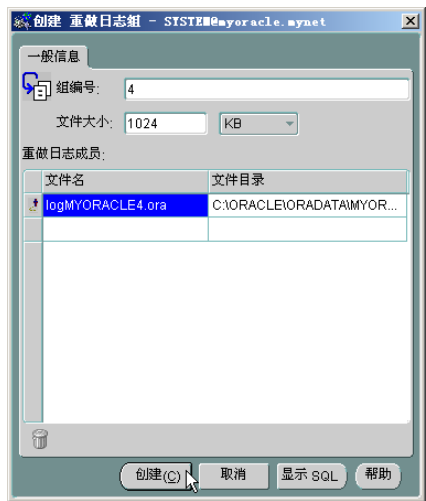


图 11.29 创建重做日志组的【一般信息】选项卡

(3) 出现如图 11.30 所示界面。



图 11.30 【成功创建重做日志组】界面

(4) 上述创建重做日志组的 SQL 代码如下。

```
ALTER DATABASE
  ADD LOGFILE GROUP 4
    ('C:\ORACLE\ORADATA\MYORACLE\logMYORACLE4.ora') SIZE 1024K
```

【参见光盘文件】：第 11 章\creatededologgroup.sql。

11.6 历史档案——归档日志

11.6.1 更改数据库的日志工作模式

(1) 如图 11.31 所示的编辑数据库配置的【一般信息】选项卡。

(2) 如图 11.32 所示的编辑数据库配置的【恢复】选项卡。



图 11.31 工作在非归档模式的数据库

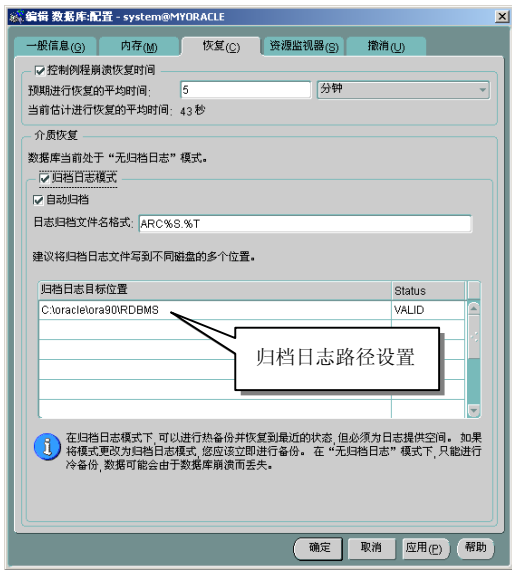


图 11.32 在编辑数据库配置的【恢复】选项卡中更改数据库工作模式

(3) 出现如图 11.33 所示的【数据库颤动】界面。

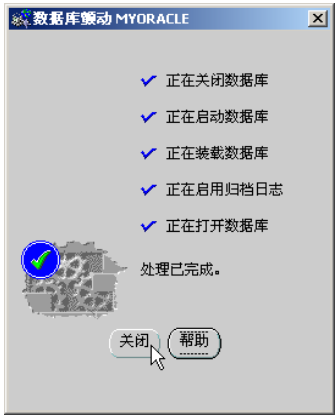


图 11.33 【数据库颤动】界面

11.6.2 日志文件的归档

(1) 执行结果如图 11.34 所示。

ALTER SYSTEM ARCHIVE LOG START;

【参见光盘文件】：第 11 章\archivelog.sql。

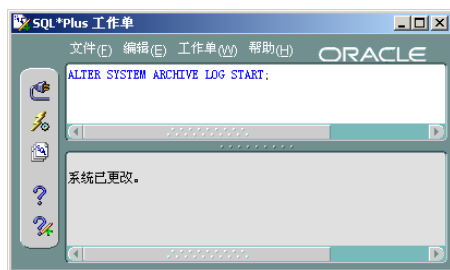


图 11.34 手动对日志文件进行归档

(2) 若执行以下 SQL 代码将手动归档所有未归档的日志文件。

```
ALTER SYSTEM ARCHIVE LOG ALL;
```

【参见光盘文件】：第 11 章\archivealllog.sql。

11.6.3 归档日志文件的信息

(1) 如图 11.35 所示。

(2) 出现如图 11.36 所示的编辑归档日志的【一般信息】选项卡。

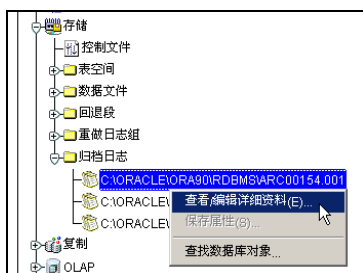


图 11.35 选择查看已经归档的日志文件的信息

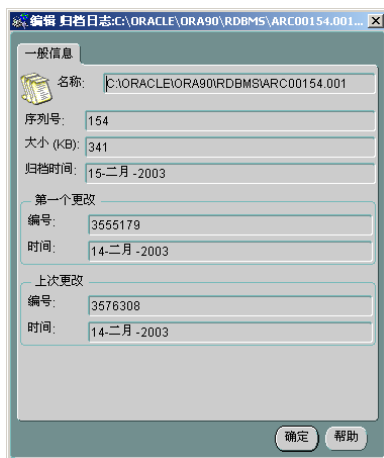


图 11.36 编辑归档日志的【一般信息】选项卡

11.7 习题

- (1) 控制文件有什么作用？数据库是如何利用控制文件启动的？
- (2) 表空间有什么作用？如何创建表空间？
- (3) 数据文件和表空间是什么关系？

- (4) 数据文件的联机和脱机操作和表空间的脱机和联机操作有什么关联?
- (5) 重做日志组是如何工作的?
- (6) 为什么要将重做日志组归档?
- (7) 分析 Oracle 9i 数据库的管理中, 什么时机需要执行归档操作?
- (8) 归档日志和重做日志组有什么关系?
- (9) 如何更改数据库的日志工作模式?
- (10) 比较两种日志工作模式有什么不同?

第 12 章 数据安全——备份与恢复实战

本章以实例的方式介绍 Oracle 9i 数据库的备份与恢复操作。

12.1 数据库常见故障及恢复原理

12.1.1 系统故障及恢复原理

系统故障主要是由于服务器在运行过程中，突然发生操作系统错误、停电等原因造成的非正常中断，用户对数据库进行处理的事务被突然中断，内存缓冲区中的数据全部丢失，但硬盘、磁带等外设上的数据未受损失。

12.1.2 介质故障及恢复原理

介质故障是由于硬件的可靠性较差出现的存储介质发生物理损坏。数据库的数据全部或部分丢失，破坏性较大。

12.1.3 事务故障及恢复原理

事务故障是某些对数据库进行操作的事务违反了系统设定的条件，如输入数据错误、运算溢出等，使事务未能正常完成就终止。发生事务故障时，事务对数据库的操作可能已经修改了部分数据，因此数据库管理系统必须提供某种恢复机制，强行回滚该事务对数据库的所有修改，使系统回到该事务发生前的状态。

12.2 Oracle 9i 的备份和恢复机制

12.2.1 备份和恢复的内容

1. 初始化参数文件
2. 控制文件

3. 数据文件
4. 联机重做日志文件
5. 归档日志文件

12.2.2 备份和恢复的方法及工具

1. 逻辑备份和恢复

实现逻辑备份的工具包括。

- ☐ 集成的导出向导：在【管理服务器】环境下的【企业管理器】中使用。
- ☐ EXP 命令文件：在【DOS 命令行】方式下使用。

实现逻辑恢复的工具包括。

- ☐ 集成的导入向导：在【管理服务器】环境下的【企业管理器】中使用。
- ☐ IMP 命令文件：在【DOS 命令行】方式下使用。

2. 物理备份和恢复

- ☐ 脱机备份
- ☐ 联机备份

12.3 脱机备份与恢复实战

12.3.1 脱机备份

- (1) 在【企业管理器】里关闭数据库例程。
- (2) 利用计算机的【资源管理器】查找与数据库有关的文件。如图 12.1 所示。

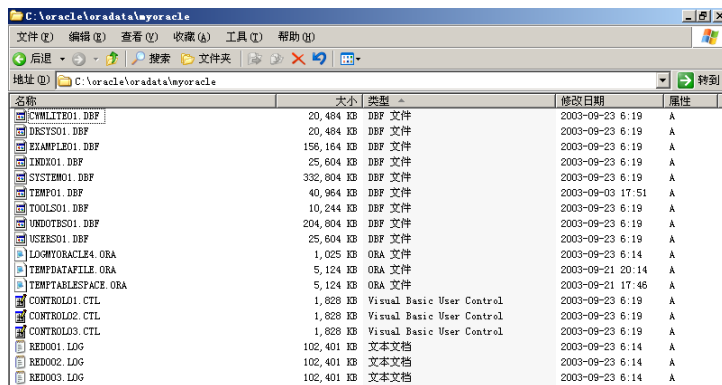


图 12.1 数据库的文件

(3) 数据库的初始化文件位于 c:\oracle\ora90\database 目录下，名为 initmyoracle.ora，将其拷贝到指定目录下。

12.3.2 脱机恢复

- (1) 在【企业管理器】里关闭数据库例程。
- (2) 将上述拷贝的文件重新覆盖原来的同路径同名文件就可以。

12.4 逻辑备份与恢复实战

12.4.1 逻辑备份与恢复的前提

1. 数据库工作在归档状态

2. 给数据库管理员授予角色权限

(1) 如图 12.2 所示的编辑用户的【角色】选项卡。

(2) 在【可用】下拉列表框里选中 EXP FULL DATABASE 和 IMP FULL DATABASE 角色，单击  按钮，在【已授予】列表框里出现已经授予的角色权限。

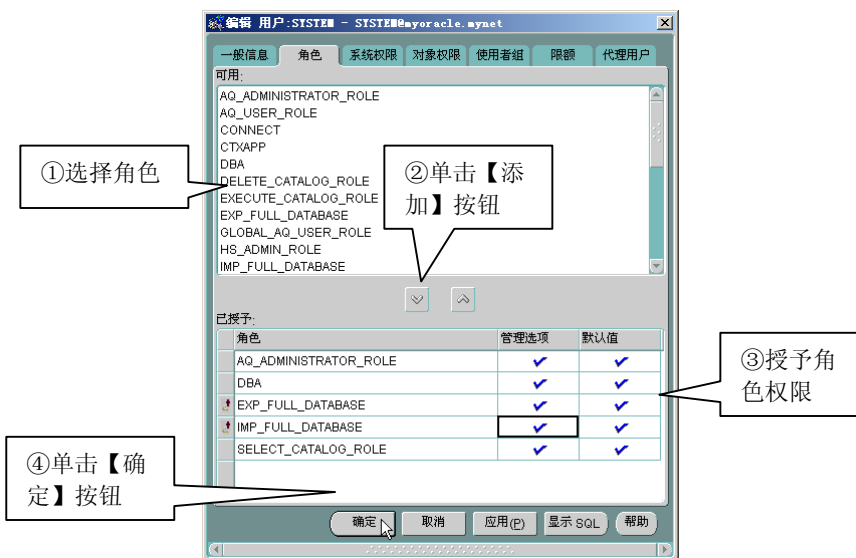


图 12.2 编辑用户的【角色】选项卡

3. 给 NT 管理员授予批处理作业权限

(1) 如图 12.3 所示的本地安全设置界面。

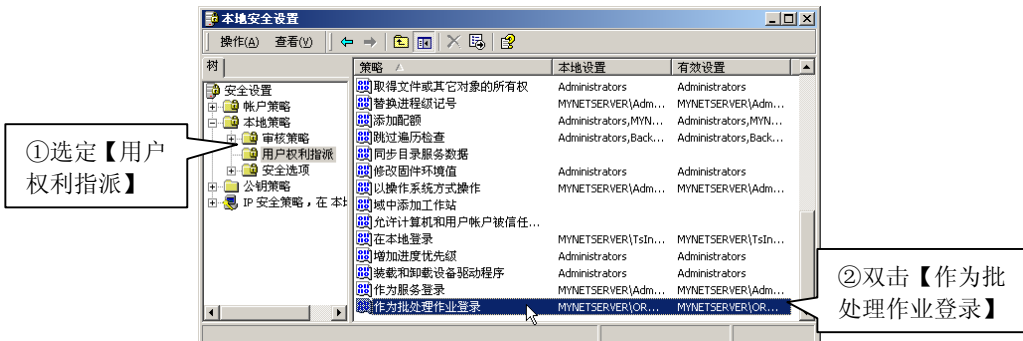


图 12.3 本地安全设置

(2) 出现如图 12.4 所示的【本地安全策略设置】界面。

(3) 出现如图 12.5 所示的【选择用户或组】界面。

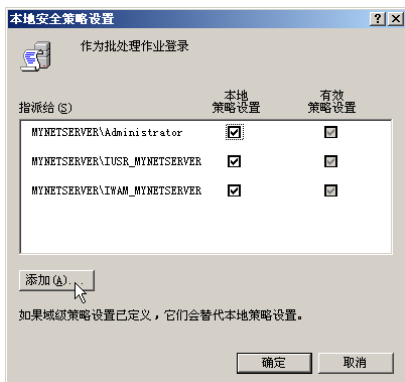


图 12.4 【本地安全策略设置】界面

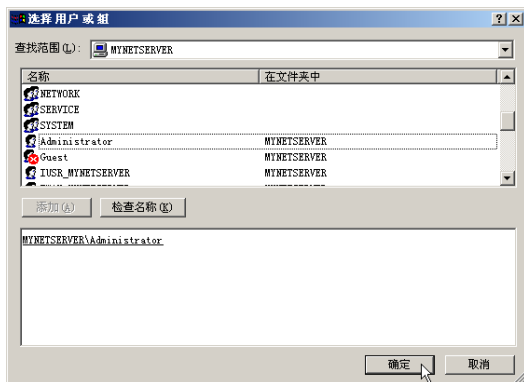


图 12.5 【选择用户或组】界面

4. 设置节点的首选身份证明

(1) 如图 12.6 所示。

(2) 切换到如图 12.7 所示的编辑管理员首选项的【首选身份证明】选项卡。

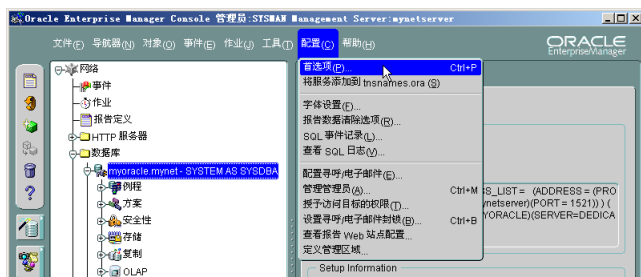


图 12.6 选择配置节点首选身份证明



图 12.7 设置节点首选身份证明

5. 设置数据库的首选身份证明



图 12.8 设置数据库首选身份证明

12.4.2 用 exp 命令文件实现逻辑备份

(1) 数据库连接成功后出现如图 12.9 所示界面。

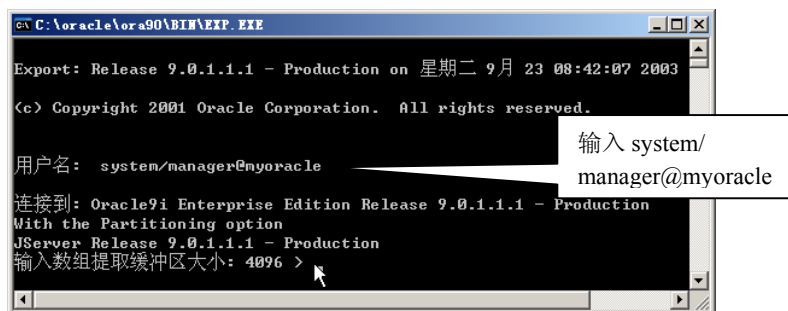


图 12.9 执行 exp.exe 命令

(2) 出现如图 12.10 所示界面。



图 12.10 设置逻辑备份参数

(3) 开始逻辑备份过程，出现如图 12.11 所示界面。

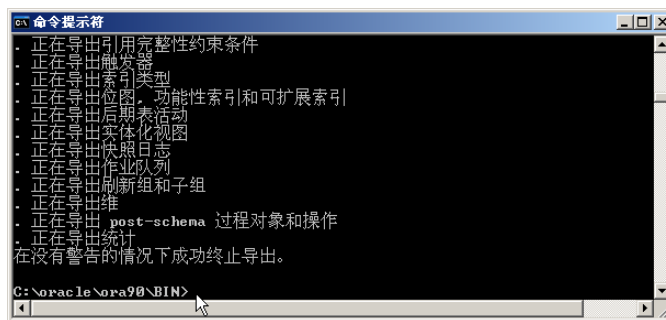


图 12.11 成功完成逻辑备份

(4) 在 c:\oracle\ora90\bin 目录下已经有名为 EXPDAT.DMP 的二进制文件存在。

12.4.3 用 imp 命令文件实现逻辑恢复

(1) 数据库连接成功后出现如图 12.12 所示界面。

(2) 出现如图 12.13 所示界面，

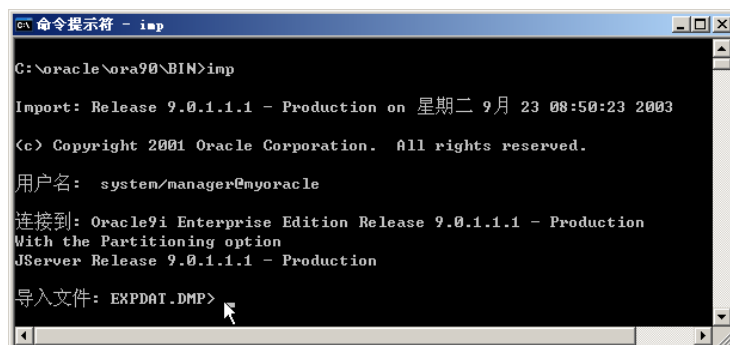


图 12.12 执行 imp 命令



图 12.13 设置逻辑恢复参数

(3) 出现如图 12.14 所示界面表明利用 imp 命令文件成功完成逻辑恢复。



图 12.14 成功完成逻辑恢复

(4) 出现如图 12.15 所示的界面显示其参数配置。

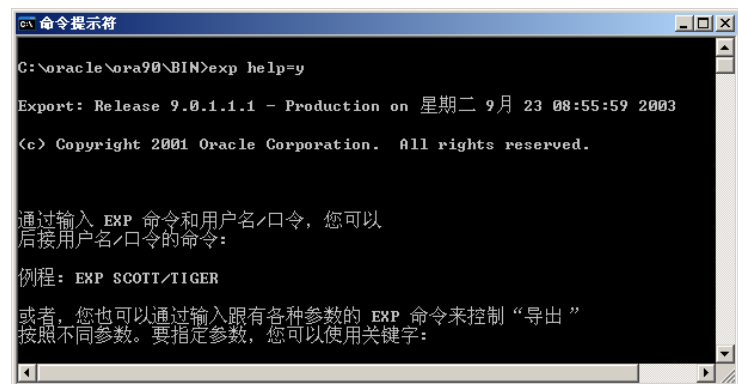


图 12.15 exp 命令的参数

12.4.4 用导出向导实现逻辑备份

(1) 如图 12.16 所示。

(2) 出现如图 12.17 所示的导出向导的【简介】界面。

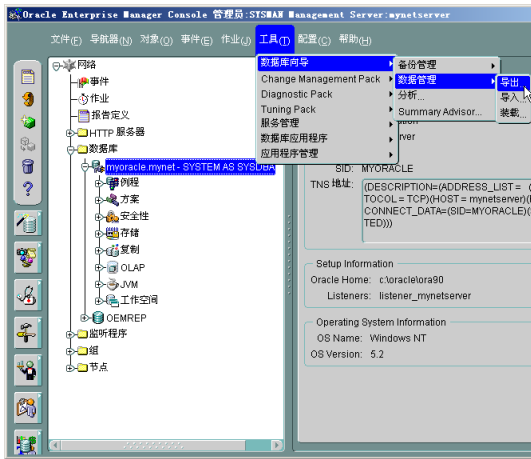


图 12.16 选择使用导出向导



图 12.17 导出向导的【简介】界面

(3) 出现如图 12.18 所示的导出向导的【导出文件】界面。

(4) 出现如图 12.19 所示的导出向导的【导出类型】界面，有 3 种导出类型。



图 12.18 导出向导的【导出文件】界面



图 12.19 导出向导的【导出类型】界面

(5) 出现如图 12.20 所示的导出向导的【关联对象】界面，指定要导出的关联对象。



图 12.20 导出向导的【关联对象】界面

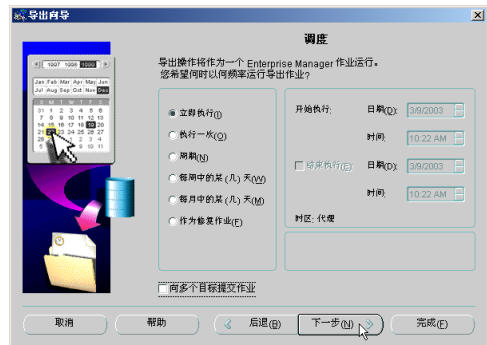


图 12.21 导出向导的【调度】界面

(6) 出现如图 12.21 所示的导出向导的【调度】界面，包括 6 种调度方式。

(7) 出现如图 12.22 所示的导出向导的【作业信息】界面。



图 12.22 导出向导的【作业信息】界面

(8) 出现如图 12.23 所示的导出向导的【概要】界面。

(9) 出现如图 12.24 所示界面。

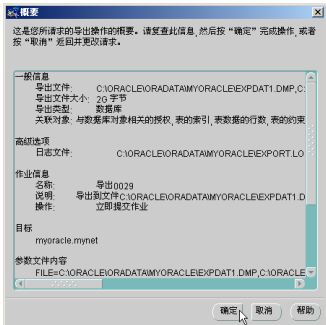


图 12.23 导出向导的【概要】界面



图 12.24 【成功提交导出作业】界面

(10) 成功完成的备份作业如图 12.25 所示。

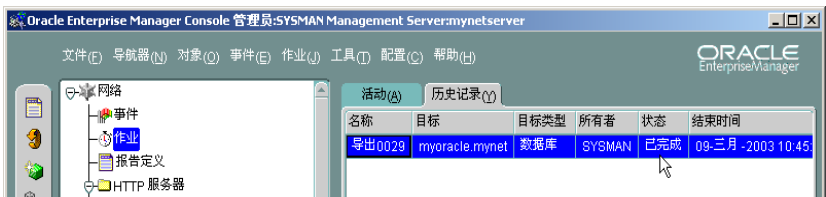


图 12.25 成功完成的备份作业

12.4.5 用导入向导实现逻辑恢复

(1) 如图 12.26 所示。



图 12.26 选择使用导入向导

(2) 出现导入向导的【简介】界面。

(3) 出现如图 12.27 所示的导入向导的【导入文件】界面。



图 12.27 导入向导的【导入文件】界面

(4) 出现如图 12.28 所示的导入向导的【进度】界面。

(5) 出现如图 12.29 所示的导入向导的【导入类型】界面。



图 12.28 导入向导的【进度】界面



图 12.29 导入向导的【导入类型】界面

(6) 出现如图 12.30 所示的导入向导的【用户选择】界面。

(7) 出现如图 12.31 所示的导入向导的【用户映射】界面。



图 12.30 导入向导的【用户选择】界面

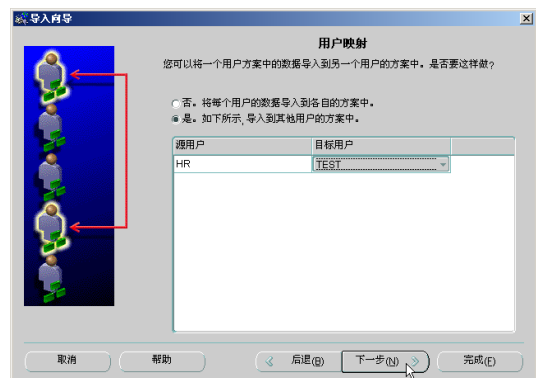


图 12.31 导入向导的【用户映射】界面

(8) 出现如图 12.32 所示的导入向导的【关联对象】界面，用于设置要导入的关联对象，包括。



图 12.32 导入向导的【关联对象】界面

- (9) 出现导入向导的【调度】界面。
- (10) 出现导入向导的【作业信息】界面。
- (11) 出现导入向导的【概要】界面。
- (12) 出现作业成功提交界面。

12.5 联机热备份与恢复实战

12.5.1 联机备份的特点

联机备份又称为热备份，在备份的同时，数据库的用户可以进行操作，因此，数据库对应的物理文件的内容是不断变化的，对这些物理文件内容的更新是保留到有关操作已经写到重做日志文件中后再进行的。

12.5.2 用备份向导实现联机备份

- (1) 如图 12.33 所示。



图 12.33 选择执行备份向导

- (2) 出现如图 12.34 所示的备份向导的【简介】界面。
- (3) 出现如图 12.35 所示的备份向导的【策略选择】界面。



图 12.34 备份向导的【简介】界面



图 12.35 备份向导的【策略选择】界面

(4) 出现如图 12.36 所示的备份向导的【备份选择】界面。

(5) 出现如图 12.37 所示的备份向导的【归档日志】界面。



图 12.36 备份向导的【备份选择】界面



图 12.37 备份向导的【归档日志】界面

(6) 出现如图 12.38 所示的备份向导的【备份选项】界面，有两个选项。

(7) 出现如图 12.39 所示的备份向导的【配置】界面。



图 12.38 备份向导的【备份选项】界面

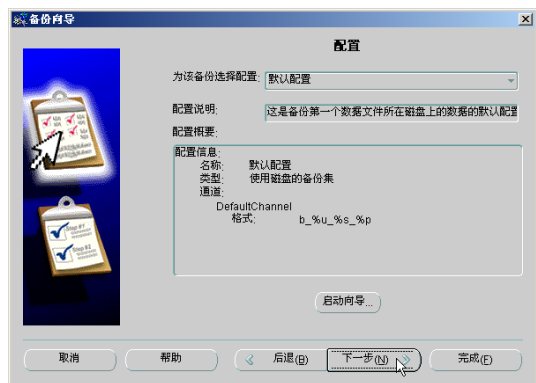


图 12.39 备份向导的【配置】界面

(8) 出现如图 12.40 所示的备份向导的【调度】界面。

(9) 出现如图 12.41 所示的备份向导的【作业信息】界面。



图 12.40 备份向导的【调度】界面



图 12.41 备份向导的【作业信息】界面

(10) 出现备份向导的【概要】界面。

(11) 备份作业成功完成后，在指定的备份目录下出现名为 B_01EHJ8RH_1_1 和 B_02EHJ96S_2_1 的文件，这就是备份后的文件。

12.5.3 用恢复向导实现联机恢复

(1) 数据库必须工作在归档方式的已装载状态，才能执行对整个数据库的联机恢复。

(2) 如图 12.42 所示。



图 12.42 选择执行恢复向导

(3) 出现恢复向导的【简介】界面。

(4) 出现如图 12.43 所示的【恢复选择】界面。

(5) 出现如图 12.44 所示的恢复向导的【复原至】界面。



图 12.43 恢复向导的【恢复选择】界面



图 12.44 恢复向导的【复原至】界面

(6) 出现如图 12.45 所示的恢复向导的【重命名】界面。

(7) 出现如图 12.46 所示的恢复向导的【配置】界面。



图 12.45 恢复向导的【重命名】界面

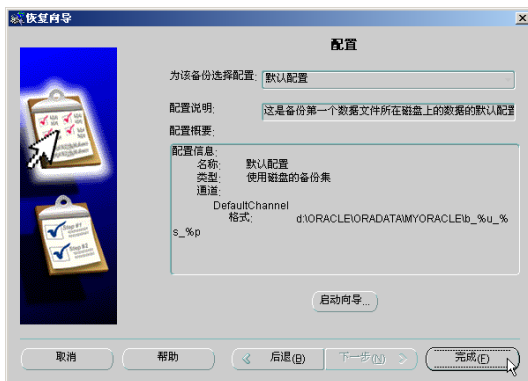


图 12.46 恢复向导的【配置】界面

(8) 出现恢复向导的【概要】界面。

(9) 恢复管理器将自动执行数据库的联机恢复操作。

12.6 习题

(1) 通过试验完成数据库的脱机备份与恢复。

(2) 通过 IMP 和 EXP 命令文件完成逻辑备份与恢复。

(3) 通过集成向导完成逻辑备份与恢复。

(4) 通过集成向导完成联机备份与恢复。

(5) 通过试验比较表空间备份、数据库备份和表备份的操作步骤。

(6) 在利用集成备份和恢复环境完成备份与恢复的操作过程中，可能出现的故障及解决办法是什么？

第 13 章 性能优化——通向 OCP 之路

数据库系统在实际运行过程中，可能会因为各种原因导致性能的下降。对数据库的总体性能进行分析并提出解决的方案也是管理员需要面对的重要问题。本章介绍这些可以用来进行性能调节与分析的工具，以及如何利用这些工具进行性能分析和调整。

13.1 节介绍了常用性能分析工具的使用方法。

13.2 节介绍了数据表性能的优化。

13.3 节介绍了索引性能的优化。

13.4 节介绍了表空间性能的优化。

13.1 性能分析工具的使用

13.1.1 性能规划器的使用

性能规划器（Capacity Planner）是集成在 Oracle 企业管理包（Oracle Enterprise Management Packs）中用来对反映系统性能的参数进行收集的工具，可以指定要收集的数据、收集数据的频率和数据装载到 Oracle Capacity Planner 历史记录数据库的时间。这样便于管理员对一定时间范围内的系统性能参数进行比较分析。

1. 性能规划器的设置

(1) 在服务器的桌面选择【开始】/【程序】/【Oracle-OraHome90】/【Enterprise Management Packs】/【Diagnostics】/【Capacity Planner】选项，将出现如图 13.1 所示的性能规划器登录界面，利用登录管理服务器的用户名和密码就可以正常登录。

(2) 出现如图 13.2 所示的性能规划器的【收集】网络导航树，选择要收集性能数据的数据库 myoracle.mynet，出现【收集选项】选项卡界面。

其中关于【收集范围】的设置包括两个选项。

- ☐ 【Oracle 建议的收集】：可以对历史性能数据进行分析，支持性能诊断和报告。
 - ☐ 【用户自定义收集】：可以对历史性能数据进行分析，但不支持性能诊断和报告。
- 关于【收集采样频率】有两个选项。

【每时间间隔的采样数】：默认是每小时采样 4 次。

【采样时间间隔】：设置采样的时间间隔。

(3) 如图 13.3 所示为性能规划器的【收集】/【存储选项】选项卡界面，收集到的性能



图 13.1 性能规划器的登录

数据形成可在任何 Oracle 数据库中存储的一组数据库表。可以选择将这些数据表存储在安装 Oracle Management Server 时创建的资料档案库（用于保存受管理环境的状态信息）或者是管理员制定的某个数据库中。设置的参数包括。

- ❑ 选择将历史记录存放在资料档案库中或者是指定的数据库中。
- ❑ 设置将样本数据传输到历史数据存放数据库的时间间隔。

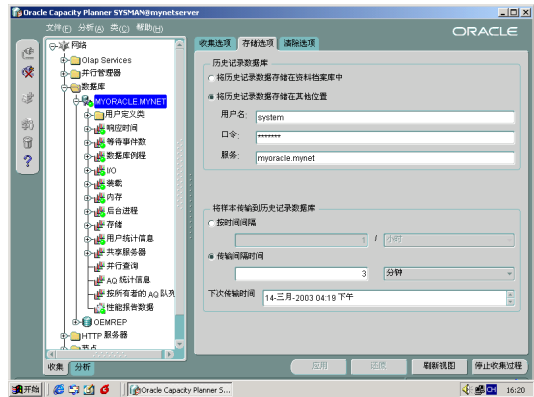
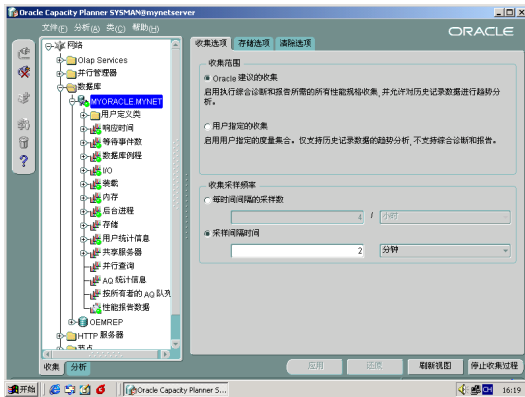


图 13.2 性能规划器的【收集】/【收集选项】选项卡 图 13.3 性能规划器的【收集】/【存储选项】选项卡

(4) 如图 13.4 所示为性能规划器的【收集】/【清除选项】选项卡界面，用于指定保存数据的时间间隔和下次采样时将自动清除某个具体时间之前的数据。

(5) 如图 13.5 所示为性能规划器的【分析】选项卡，由于在图 13.3 所示界面中选择的存储性能参数的数据库在指定的数据库“myoracle.mynet”，因此这里要连接该数据库。

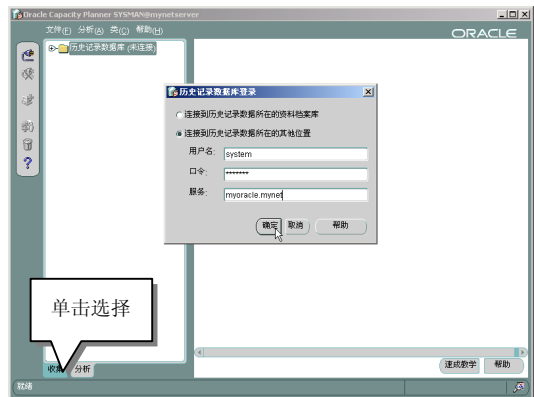


图 13.4 性能规划器的【收集】/【清除选项】选项卡

图 13.5 连接历史记录数据库

(6) 成功连接历史记录数据库后在【管理目标导航器】中选择【历史记录数据库】/【数据库】/【myoracle.mynet】选项，历史记录数据库中存储的有关数据库的统计信息包括以下几类。

- ❑ 响应时间：例程响应时间的统计数据。
- ❑ 等待事件数：包括例程等待事件和等待事件的会话数的统计信息。
- ❑ I/O：包括文件统计信息和例程 I/O 统计信息。
- ❑ 数据库例程：包括例程效率和初始化参数统计信息。
- ❑ 后台进程：包括对重做统计信息、回退段和 DBWR 进程的统计信息。

- ☐ 内存：对数据库字典和共享池的统计信息。
 - ☐ 用户统计信息：对会话动作计数的统计信息。
 - ☐ 装载：包括每秒的例程统计数和每个事务处理的统计数信息。
- 选择每一类下面的具体统计信息，就可以进行历史数据的分析和判断了。


2. 性能规划器的历史数据分析

下面以对【响应时间】/【例程响应时间】的历史统计数据进行分析为例，介绍如何利用性能规划器分析历史数据，并利用该工具对未来的性能进行预测分析。

(1) 单击【响应时间】/【例程响应时间】选项，在出现的【数据源】列表框中选择数据库“myoracle.mynet”，在【数据项】列表框中显示了历史数据中有关例程响应时间的参数。

- ☐ 等待时间的百分比。
- ☐ CPU 时间的百分比。
- ☐ 每个事务处理的响应时间。
- ☐ 提交计数。
- ☐ 回退计数。
- ☐ 每次执行的响应时间。
- ☐ 执行计数。

单击选择要查看的数据项后，单击【显示新图表】按钮，如图 13.6 所示。

(2) 出现如图 13.7 所示的采样数据的性能图表，单击如图所示的按钮系统将按照采样的性能数据提供对未来性能的预测。

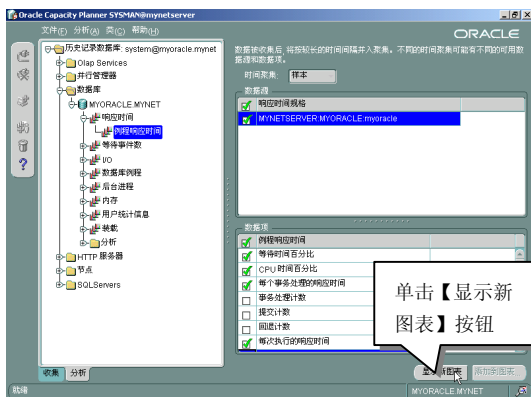


图 13.6 选择以图表方式显示样本性能数据



图 13.7 图表方式显示的样本性能数据

(3) 出现如图 13.8 所示的趋势分析向导的【欢迎使用】界面，单击按钮。

(4) 出现如图 13.9 所示的趋势分析向导的【日期范围】界面，Capacity Planner 将以此日期范围为依据对数据如何随时间变化作出估计。所选的日期范围对趋势分析的结果会产生重要影响。如果估计依据的是系统非常忙的一段很短的时间会得到一个结果。但是，如果估计依据的是很长一段时间（包含系统不忙的时段）内收集的数据，则得到的结果将迥然不同。包括两个选项。

☐ 如果选择【使用为图表选择的日期范围】单选钮，则 Capacity Planner 将分析在分析窗口中当前显示的所有数据，然后生成基于该数据的估计值。

□ 如果要分析特定范围的数据，而不是分析显示在图表中的数据，可选择【选择新的图表日期范围】单按钮。然后可以在两个选项中选择其一：生成基于最近一段时间的估计值和生成基于指定的特定日期范围的估计值。

完成设置后单击  按钮。



图 13.8 趋势分析向导的【欢迎使用】界面

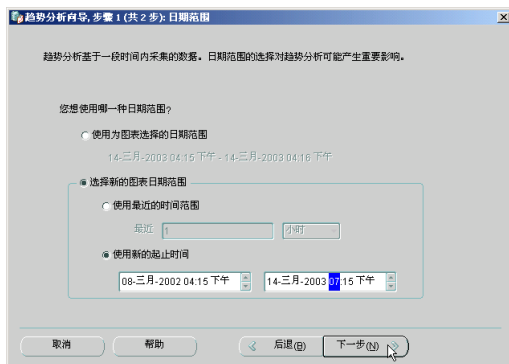
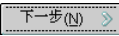


图 13.9 趋势分析向导的【日期范围】界面

(5) 出现如图 13.10 所示的趋势分析向导的【类型】界面，使用趋势分析向导可以为所选的数据项执行两种趋势分析。

□ **【时间点趋势分析】**：为数据项提供目标时间，然后 Oracle Capacity Planner 使用历史记录数据库中的值推测在此目标时间点上该数据项的值。

□ **【值趋势分析】**：为数据项提供目标值，然后 Oracle Capacity Planner 使用历史记录数据库中的值推测数据项将达到目标值的时间。

选择【估计在以下日期数据项将达到的值】单按钮进行值趋势分析，单击  按钮。

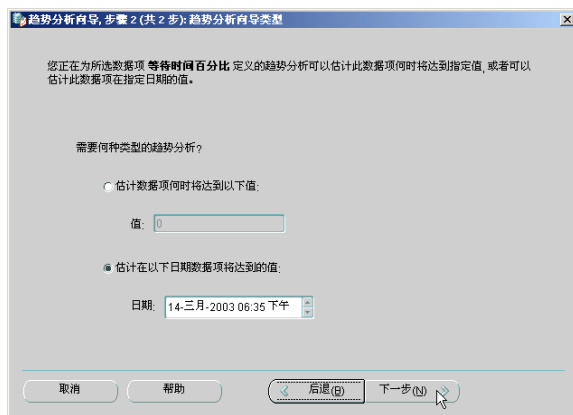

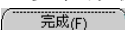


图 13.10 趋势分析向导的【类型】界面

(6) 出现如图 13.11 所示的趋势分析向导的【说明】界面，显示了系统计算后的趋势分析的结果。要在数据显示在图表上之前先修改趋势分析，可单击  按钮，要更新图表，单击  按钮。

(7) 出现如图 13.12 所示的对例程响应时间参数的分析结果，可将分析结果保存下来，也可以将其生成基于 Web 浏览器可以查看的网页。

2. 顶层会话的信息分析

(1) 在图 13.15 所示界面中双击某个会话，将出现如图 13.16 所示的会话的【详细资料】选项卡，主要的信息包括。

- ☐ 会话标识。
- ☐ 当前状态。
- ☐ 会话的 CPU 活动。
- ☐ 会话的内存使用。
- ☐ 会话的 I/O 使用。

(2) 如图 13.17 所示为会话信息的【统计信息】选项卡，以表格的形式显示了该会话的各项性能参数。

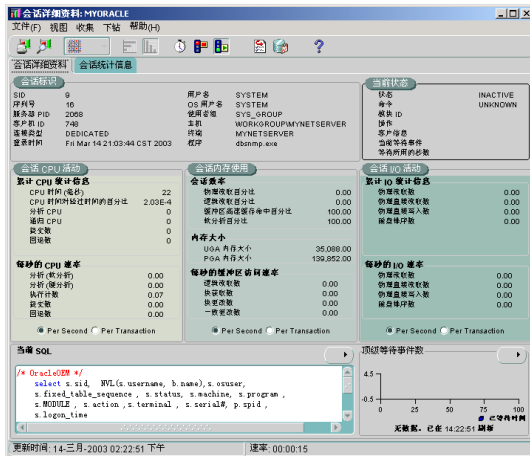


图 13.16 会话信息的【详细资料】选项卡

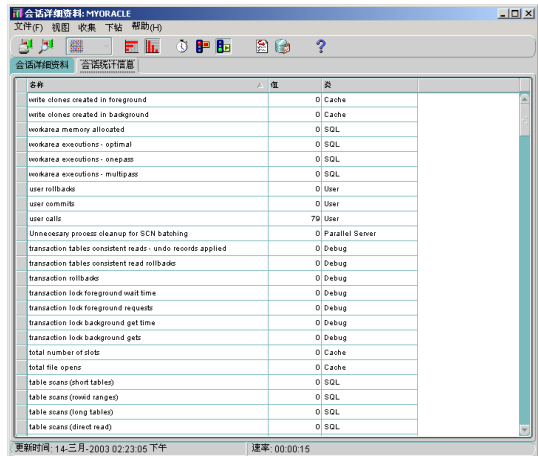


图 13.17 会话信息的【统计信息】选项卡



由于会话信息是由系统定时采样得到的，因此，管理员可以通过设置采样的频率来收集会话的信息，默认为 15 秒。顶层会话工具也可以作为性能管理员的子工具被嵌套使用。

13.1.3 Oracle 专家的使用

Oracle 专家 (Oracle Expert) 是集成在调节包 (Tuning) 中用来对系统性能进行优化和调整的工具。

1. 创建优化会话

(1) 在服务器的桌面选择【开始】/【程序】/【Oracle-OraHome90】/【Enterprise Management Packs】/【Tuning】/【Expert】选项，将出现如图 13.18 所示的 Expert 登录界面。

有两种登录方式。

- ☐ 【登录到 Oracle Management Server】：通过管理服务器进行管理。
- ☐ 【登录到独立的资料档案库】：在没有管理服务器的环境下选用。



图 13.18 Oracle Expert 的登录

选择【登录到 Oracle Management Server】单选按钮，按照管理员登录。

(2) 出现如图 13.19 所示的创建优化会话向导的欢迎界面，优化会话是 Oracle Expert 在优化数据库环境收集分析数据，生成优化建议方案、报告和实施脚本的基本配置。初次使用 Oracle Expert 时需要创建一个新的优化会话。

选择【创建一个新的优化会话】单选按钮，单击 **下一步(N) >** 按钮。

(3) 出现如图 13.20 所示的优化会话向导的创建界面，在【要优化哪个数据库】下拉列表框中选择“myoracle.mynet”，在【如何命名新的优化会话】文本框中输入“新建优化会话”，单击 **完成** 按钮。



图 13.19 优化会话向导的欢迎界面

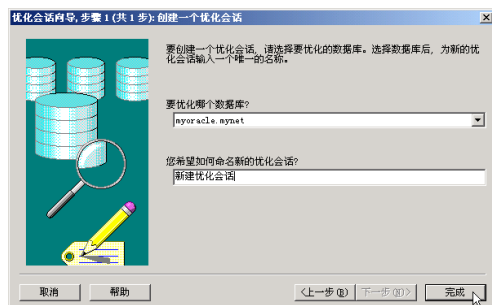


图 13.20 优化会话向导的创建界面

(4) 出现如图 13.21 所示的新建优化会话的【范围】选项卡，用于设置优化的范围和优化会话的特征。

优化范围的主要设置包括。

❑ **【检查例程优化】**：使用此优化范围来确定是否设置了正确的优化参数以及数据库例程是否能够有效利用系统资源。

❑ **【检查 SQL 复用可能性】**：使用此优化范围来确定优化会话工作量是否包含性质相同而语法上稍有不同 SQL 语句。

这样的 SQL 语句必须被单独分析和高速缓存。如果语法上的差异已被排除，那么 Oracle Expert 将高速缓存该语句的单个版本，允许应用程序重复使用该高速缓存的 SQL 语句。

❑ **【检查相应的空间管理】**：使用此优化范围来评估数据库空间管理问题，如表空间结构、方案对象的大小调整和布局，以及数据库用户的表空间分配。

❑ **【检查最佳的数据访问】**：使用此优化会话来优化指定表的索引，并检查需要重建的索引。共有 3 种选项。选择**【对执行性能最差的 SQL 语句引用的表执行综合索引评估】**单选按钮，Oracle Expert 将自动在执行性能最差的 SQL 语句（在优化会话工作量中标识）引用的表中集中进行数据访问优化，优化会话的 SQL 语句将根据每个语句的每次执行的物理读取比率来划分等级，Oracle Expert 也将自动检查目标表中现有索引上的索引碎片。选择**【对**

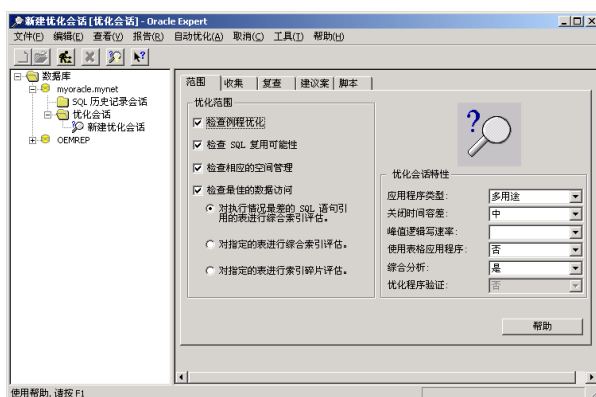


图 13.21 新建优化会话的【范围】选项卡

指定的表进行综合索引评估】单选钮，Oracle Expert 将在指定的特定方案或表中集中进行数据访问优化，Oracle Expert 也将自动检查目标表中现有索引上的索引碎片。如果只希望执行索引碎片检查，则选择【对指定的表进行索引碎片评估】单选钮。Oracle Expert 将只标识目标表中的索引，该目标表必定是遇到了索引滞流，并且需要重建以提高性能。

有关优化会话的特性参数设置包括。

❑ **【应用程序类型】**：向 Oracle Expert 表明数据库环境中使用的工作量类型。这使得 Oracle Expert 可以根据工作量类型来优化数据库。可能的值为：OLTP（OLTP 工作量通常对包含混合读写请求的表使用需要快速响应时间的简单查询）、数据仓库（数据仓库工作量通常对大型的，通常为只读的数据库表使用综合查询）和多用途（多用途工作量通常具有非常宽的响应时间限制，其特征通常是一个或少数几个用户进行大量写密集型的事务处理）。

❑ **【关闭时间容差】**：可以确定系统的建议案将倾向于优化恢复还是优化性能。如果容差很大，Oracle Expert 将优化性能。如果容差很小，Oracle Expert 将优化恢复时间。

❑ **【峰值逻辑写速率】**：向 Oracle Expert 表明最大写入事务处理量，该信息用来评估服务器是否已配置为支持预期的写入事务处理速率。

❑ **【使用的表格应用程序】**：告知 Oracle Expert 是否在数据库环境中使用表格应用程序。Oracle Expert 包含专用于表格应用程序的规则，如为例程设置打开游标的最小数量等。

❑ **【综合分析】**：告诉 Oracle Expert 在当前数据库中有完整的工作量。

❑ **【优化程序验证】**：通知 Oracle Expert 在实施建议案之前对建议案加以验证测试，以确保充分改善了性能。Oracle Expert 建议只实施那些确实能改善性能的建议案。

(5) 图 13.22 所示为新建优化会话的【收集】选项卡，指定要为优化会话收集的数据的类型，包括系统、数据库、例程、方案和工作量 5 类。如果某个收集类被启用，表明该信息对当前优化范围是必需的。如果整行被禁用，则表明当前优化范围不需要该信息。如果收集类选项被启用，但【收集类】复选框未被选择，【上次收集时间】单元格和【选项是否已设置】单元格将被禁用，表示该类信息将不会被收集，但通过选择【收集类】复选框可使它能被收集。

【上次收集时间】单元格显示每类数据上一次收集的日期和时间。

【是否已设置选项】单元格表示某一类是否已准备好可进行收集。

(6) 图 13.23 所示为新建优化会话的【复查】选项卡界面，用于在优化会话和 SQL 历史记录收集的分层视图中查看已收集的数据。

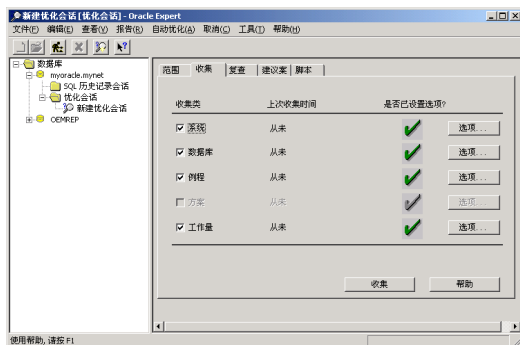


图 13.22 新建优化会话的【收集】选项卡

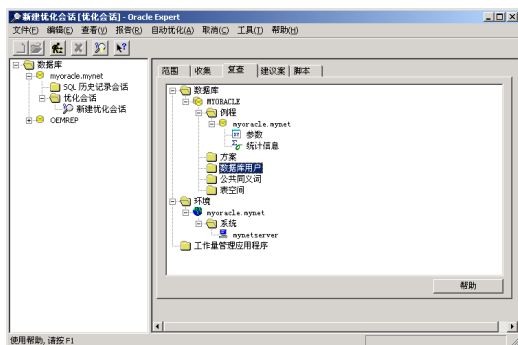


图 13.23 新建优化会话的【复查】选项卡

(7) 图 13.24 所示为新建优化会话的【建议案】选项卡界面，用于复查 Oracle Expert 作

为分析优化会话数据的一部分生成的建议案。单击【生成】按钮系统将自动生成建议方案。

(8) 图 13.25 所示为新建优化会话的【脚本】选项卡界面, 显示有关 Oracle Expert 可创建的、能够帮助实施当前建议案的文件和脚本的说明。同时还显示 Oracle Expert 创建每个文件的位置。

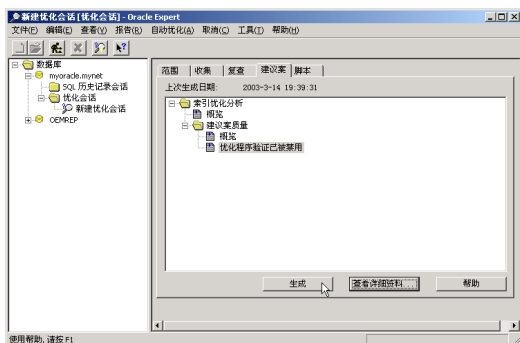


图 13.24 新建优化会话的【建议案】选项卡

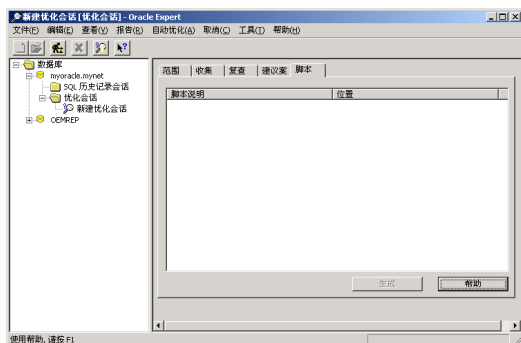


图 13.25 新建优化会话的【脚本】选项卡

2. 分析优化会话

分析优化会话用于由系统自动对创建的优化会话进行分析, 得到一份评估报告。

(1) 在【工具栏】选择【报告】/【分析】选项, 如图 13.26 所示。

(2) 出现如图 13.27 所示界面, 提示将分析报告保存在网页中供打开查看, 单击 **确定** 按钮。

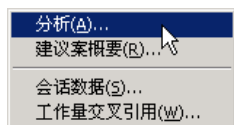


图 13.26 选择对优化会话进行分析

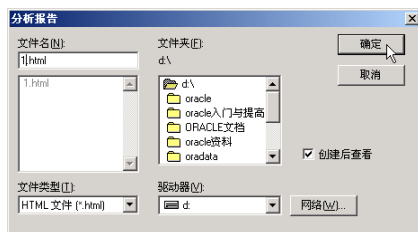


图 13.27 保存分析报告

(3) 打开生成的网页, 系统已经自动生成了分析报告, 主要的方面包括。

- ❑ 建议案概要。
- ❑ 例程分析 (包括数据库例程分析、兼容参数、并行查询参数评估、排序参数评估、SGA 参数评估、操作系统特定参数评估、争用问题评估、共享服务器选项)。
- ❑ 推荐回退段的空闲管理建议案。

3. 系统性能的自动优化

Oracle Expert 提供了对系统性能的自动优化功能。

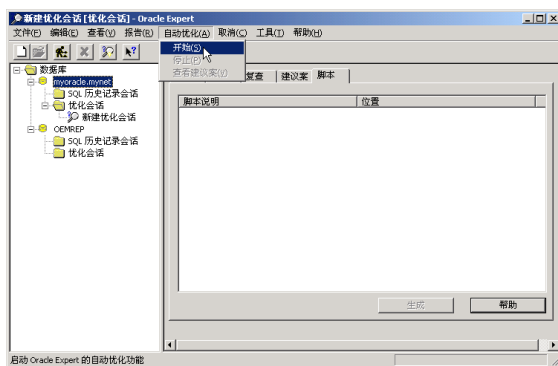


图 13.28 选择对数据库进行性能的自动优化

单击选择数据库 myoracle.mynet，在【工具栏】选择【自动优化】/【开始】选项将执行对 Oracle 9i 数据库系统性能的自动优化，如图 13.28 所示。

13.1.4 索引调节向导

索引调节向导（Index Tuning Wizard）是集成在企业管理器中可以查找并优化数据库中的索引问题。使用索引的目的是要提高对数据库表中数据进行访问的速度。Index Tuning Wizard 将对访问并更新某些表的 SQL 语句进行检查，确定表的索引是否正确编入。对于某些有可能受益于索引优化的表，向导将引导管理员逐步完成评估步骤，并进行所需的更改，以便实施索引建议方案。

(1) 在按照登录【管理服务器】后的【企业管理器】的【管理目标导航器】中选择要进行索引性能分析的数据库“myoracle.mynet”，在【工具栏】选择【工具】/【Tuning Pack】/【Index Tuning Wizard】选项，如图 13.29 所示。

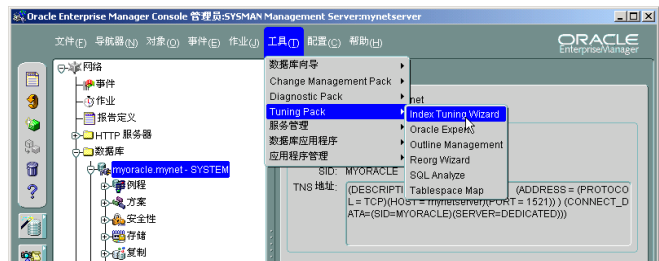


图 13.29 选择执行索引调节向导

(2) 出现如图 13.30 所示的索引调节向导的【欢迎使用】界面，单击 **下一步(N) >** 按钮。

(3) 出现如图 13.31 所示的索引调节向导的【应用程序类型】界面，用于选择目标数据库的应用程序类型，了解数据库应用程序的类型有助于 Index Tuning Wizard 确定要建议的索引的类型和数量。该信息还有助于确定是否使用某种数据库功能。有 3 种选项。

❑ **【联机事务处理 (OLTP)】**：应用程序必须提供快速的最终用户响应时间。OLTP 应用程序的一个典型示例是银行系统，该系统的联机客户账户事务处理要求数据库能快速响应。

❑ **【数据仓库】**：应用程序提供对大量数据的灵活存取。数据仓库应用程序的一个示例是分析人员用来进行产品调查的一个消费品市场数据库。

❑ **【多用途】**：有些数据库可用于多个应用程序类型，包括 OLTP 和数据仓库的组合类型。这里选择【多用途】单选按钮，单击 **下一步(N) >** 按钮。



图 13.30 索引调节向导的【欢迎使用】界面

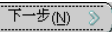


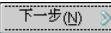
图 13.31 索引调节向导的【应用程序类型】界面

(4) 出现如图 13.32 所示的索引调节向导的【方案选择】界面，用于指定要在何处搜索索引的优化机会，有两个选项。

❑ **【任意方案】:** Index Tuning Wizard 将在数据库中的所有方案中搜索索引优化机会。这是推荐使用的选项，因为如果将所有方案包括在搜索范围内将得到更为有效的索引建议案。

❑ **【已选方案】:** Index Tuning Wizard 将只在已选方案中搜索索引优化机会。如果数据库管理员要将对索引问题的搜索范围限制在其管理的方案范围内，应使用该选项。通过这种搜索得到的索引建议案的效果会较差，因为有许多方案未包括在搜索范围中。

这里选择【任意方案】单选按钮，单击  按钮。

(5) 出现如图 13.33 所示的索引调节向导的【索引建议案】界面，单击【生成】按钮系统将自动对系统使用的索引性能进行分析和评价，该过程可能持续几分钟，具体时间根据具体的方案以及分析过程中涉及的 SQL 语句的数量而定。该操作在执行过程中将显示在“建议案”窗口中，使用该窗口可以监视评估进程的进度。如果没有检查出问题，将显示“未检测到任何索引问题”的信息，单击  按钮。

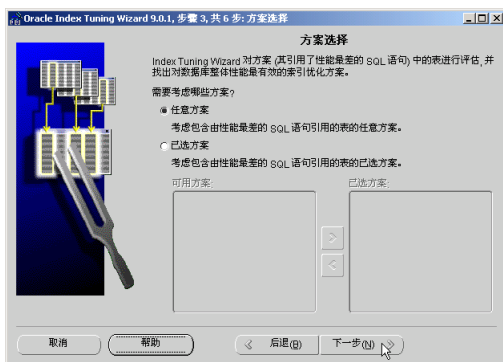


图 13.32 索引调节向导的【方案选择】界面



图 13.33 索引调节向导的【索引建议案】界面

(6) 出现如图 13.34 所示的【分析报告和脚本】界面，生成索引建议案后，可使用 Index Tuning Wizard 来查看、保存和打印所生成的分析报告和相关脚本。

(7) 出现如图 13.35 所示的【完成】界面。



图 13.34 索引调节向导的分析报告和脚本界面




图 13.35 索引调节向导的分析报告和脚本界面

Index Tuning Wizard 为使用索引建议案提供了几个选项。

☐ 【实施建议案】：如果要立即实施索引建议案，则选择该复选框。

☐ 【另存为实施脚本】：如果要创建可供今后复查、修改和实施的 SQL 脚本，则选择该复选框。

☐ 【另存为 Oracle Expert 优化会话】：如果已安装 Oracle Expert，并且要在该 Index Tuning Wizard 会话中创建一个 Oracle Expert 优化会话，则选择该复选框。

这里选择【另存为 Oracle Expert 优化会话】复选框，单击  按钮。



一般而言，当数据库应用程序执行 SQL 的性能下降，或者为数据库开发了新的应用程序，或者修改了现有应用程序的 SQL 语句，都可以执行索引调节向导，用于对基于执行的成本的 Oracle 应用程序进行优化。

13.1.5 SQL 分析的使用

SQL 分析（SQL Analyze）是集成在企业管理器中用于对具体的 SQL 语句进行性能分析的工具。完成同样任务的 SQL 语句，按照不同的语法进行书写，可以得到不同的执行性能。Oracle SQL 分析工具提供了直观观察 SQL 语句执行性能的手段，管理员和程序开发人员可以通过对 SQL 语句不断进行优化来得到最佳的执行方案，从而提高系统的性能。

(1) 在【企业管理器】的【管理目标导航器】中选择要进行索引性能分析的数据库“myoracle.mynet”，在【工具栏】选择【工具】/【Tuning Pack】/【SQL Analyze】选项。

(2) 出现如图 13.36 所示的 Oracle SQL 分析的初始化参数界面。

Oracle SQL 分析工具对每个数据库记录了 3 类信息的 SQL 分析。

☐ 初始化参数：包括例程参数和会话参数。

☐ TopSQL：顶层 SQL，使用系统资源密集的 SQL 语句的分析结果。

☐ SQL 历史记录：所有使用的 SQL 语句的分析结果。

(3) 如图 13.37 所示为 SQL 分析的 TopSQL 界面，单击选择 SQL 文本，将显示该语句对应的性能分析参数，这些参数是 Oracle 数据库系统为 SQL 语句选择执行代价优化的依据。管理员了解这些参数的目的，是尽可能地通过设计使用资源最少的 SQL 语句来完成同样的工作，从而优化系统的性能。



图 13.36 SQL 分析的初始化参数界面

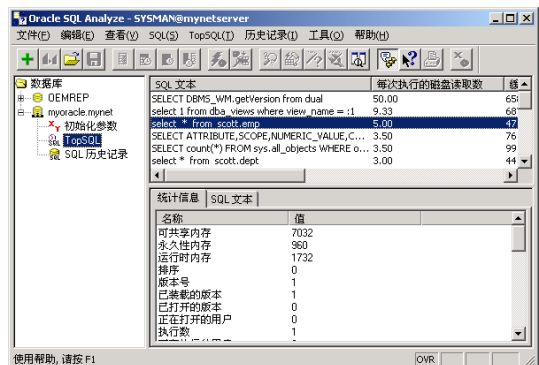


图 13.37 SQL 分析的 TopSQL 界面



管理员可以利用该工具来测试不同的 SQL 语句使用的系统资源，优化程序设计。

13.1.6 锁管理器

当数据库的用户数目越来越多，对服务器的资源将产生竞争，如果没有合理的机制来协调这种资源的竞争，就可能导致一些用户占用大量的资源，而另外一些用户永远得不到其希望的资源，这称为死锁。死锁发生后将大大影响系统的性能。这就犹如在公路上发生了交通事故后，其他汽车也不能或只能绕道行驶，交通系统的性能大大降低。而管理员要做的就是发生事故时，赶紧查明在哪里发生的事故，然后尽快解决。

锁管理器就是 Oracle 提供的对系统资源使用的锁进行监控的工具。通过使用该工具，管理员可以发现哪些用户导致了死锁的发生，从而得出解决的方法。

(1) 在按照登录管理服务器后的企业管理器中，在【管理目标导航树】中选择要进行索引性能分析的数据库“myoracle.mynet”，在【工具栏】选择【工具】/【Diagnostic Pack】/【Lock Monitor】选项。

(2) 出现如图 13.38 所示的锁管理器界面，在【工具栏】选择【下钻】有 4 个选项，可以帮助管理员分析系统资源有无死锁发生。

- ☐ 【历史记录数据】：对历史数据的记录。
- ☐ 【用户类型锁图表】：用户获得的锁。
- ☐ 【阻塞/等待锁图表】：正在等待或处于阻塞/等待状态的锁。
- ☐ 【终止会话】：终止该用户的会话。

以上介绍了一些主要的可以对性能进行分析和调整的集成工具，由于这些工具的功能都被集成在了“性能管理员”里。

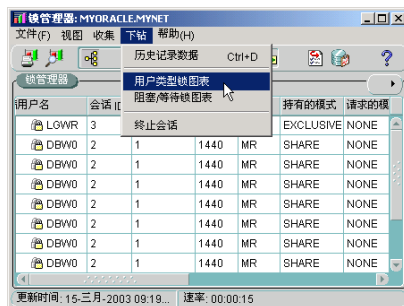


图 13.38 锁管理器

13.1.7 性能管理员

Oracle Performance Manager 是集成在 Oracle 企业管理包（Oracle Enterprise Management Packs）中的性能分析和调整工具，用于对节点、数据库和 HTTP 服务器的资源和操作系统性能进行分析和管理的。

(1) 在服务器的桌面选择【开始】/【程序】/【Oracle-OraHome90】/【Enterprise Management Packs】/【Diagnostics】/【Performance Manager】选项，出现如图 13.39 所示的登录界面。

有两个选项。

- ☐ 【登录到 Oracle Management Server】：登录到管理服务器后使用性能管理员。
- ☐ 【独立，无资料档案库连接】：在直接连接数据库时使用，若选择该项，性能管理员界面将出现乱码。

选择【登录到 Oracle Management Server】单选钮，并在【管理员】、【口令】文本框中按照自己的设置输入。

(2) 出现如图 13.40 所示的性能管理员界面，性能管理员可以对下列资源进行性能分析。

- ☐ OLAP Services：对联机分析处理服务性能进行分析。

- ❑ 并行管理器：对并行数据库系统性能进行分析。
- ❑ 数据库：对数据库的性能进行分析。
- ❑ HTTP 服务器：对 HTTP 服务器的性能进行分析。
- ❑ 节点：对节点管理的资源进行性能分析。
- ❑ SQL Servers：对 SQL Server 服务器进行性能分析。



图 13.39 性能管理员的登录界面

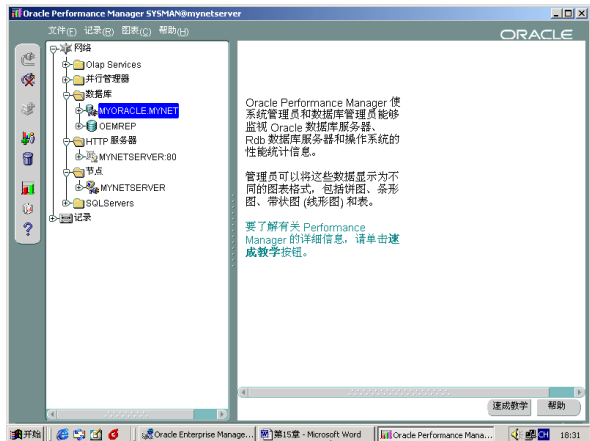


图 13.40 性能管理员

(3) 若出现乱码的现象，可按照下列步骤重新登录数据库。在【管理目标导航器】中选择【网络】/【数据库】/【myoracle.mynet】选项，重新登录数据库，出现如图 13.41 所示的数据库登录界面，单击【连接详细资料】按钮。

(4) 出现如图 13.42 所示的连接详细资料界面，有两个选项。

- ❑ 【直接连接】：客户机直接连接监控资源，容易产生乱码。
- ❑ 【通过 Intelligent Agent 连接】：通过智能代理建立连接，不产生乱码。

选择【通过 Intelligent Agent 连接】单选钮，在【代理主机】文本框中自动出现管理服务器的名称，单击【确定】按钮，这样就可以消除性能管理员管理中出现的乱码现象。



图 13.41 数据库登录界面



图 13.42 选择通过智能代理来连接数据库

13.2 数据表性能优化实例

无论数据表采用了什么样的存储技术，经过一定的时间和事务处理运行后，在存储空间里都会产生一些“碎片”，从而导致性能下降。此外，由于预先估计的不足，很可能会发现已

经定义的表空间的大小不能满足需要等情况，这些问题都是管理员进行数据库的日常管理维护过程中会遇到的问题，解决这些问题的一个有效的办法就是表的重组。

13.2.1 数据表性能下降的原因

1. 迁移的数据行

对数据表的更新操作会引起数据行的扩展，因此当一个数据块中的空闲空间不足以容纳某个数据行时，这行数据将从初始的数据块移到重新分配的数据块中，这就是数据行的迁移，Oracle 9i 系统必须在初始块中保留一个指针指向这一数据行所在的新块的 ID，如果一个数据表中有很多迁移行，为了检索或更新发生迁移的数据行，每次操作必须读取两个数据块，这样表的性能必然会降低。

解决数据行的迁移的一个有效的办法就是设置有效的存储参数值“PCTFREE”，通过对表的数据的分析得出的统计信息可以帮助管理员确定一个合适的 PCTFREE 值。

2. 链接的数据行

如果某个数据行的大小大于数据块的大小，比如 LOB (Large Object, 大对象) 数据类型的数据表就极有可能会产生数据行的链接。在每个数据块里都要设计指针指向下一个数据块的 ID。因此，当链接过多时也会造成性能的下降。

解决数据行的链接问题只能通过优化这些构成同一个数据行的数据块的存储，尽可能将他们存储在物理上相邻的数据块中。

13.2.2 重组的含义

重组，顾名思义，就是结构的重新组织。在 Oracle 9i 中提供了名为 Reorg Wizard (重组向导) 的操作。Reorg Wizard 通过解决空间利用率来帮助管理员维护数据库，使之运行良好，Reorg Wizard 提供了 3 个重组功能。

❑ 重组特定方案对象：可以对特定的方案对象进行重组，尤其是对数据表的重组，可以在表空间之间移动数据表，可以更改表的存储参数，指定新的空闲表管理参数和新的事务处理参数。Oracle 9i 对表的重组是通过创建新的数据段、复制数据、删除旧的数据段 3 个步骤完成的，可以在同一表空间完成，也可以在不同的表空间完成，但表空间的空闲空间要足够大。

❑ 重组整个表空间：主要完成修复表空间的空闲空间碎片以提高性能。

❑ 修复移植行：修复数据表或数据表分区中的迁移数据行。



Oracle 9i 中可重组的方案对象包括：表 (包括含 LOB 列的表)、表范围分区、索引、索引范围分区、索引簇和散列簇。修复移植行功能移植的行已被修复，但并未执行整个对象的完全重组，可以修复移植的行的方案对象包括表和表分区。

13.2.3 利用重组向导进行表的重组

在 Oracle 9i 中表的重组可以通过图形化的重组向导操作来进行。重组操作将由一个【企业管理器】的作业使用 Oracle Agent (代理) 来执行，这就要求执行重组的服务器节点上有 Enterprise Manager Console (企业管理器控制台)、Oracle Management Server (管理服务器) 和

Agent, 而且 Agent 必须运行在数据库重组处的同一节点上。

(1) 以【登录到管理服务器】方式来登录【企业管理器】，在【管理目标导航器】中选择【网络】/【数据库】/【myoracle.mynet】/【方案】/【表】/【HR】/【EMPLOYEES】选项，单击鼠标右键，在弹出的快捷菜单里选择【重组】选项，如图 13.43 所示。

(2) 出现如图 13.44 所示的重组向导的【欢迎使用】的界面，单击 按钮。

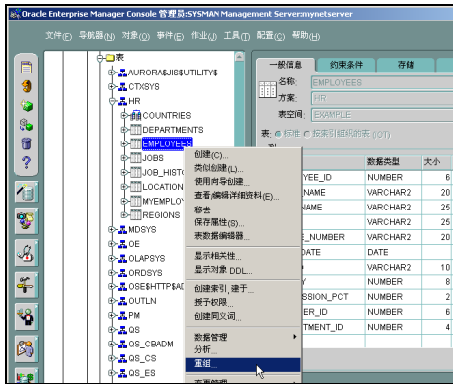


图 13.43 选择执行表的重组

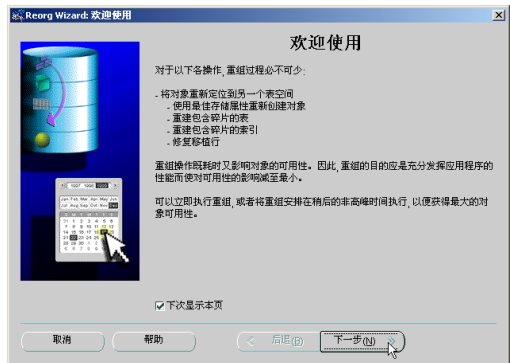


图 13.44 重组向导的【欢迎使用】界面

(3) 出现如图 13.45 所示的重组向导的【对象属性】界面的对象的【一般属性】选项卡。在【对象导航器】中是按照【用户名】/【表】/【表名】/【从属对象】分级组织对象的，单击选择某个对象后，在右边的【对象属性编辑区】里出现该对象的【一般信息】选项卡。

这里选择【表】/【EMPLOYEES】，在【一般信息】选项卡的【表空间】下拉列表框里可以更改数据表所属的表空间。

(4) 切换到如图 13.46 所示的重组向导的【对象属性】界面的对象的【存储】选项卡。可以修改表的数据区的【初始大小】参数、事务处理数量的【初始值】参数、空闲列表的【空闲列表】参数等，设置修改完毕后单击 按钮。



图 13.45 【一般属性】选项卡

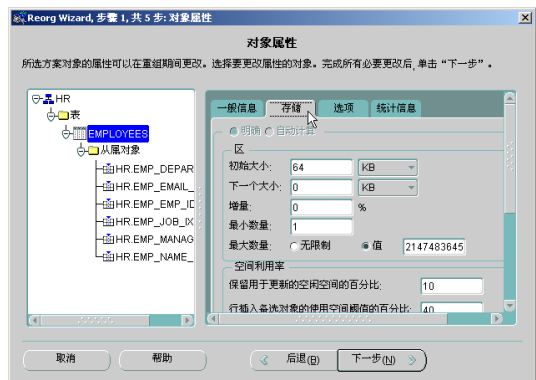


图 13.46 【存储】选项卡



表的重组操作可以更改表空间，但无法修改表的结构，如果选择修改表的结构是无法更改表空间的。

(5) 出现如图 13.47 所示的重组向导的【重组方法】界面，有两种重组的方法供选择：脱机重组和联机重组，两种重组方法比较如下。

❑ 【速度（脱机重组）】单选钮：表示使用脱机重组的方法，侧重于重组的速度，但需要先将数据库脱机，脱机重组的所有操作不记入 REDO 日志文件因此速度较快。

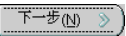
❑ 【可用性（联机重组）】单选钮：表示使用联机重组的方法，侧重于可用性。生成的脚本将在相应服务器可以支持的情况下充分利用联机功能的长处，在重组的过程中，数据表上仍然可以执行数据的查询操作，但不允许有插入、更新和删除操作，仍然保留表的各种存取权限。

这里选择【可用性（联机重组）】单选钮，单击  按钮。

(6) 出现如图 13.48 所示的重组向导的【暂存表空间】界面。重组向导一般通过将数据移到它在数据库中创建的临时对象来执行重组。当重组内容小于整个表空间时，可选择在其当前表空间创建这些临时对象，也可以另外使用一个暂存表空间。使用当前表空间更为快捷，因为对象只移动了一次。但是创建并使用暂存表空间可以避免在当前表空间内进行的重组对空间的影响，有两个选项可供选择。

❑ 【当前表空间】单选钮：对象所在的表空间如果有足够的空间来维护当前对象的两个副本时选择此项，数据只需要移动一次，因此效率最高。

❑ 【暂存表空间】单选钮：若对象所在的表空间没有足够的空间来维护当前对象的两个副本时选择此项，数据需要移动两次。

这里选择【当前表空间】单选钮，单击  按钮。

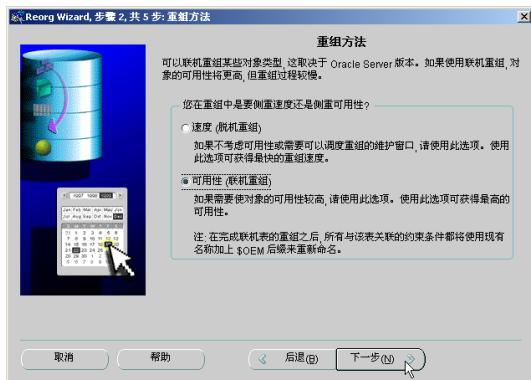


图 13.47 重组向导的【重组方法】界面

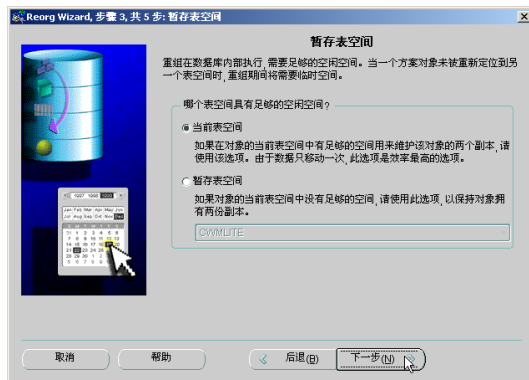



图 13.48 重组向导的【暂存表空间】界面

(7) 出现如图 13.49 所示的重组向导的【效果报告和作业概要】界面，共有两个选项卡。

❑ 【效果报告】选项卡：效果报告提供一个错误列表，或者提供被发现的关于重组作业的其他问题，还提供有关被重组的对象以及将要执行的操作的信息。

❑ 【作业概要】选项卡：生成的作业脚本，包括用来执行重组的数据库命令的概要。实际脚本中既有来自概要的数据库命令，也有执行该作业所需的 Oracle 事务处理语句。

单击  按钮将自动生成效果报告和作业概要。


(8) 重组向导将完成效果报告和作业概要的生成过程，该过程持续时间较长，生成完毕后的界面如图 13.50 所示，单击  按钮。



图 13.49 重组向导的【效果报告和作业概要】界面

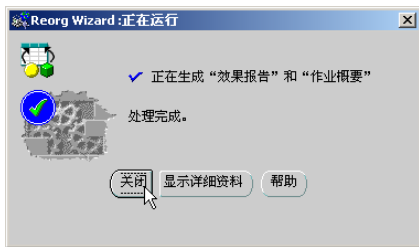


图 13.50 重组向导生成效果报告和作业概要

(9) 在图 13.49 中单击 **下一步(N) >** 按钮，出现如图 13.51 所示的重组向导的【调度】界面。在这里可以用于指定需要运行重组作业的时间。

选择【立即执行】单选按钮表示将作业提交给 Oracle Agent 作业系统并立即执行该作业。

选择【稍后提交】单选按钮可以设置运行作业的日期和时间，如果希望在访问系统的用户较少时运行作业，则该项特别有用，它将重组作业提交给 Oracle Agent，以便在预定的时间执行。这里选择【立即执行】单选按钮，单击 **完成(F)** 按钮。

(10) 出现如图 13.52 所示的重组向导的【概要】界面。概要报告重组向导的设置参数，包括以下 3 类。

- ☐ 一般信息：包括重组选项、暂存表空间、重组方法参数。
- ☐ 作业信息：包括名称、说明、目标、调度参数。
- ☐ 已选对象：包括选择用于重组的对象。

单击 **确定** 按钮。

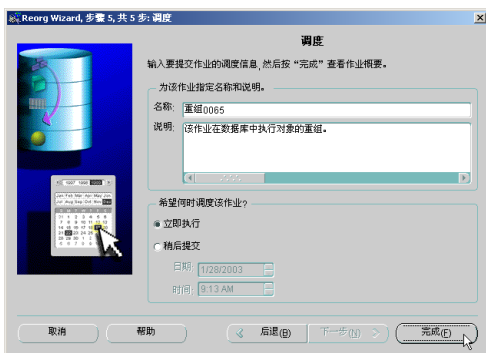


图 13.51 重组向导的【调度】界面

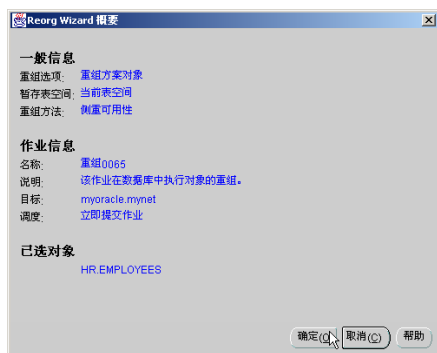


图 13.52 重组向导的【概要】界面

(11) 重组操作最后形成一个作业被提交给 Oracle 9i 的作业系统，由作业系统自动调度完成，成功提交后出现如图 13.53 所示界面。成功提交并不等于作业就能成功执行，如果选择的表空间空闲空间不够或者表空间读写有问题都将导致作业的执行失败。



图 13.53 重组作业成功提交到作业系统

13.3 索引性能优化实例

建立好的索引经过一定时间的执行后，索引块中容易产生碎片从而影响系统的性能。管理员可以对索引块进行压缩，对索引进行重组从而优化索引数据的存储。

下面以对数据表 HR.EMPLOYEES 的索引 EM_DEPARTMENT_IX 进行优化为了介绍优化的步骤。

(1) 以【登录到管理服务器】方式来登录【企业管理器】，在【管理目标导航器】中选择【网络】/【数据库】/【myoracle.mynet】/【方案】/【表】/【HR】/【EMPLOYEES】/【索引】/【EM_DEPARTMENT_IX】选项，单击鼠标右键，在弹出的快捷菜单里选择【重组】选项，如图 13.54 所示。

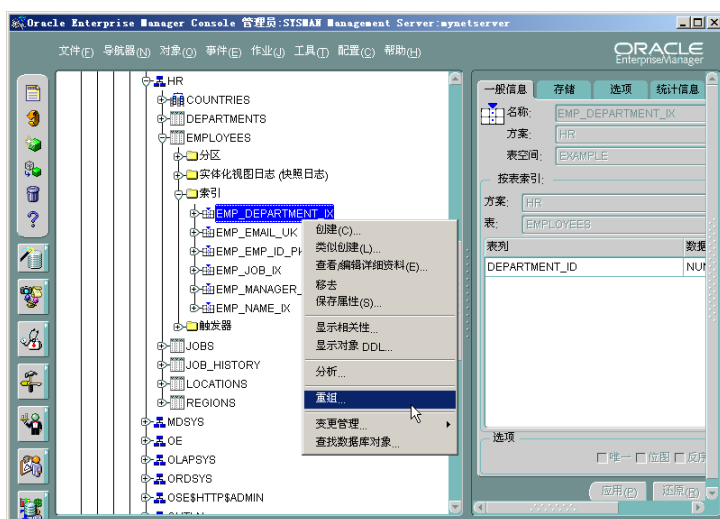


图 13.54 选择执行索引的重组

(2) 出现重组向导的【欢迎使用】的界面，单击 **下一步(N) >** 按钮。

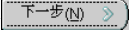
(3) 出现如图 13.55 所示的【对象属性】界面，单击 **下一步(N) >** 按钮。





图 13.55 【对象属性】界面

(4) 出现重组向导的【重组方法】界面。选择【可用性(联机重组)】单选钮，单击 **下一步(N) >** 按钮。


按钮。

(5) 出现重组向导的【暂存表空间】界面。选择【当前表空间】单选钮，单击  按钮。

(6) 出现重组向导的【效果报告和作业概要】界面。单击  按钮将自动生成效果报告和作业概要。

(7) 重组向导将完成效果报告和作业概要的生成过程，该过程持续时间较长，在生成完毕后的界面中单击  按钮。

(8) 出现重组向导的【调度】界面。选择【立即执行】单选钮，单击  按钮。

(9) 出现重组向导的【概要】界面。单击单击  按钮。

(10) 索引重组作业被提交给作业系统后台执行。

13.4 表空间性能优化实例

13.4.1 合并表空间的空闲区

表空间在使用过程中将会出现空闲区，这会影响系统的性能。表空间空闲区的合并就是将相邻的小空闲区合并成完整的大空闲区，其原理如图 13.56 所示。

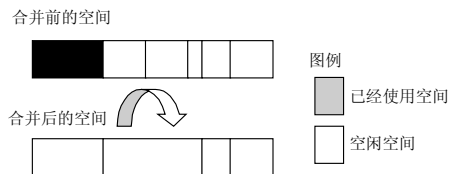


图 13.56 表空间空闲区合并的原理

(1) 以【登录到管理服务器】方式来登录【企业管理器】，在【管理目标导航器】中选择【网络】/【数据库】/【myoracle.mynet】/【存储】/【表空间】/【USERS】选项，单击鼠标右键，在弹出的快捷菜单里选择【显示 Tablespace Map】选项，如图 13.57 所示。

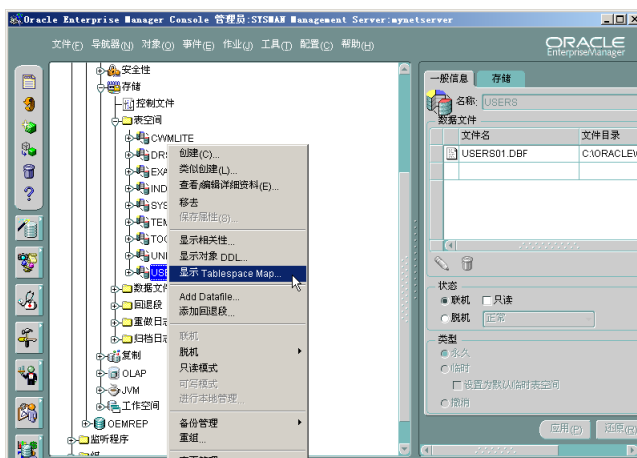


图 13.57 选择显示表空间地图

(2) 出现如图 13.58 所示的表空间地图界面。在【菜单栏】选择【工具】/【合并空闲区】选项。



图 13.58 选择合并表空间空闲区

(3) 出现如图 13.59 所示的表空间空闲区合并界面，将完成以下步骤。

- ☐ 合并空闲区。
- ☐ 刷新表空间和段列表资料。
- ☐ 刷新行链接资料。
- ☐ 刷新索引碎片数据。
- ☐ 检查空间管理问题。
- ☐ 生成并装载表空间分析报告。

(4) 在图 13.60 所示界面中可以查看合并的结果。



图 13.59 表空间空闲区合并



图 13.60 表空间空闲区合并的结果

13.4.2 表空间的重组

与前面介绍的表的重组一样，表空间经过一定时间的运行后，其性能可能受到诸如链接行、迁移行和索引滞流等问题的影响。管理员可以通过重组数据库空间利用率来消除空间问题，同时还可以更改对象的存储设置和位置。目的是通过解决空间利用率问题，由系统来维护数据库，使之运行良好。下面以对 CWMLITE 表空间进行重组操作为例予以介绍。

(1) 在如图 13.57 所示界面中选择【重组】选项，出现如图 13.61 所示的【暂存表空间】界面，表空间重组过程中将把所有的方案对象暂时存放在选定的表空间中，因此一般要求该表空间的容量要足够存放临时的信息。在【哪个表空间具有足够的空闲空间】下拉列表框中选择 USERS，单击 **下一步(N) >**。

(2) 出现如图 13.62 所示的【效果报告和作业概要】界面，单击 **生成...** 按钮系统将自动生

成表空间重组效果报告和作业执行概要。

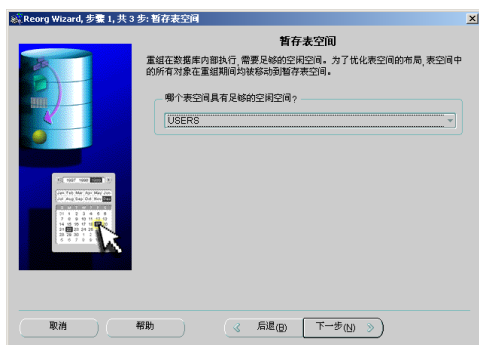


图 13.61 【暂存表空间】界面

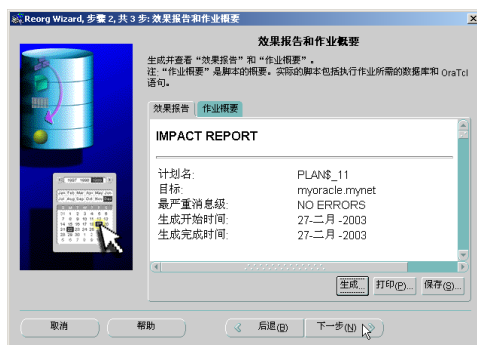


图 13.62 【效果报告和作业概要】界面

(3) 其他步骤与表的重组相同, 这里不再赘述。

13.5 习题

- (1) 通过试验熟悉性能调节分析和调节工具的使用。
- (2) 通过试验熟悉数据表性能的优化。
- (3) 通过试验熟悉索引性能的优化。
- (4) 通过试验熟悉表空间性能的优化。
- (5) 说明链接和迁移行影响数据表性能的原因。

作 者

赵松涛，男，海军少校，工程师。曾经从事高校计算机教学工作 5 年，讲师。后来又改行做计算机网络管理和研发工作，又作了 5 年的工程师。

从 2002 年开始计算机图书写作开始，算起来这是我写的第 5 本书了。每本书从选题、写作、审校到最后出版，我就如一个即将分娩的母亲一样，怀着几许激动，几许期待，等待自己的孩子出生。看见每个字符从自己的指间流出，这种感觉就犹如母亲用自己的乳汁哺育孩子。

写书其实是一种苦差事，一个认真负责的作者好比是蜜蜂，需要将花瓣采摘回来，通过自己的劳动变成蜂蜜。中间的这个过程是快乐而又辛苦的。

写书的原则是“写自己懂的东西，而且一定要是自己经过实践，有收获的东西”。决不为了出书而写书，做一个认真负责的作者。

感谢曾经阅读过我的作品的读者和老师们的热情鼓励和支持，感谢所有曾经来信的读者朋友。

感谢我挚爱的母亲、妻子和所有的亲人们对我的鼓励和支持。

如果您在阅读我所著的书的过程中有疑问、意见、批评，可以通过电子邮件和我联系，我会在最短的时间内及时回复。

与浪共舞：<mailto:dancewithwave@sina.com>

与浪共舞：<mailto:dancewithwave@163.com>

中文版 SQL Server 2000 应用及实例集锦

作者: 赵松涛
书号: 7-115-09726-7
页码: 372
开本: 16 开
定价: 40 元
出版社: 人民邮电出版社
出版日期: 2002-1-1

内容简介

本书围绕 SQL Server 2000 的系统管理和基于 Internet/Intranet 的网络数据库应用开发, 以大量的实例引导读者一步一步轻松管理 SQL Server 2000 数据库系统并建立基于 Web 的数据库应用。本书共分为 3 篇, 第一篇为 SQL Server 2000 管理篇, 介绍 SQL Server 2000 的安装环境、数据库服务器的建立及日常管理、项目数据库的建立及管理、数据库性能监测等。第二篇为基于 ASP 技术的 Web 数据库应用开发, 结合图书销售系统的编程设计介绍目前主流数据库开发技术 ASP 在 SQL Server 2000 上的具体实现。第三篇为基于 JSP 技术的 Web 数据库应用开发, 结合图书销售系统的编程设计介绍 JSP 在 SQL Server 2000 上的具体实现。本书内容丰富, 步骤详尽, 具有很强的实用性。本书适合以前学过 Access、Excel、FoxPro、SQL Server 7.0、Oracle 等数据库, 并希望快速掌握 SQL Server 2000 的读者阅读, 也可供 Internet/Intranet 网络数据库管理及开发人员参考。



Oracle 9i 中文版入门与提高

作者: 赵松涛
定价: 35 元
书号: 7-115-10415-8
页码: 404
开本: 16 开
丛书名: 计算机技术入门与提高系列
出版社: 人民邮电出版社
出版日期: 2002-7-1

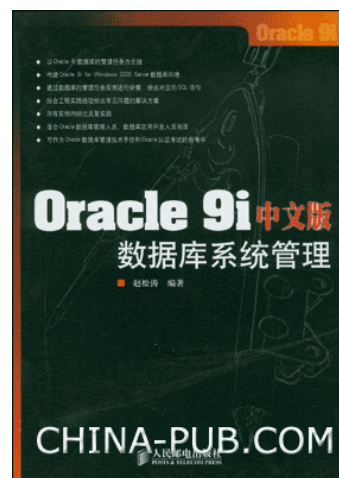


内容简介

本书由浅入深地介绍了 Oracle 9i 数据库的基本概念和管理技术, 简要介绍了通过第三方编程工具进行 Oracle 应用开发的基本思路和方法。第一部分为入门篇 (第 1~6 章), 内容包括 Oracle 9i 基础、Oracle9i 的体系结构、如何构建数据库服务器环境、Oracle 9i 的管理方法、Oracle 9i 的集成管理工具的使用和 SQL 语言基础; 第二部分为提高篇 (第 7~12 章), 内容包括数据库服务器的管理、数据库的管理、数据库备份与恢复、PL/SQL 编程, 以及以 Visual Basic 和 PowerBuilder 为开发工具进行应用开发的基本方法。本书内容丰富, 循序渐进, 书中所有实例全部经过上机反复实践, 具有很强的实用性。本书适合 Oracle 数据库的初学者学习, 可作为 Oracle 数据库技术培训班的教材。

Oracle 9i 中文版数据库系统管理

作者: 赵松涛
 定价: 55 元
 书号: 7-115-11316-5
 页码: 546
 开本: 16 开
 丛书名
 出版社: 人民邮电出版社
 出版日期: 2003-7-1

**内容简介**

本书以 Oracle 9i for Windows 2000 数据库的管理任务为主线, 全面详尽地介绍了 Oracle 9i 数据库系统管理员需要掌握的管理技术。全书分为 7 个部分, 共 15 章, 内容包括 Oracle 9i 基础知识、例程的日常管理、21 种方案对象的管理、数据库安全性管理、存储的管理、数据库和网络的配置、系统备份与恢复、性能分析与调整等等。本书内容全面、思路流畅、实用性强, 所有实例均经过上机反复实践; 对常见的问题提出了解决方案, 具有很强的指导性。本书适合 Oracle 数据库管理人员、数据库应用开发人员阅读和参考, 可作为管理 Oracle 数据库的技术手册以及参加 Oracle 认证考试的参考书。

中文版 Windows Server 2003 网络服务配置案例

作者: 赵松涛 萧卫
 定价: 38 元
 书号: 7-115-11705-5
 页码: 372
 开本: 16 开
 丛书名
 出版社: 人民邮电出版社
 出版日期: 2003-10-1

内容简介

本书以当前 Internet / Intranet 上最流行、最常用的 13 种网络服务为主线, 按照原理、服务器的构建及管理、客户机的配置和使用的思路, 以成熟的案例来帮助读者快速掌握网络服务的配置原理和使用方法, 同时兼顾对目前一些技术解决方案的前瞻性介绍。全书精选的网络服务包括 DNS 服务、WINS 服务、DHCP 服务、Web 服务、E-mail 服务、BQQ 服务、FTP 服务、数字证书服务、SSL 站点、加密的电子邮件、News 服务、路由和远程访问服务。书中所有案例以 Windows Server 2003 为服务器操作系统, 以内嵌的服务器软件为主, 同时精选了目前成熟的第三方服务器软件。本书案例翔实、深入浅出, 所有精选的第三方服务器软件都可以在 Internet 上的一些免费网站下载。本书适合网络管理人员、维护人员和想要构建 Windows Server 2003 网络应用的开发人员阅读, 也可以作为网络技术培训班的教材使用。