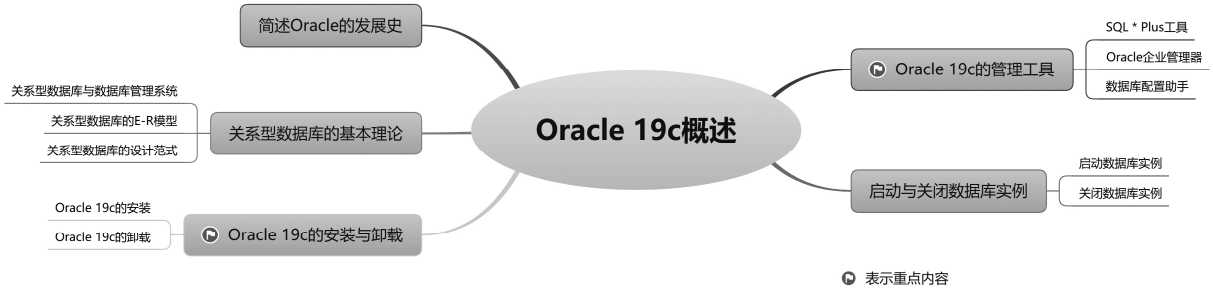
第 1 章

Oracle 19c 概述

本章将首先介绍与关系型数据库密切相关的基础理论知识，然后讲解如何安装和卸载 Oracle 19c 数据库，接着讲解 Oracle 19c 的 3 种常用管理工具，最后为读者介绍如何启动和关闭数据库实例。

本章知识架构及重难点如下。



1.1 Oracle 的发展史

Oracle，西方人认为有“神谕、预言”之意，中国人则译作“甲骨文”，是当今世界上最强大的数据库管理软件之一。所有这一切要从 IBM 的一篇论文谈起。1970 年 6 月，IBM 公司的研究员埃德加 •泰德 • 科德（Edgarh Ted Cod）发表了一篇著名的论文——《大型共享数据库数据的关系模型》。这可以称为数据库发展史上的一个转折点，在当时还是层次模型和网状模型的数据库产品占据市场主要地位的情况下，这篇论文拉开了关系型数据库软件革命的序幕。

1977 年 6 月，拉里 • 埃里森（Larry Ellison）与鲍勃 • 迈纳（Bob Miner）和埃德 • 奥茨（Ed Oates）在硅谷共同创办了一家名为“软件开发实验室”的软件公司（英文缩写 SDL，Oracle 公司的前身）。奥茨看到科德的那篇著名的论文连同其他几篇相关的文章之后非常兴奋，他找来埃里森和迈纳共同阅读，埃里森和迈纳也预见到关系型数据库软件的巨大潜力。于是，数据库界的三位巨人开始共同筹划构建可商用的关系型数据库管理系统（RDBMS），并把这种商用数据库产品命名为 Oracle。因为他们相信，Oracle 是一切智慧的源泉。就这样，堪称当今世界最强大、最优秀的 Oracle 数据库诞生了。

1979 年，“软件开发实验室”更名为“关系软件有限公司”（英文缩写 RSI）。同年夏季发布了可用于 DEC 公司的 PDP-11 计算机上的商用 Oracle 产品（Oracle 第 2 版），这个数据库产品整合了比较完整的 SQL 实现，其中包括子查询、连接及其他特性。

1983 年 3 月，RSI 发布了 Oracle 第 3 版，这个版本是用 C 语言重新编写的。由于 C 编译器具有很

好的可移植性，从此之后，Oracle 产品有了一个关键的特性——可移植性。

1984 年 10 月，Oracle 发布了第 4 版产品，产品的稳定性总算得到了一定的增强，用迈纳的话说，达到了“工业强度”。

1985 年，Oracle 发布了第 5 版，这个版本算得上是 Oracle 数据库诞生以来比较稳定的版本，也是首批可以在 Client/Server 模式下运行的 RDBMS 产品。在技术方向上，Oracle 数据库始终没有落后。

1988 年，Oracle 发布了第 6 版，该版本引入了行级锁（row-level locking）这个重要的特性以及还算不上完善的 PL/SQL（Procedural Language/SQL）语言。此外，该版本还引入了联机热备份功能，使数据库能够在使用过程中创建联机的备份，这极大地增强了可用性。

1992 年 6 月，Oracle 发布了第 7 版，该版本增加了许多新的特性，即分布式事务处理功能、增强的管理功能、用于应用程序开发的新工具以及安全性方法。这一版本才是真正出色的产品，取得了巨大的成功，Oracle 借助这一版本的成功在数据库市场确立了主导地位。

1997 年 6 月，Oracle 发布了第 8 版，该版本支持面向对象的开发及新的多媒体应用，该版本也为支持 Internet、网络计算等奠定了基础，并开始具有同时处理大量用户和海量数据的特性。

1998 年 9 月，Oracle 公司正式发布 Oracle 8i，其中 i 代表 Internet。这一版本中添加了大量为支持 Internet 而设计的特性。此外，这一版本还为数据库用户提供了全方位的 Java 支持。Oracle 8i 成为第一个完全整合了本地 Java 运行时环境的数据库，用 Java 就可以编写 Oracle 的存储过程。

在 2001 年 6 月的 Oracle OpenWorld 大会中，Oracle 发布了 Oracle 9i。在 Oracle 9i 的诸多新特性中，最重要的就是 Real Application Clusters（RAC）——集群技术。

2003 年 9 月 8 日，在旧金山举办的 Oracle OpenWorld 大会上，埃里森宣布下一代数据库产品为 Oracle 10g。Oracle 应用服务器 10g 也将作为 Oracle 公司下一代应用基础架构软件集成套件，g 代表 grid（网格），这一版最大的特性就是加入了网格计算的功能。

2007 年 11 月，Oracle 11g 正式发布。11g 是 Oracle 公司 30 年来发布的最重要的数据库版本，根据用户的需求实现了信息生命周期管理等多项创新，大幅地提高了系统性能的安全性。全新的高级数据压缩技术降低了数据存储的支出，明显缩短了应用程序测试环境部署及分析测试结果所花费的时间，增加了 RFID Tag、DICOM 医学图像、3D 空间等重要数据类型的支持，加强了对 Binary XML 的支持和性能优化。

2013 年 6 月 26 日，Oracle Database 12c 正式发布。像之前版本 10g、11g 中的 g 代表 grid 那样，12c 中的 c 代表 cloud，也就是云计算的意思。

2018 年 2 月 16 日，Oracle 18c 发布，还是秉承着 Oracle 的 Cloud first 理念。

Oracle Database 19c 在 2019 年发布，作为 Oracle Database 12c 和 18c 系列产品的长期支持版本，它能提供最高级别的版本稳定性和最长时间的支持服务和错误修复帮助。

一直以来，Oracle 都以绝对的优势占据着数据库市场的第一位。例如，2019 年主流数据库市场占有率调研中显示，Oracle 占有 56%的市场份额，地位难以撼动，而 IBM 以 15.9%占据第二位，Microsoft 以 9.5%占据第三，其他数据库厂商占有的市场份额很小，如图 1.1 所示。

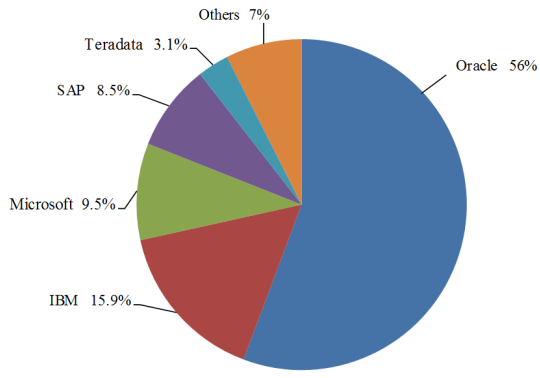


图 1.1　2019 年度主流数据库市场占有率

随着人类社会信息资源的不断增长，需要更加

强大而安全的数据库管理系统，这导致 Oracle 数据库的市场占有率在这些年来不断地增加，其独占鳌头的市场地位是不容置疑的。

1.2 关系型数据库的基本理论

数据库技术是应对信息资源（即大量数据）的管理需求而产生的。信息技术的不断发展，尤其是人类迈入网络时代后，社会信息资源呈爆炸式地增长，对数据管理技术也随之提出更高的要求。数据管理技术先后经历了人工管理、文件系统、数据库系统 3 个阶段。在数据库系统中，数据模型主要有层次模型、网状模型和关系模型 3 种（另一种面向对象模型还处于探索研究中），目前使用最普遍的模型就是关系模型，它是关系型数据库的理论基础。

1.2.1 关系型数据库与数据库管理系统

关系型数据库是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据，现实世界中的各种实体以及实体之间的各种联系均用关系模型来表示。

关系模型以二维表来描述数据。在关系模型中，每张表有多个字段列和记录行，每个字段列有固定的类型属性（如数字、字符、日期等类型）。关系模型中，数据结构简单、清晰，具有很高的数据独立性，因此是目前主流的数据库数据模型。

在关系数据模型中，关系可以看成由行和列交叉组成的二维表格，表中的行称为元组，可以用来标识实体集中的一个实体。表中的列称为属性，给每一列起个名称即为属性名，表中的属性名不能相同。列的取值范围称为域，同列具有相同的域，不同的列也可以有相同的域。表中任意两行（元组）不能相同。能唯一标识表中不同行的属性或属性组（即多个属性的组合）称为主键或复合主键。

关系与传统的二维表格数据文件具有类似之处，但是它们又有区别。严格地说，关系是一种规范化的二维表格，它具有如下性质。

- ☑ 属性值具有原子性，不可分解。
- ☑ 没有重复的元组，即没有重复的行。
- ☑ 理论上没有行序，但是在使用时有时可以有行序。

在关系型数据库中，关键码（简称键）是关系模型的一个非常重要的概念，它通常是行（元组）的一个或几个列（属性）。如果键是由一个列组成的，则称之为唯一键；如果是由多个列（属性）组成的，则称之为复合键。键的主要类型如下。

- ☑ 超键：在一个关系中，能唯一标识元组的属性或属性集称为关系的超键。
- ☑ 候选键：如果一个属性集能唯一标识元组，且又不含有多余的属性，那么这个属性集称为关系的候选键。
- ☑ 主键：如果一个关系中有多个候选键，则选择其中的一个键为关系的主键。用主键可以实现关系定义中“表中任意两行（元组）不能相同”的约束。
- ☑ 这里以管理学生信息为例，我们在“学生信息表”中设置学号、姓名、性别、年龄、院系、班级等列。在该表中，“学号”能够唯一标识一名学生，因此，把学号作为主键是最佳的选择。

而如果把“姓名”作为主键则会存在问题，因为有可能存在同名的学生。为此，最好创建一个单独的键将其明确地指定为主键，这种唯一标识符在现实生活中很普遍，如身份证号、银行卡号、手机号、发票号等。

☑ 外键：如果一个关系 R 中包含另一个关系 A 的主键所对应的属性组 T，则称此属性组 T 为关系 R 的外键，并称关系 A 为参照关系，关系 R 是依赖关系。为了表示关联，可以将一个关系的主键作为属性放入另一个关系中，第二个关系中的那些属性就称为外键。

这里以商品销售为例，在填写一张商品销售单时，可以将商品销售信息分为两大类：第一类是单据的主体信息（销售主表），如销售单号、销售金额、销售日期、收款人等；第二类是单据的明细信息（销售明细表），如商品序号、商品名称、商品数量等。在数据库的销售主表中通常以“销售单号”作为主键；在销售明细表中，为了标识被销售出去的商品隶属于哪张单据，需要对每一条商品销售记录标明“单据编号”。在这种情况下，销售明细表中的“销售单号”就被称为外键，因为“销售单号”是其所在表以外（主体表）的一个主键。

当出现外键时，主键与外键的列名称可以是不同的，但必须要求它们的值集相同，即销售明细表中出现的“销售单号”一定要和主体表中的值匹配。

对于上面提到的二维表格中存储的数据信息，通常以物理文件的形式存储在磁盘上，这种物理文件称为数据文件。用户通常使用数据库软件与磁盘上的数据文件进行交互，这种数据库软件就称为数据库管理系统（DBMS）。DBMS 是建立在操作系统基础上的，它可以对数据库文件进行统一的管理和控制。用户对数据库提出的访问请求都是由 DBMS 来处理的。另外，DBMS 还提供了多种用于管理数据的实用工具。

1.2.2　关系型数据库的 E-R 模型

在设计关系型数据库时，首先需要为它建立逻辑模型。关系型数据库的逻辑模型可以通过实体和关系组成的图形来表示，这种图形称为 E-R 图，它将现实世界中的实体和实体之间的联系转换为逻辑模型。使用 E-R 图形表示的逻辑模型称为 E-R 模型，一个标准的 E-R 模型由实体、属性和联系 3 部分组成。

1. 实体和属性

实体是一个数据对象，是指客观存在并可以相互区分的事物，如一名教师、一名学生、一名雇员等。每个实体由一组属性来表示，如一名具体的学生拥有学号、姓名、性别和班级等属性，其中学号可以唯一标识该名学生实体。具有相同属性的实体组合在一起就构成实体集——实体的集合，而实体则是实体集中的某一个特例。例如，王同学这个实体就是学生实体集中的一个特例。

在 E-R 模型中，实体用矩形表示，矩形内注明实体的命名。实体名常用以大写字母开头的有具体意义的英文名词来表示，联系名和属性名也采用这种方式。图 1.2 为一个图书档案的 E-R 图。

2. 联系

在实际应用中，实体之间是存在联系的，这种联系必须在逻辑模型中表现出来。在 E-R 模型中，联系用菱形表示，菱形框内写明联系名，并用连接线将有关实体连接起来，同时在连接线的旁边标注上联系的类型。两个实体之间的联系类型可以分为以下 3 类。

☑ 一对一：若对于实体集 *A* 中的每一个实体，在实体集 *B* 中最多有一个实体与之相关，反之亦

然，则称实体集 *A* 与实体集 *B* 具有一对一的联系，可标记联系为 1∶1。

- ☑ 一对多：若对于实体集 *A* 中的每一个实体，在实体集 *B* 中有多个实体与之相关；反之，对于实体集 *B* 中的每一个实体，实体集 *A* 中最多有一个实体与之相关，则称实体集 *A* 与实体集 *B* 具有一对多的联系，可标记联系为 1∶*n*。
- ☑ 多对多：若对于实体集 *A* 中的每一个实体，在实体集 *B* 中有多个实体与之相关；反之，对于实体集 *B* 中的每一个实体，实体集 *A* 中也有多个实体与之相关，则称实体集 *A* 与实体集 *B* 具有多对多的联系，可标记联系为 *m*∶*n*。

例如，一名读者可以有多条图书借还记录，而一条借还记录只能隶属于一名读者，这样“读者档案实体”与“读书借还实体”之间就存在一对多的联系（即 1∶*n*），如图 1.3 所示。

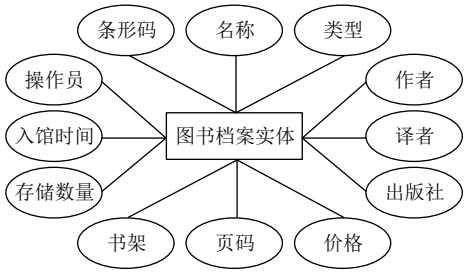


图 1.2　图书档案实体 E-R 图

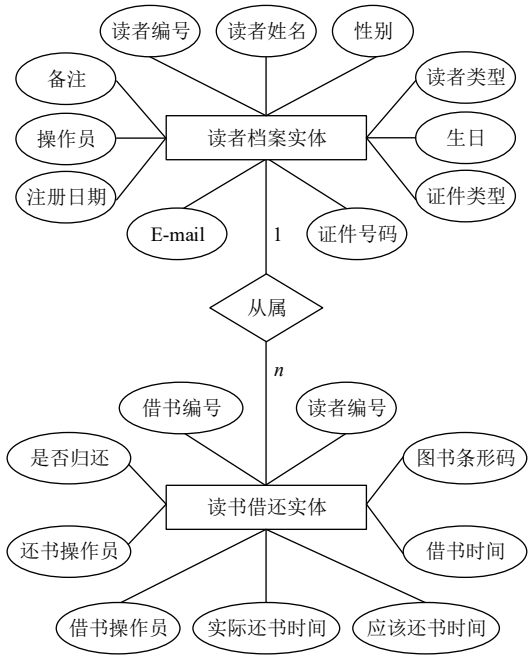


图 1.3　“读者档案实体”与“读者借还实体”之间的联系

1.2.3　关系型数据库的设计范式

在数据库中，数据之间存在着密切的联系。关系型数据库由相互联系的一组关系组成，每个关系包括关系模式和关系值两方面。关系模式是对关系的抽象定义，给出关系的具体结构；关系值是关系的具体内容，反映关系在某一时刻的状态。一个关系包含许多元组（记录行），每个元组都是符合关系模式结构的一个具体值，并且都分属于相应的属性。在关系数据库中的每个关系都需要进行规范化，使之达到一定的规范化程度，从而提高数据的结构化、共享性、一致性和可操作性。

规范化是把数据库组织成在保持存储数据完整性的同时最小化冗余数据的结构的过程。规范化的数据库必须符合关系模型的范式规则。范式可以防止使用数据库时出现不一致的数据，并防止数据丢失。关系模型的范式有第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、第四范式（4NF）、第五范式（5NF）、第六范式（6NF）和 BCNF 范式等多种。通常数据库只要满足前 3 个范式就足够使用，下面举例介绍前 3 个范式。

1. 第一范式（1NF）

第一范式是第二范式和第三范式的基础，是最基本的范式。第一范式包括下列指导原则。

☑ 数据组的每个属性只可以包含一个值。

☑ 关系中的每个数组必须包含相同数量的值。

☑ 关系中的每个数组一定不能相同。

在任何一个关系数据库中，第一范式是对关系模式的基本要求，不满足第一范式的数据库不是关系型数据库。

如果数据表中的每一个列都是不可再分割的基本数据项——同一列中不能有多个值，那么就称此数据表符合第一范式，由此可见第一范式具有不可再分解的原子特性。

在第一范式中，数据表的每一行只包含一个实体的信息，并且每一行的每一列只能存放实体的一个属性。例如，对于学生信息，不可以将学生实体的所有属性信息（如学号、姓名、性别、年龄、班级等）都放在一个列中显示，也不可以将学生实体的两个或多个属性信息放在一个列中显示，而是将学生实体的每个属性信息都放在一个列中显示。

如果数据表中的列信息都符合第一范式，那么在数据表中的字段都是单一的、不可再分的。例如，表 1.1 就是不符合第一范式的学生信息表，因为“班级”列中包含了“系别”和“班级”两个属性信息，这样“班级”列中的信息就不是单一的，它是可以再分的；而表 1.2 就是符合第一范式的学生信息表，它将原“班级”列的信息拆分到了“系别”列和“班级”列中。

表 1.1　不符合第一范式的学生信息表

学　号	姓　名	性　别	年　龄	班　级
9527	东*方	男	20	计算机系 3 班

表 1.2　符合第一范式的学生信息表

学　号	姓　名	性　别	年　龄	系　别	班　级
9527	东*方	男	20	计算机	3 班

2. 第二范式（2NF）

第二范式是在第一范式的基础上建立起来的，即满足第二范式需要先满足第一范式。第二范式要求数据库表中的每个实体（即各个记录行）可以被唯一地区分。为区分各行记录，通常需要为表设置一个区分列，用以存储各个实体的唯一标识。在学生信息表中，设置了“学号”列，由于每名学生的编号都是唯一的，因此每名学生可以被唯一地区分（即使学生存在重名的情况），那么这个唯一属性列被称为主关键字或主键。

第二范式要求实体的属性完全依赖于主关键字，即不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。

例如，这里以员工工资信息表为例，若以员工编码、岗位为组合关键字（即复合主键），就会存在如下决定关系。

（员工编码、岗位）→（决定）（姓名、年龄、学历、基本工资、绩效工资、奖金）

在上述决定关系中，还可以进一步拆分为如下两种决定关系。

```
（员工编码）→（决定）（姓名、年龄、学历）
（岗位）→（决定）（基本工资）
```

其中，员工编码决定了员工的基本信息（包括姓名、年龄和学历），而岗位决定了基本工资，所以这个关系表不满足第二范式。

对于上述这种关系，可以把上述两张关系表更改为如下 3 张表。

☑ 员工信息表：employee（员工编码、员工姓名、年龄、学历）。
☑ 岗位工资表：quarters（岗位、基本工资）。
☑ 员工工资表：pay（员工编码、岗位、绩效工资、奖金）。

3. 第三范式（3NF）

第三范式是在第二范式的基础上建立起来的，即满足第三范式需要先满足第二范式。第三范式要求关系表不存在非关键字列对任意候选关键字列的传递函数依赖，也就是说，第三范式要求一张关系表中不包含已在其他表中包含的非主关键字信息。

所谓传递函数依赖，就是指如果存在关键字段 A 决定非关键字段 B，而非关键字段 B 决定非关键字段 C，则称非关键字段 C 传递函数依赖于关键字段 A。

例如，这里以员工信息表（employee）为例，该表中包含员工编码、员工姓名、年龄、部门编码、部门经理等信息，该关系表的关键字为“员工编码”，因此存在如下决定关系。

```
（员工编码）→（决定）（员工姓名、年龄、部门编码、部门经理）
```

上述这种关系表是符合第二范式的，但它不符合第三范式，因为该关系表内部隐含着如下决定关系。

```
（员工编码）→（决定）（部门编码）→（决定）（部门经理）
```

上述关系表存在非关键字段“部门经理”对关键字段“员工编码”的传递函数依赖。对于上述这种关系，可以把这张关系表（即 employee）更改为如下两个关系表。

员工信息表：employee（员工编码、员工姓名、年龄、部门编码）。

部门信息表：department（部门编码、部门经理）。

对于关系型数据库的设计，理想的设计目标是按照“规范化”原则存储数据的，因为这样做能够消除数据冗余、更新异常、插入异常和删除异常。

1.3 Oracle 19c 的安装与卸载

1.3.1 Oracle 19c 的安装

Oracle Database 19c，是 Oracle Database 12c 和 18c 系列产品的最终版本，因此也是长期支持版本（以前称为终端版本）。“长期支持”意味着 Oracle Database 19c 提供 4 年的高级支持（截至 2023 年 1

月底）和至少 3 年的延长支持（截至 2026 年 1 月底）。

Oracle 19c 数据库服务器由 Oracle 数据库软件和 Oracle 实例组成。安装数据库服务器就是将管理工具、实用工具、网络服务和基本的客户端等组件从安装盘复制到计算机硬盘的文件夹结构中，并创建数据库实例、配置网络和启动服务等。

下面对 Oracle 19c 的安装过程进行详细的说明，具体安装过程如下。

（1）将 Oracle 19c 的安装包文件 WINDOWS.X64_193000_db_home.zip 进行解压缩，在解压后的文件夹中双击 setup.exe 可执行文件，即可安装 Oracle 19c，如图 1.4 和图 1.5 所示。

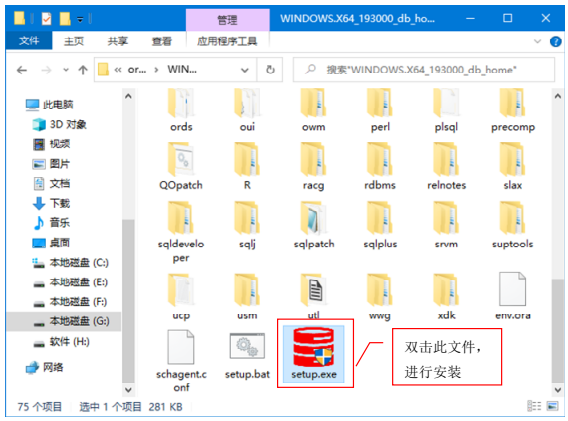


图 1.4　启动 Oracle Universal Installer

图 1.5　启动 Oracle 19c 安装界面

（2）打开安装程序后，进入“选择配置选项”界面，该界面用于选择“安装选项”，这里选中“创建并配置单实例数据库”单选按钮，然后单击“下一步”按钮，如图 1.6 所示。

（3）单击“下一步”按钮后，会打开“选择系统类”界面，如图 1.7 所示。该界面用来选择数据库被安装在哪种操作系统平台（Windows 主要有桌面类和服务器类两种）上，这要根据当前机器所安装的操作系统而定。本演示实例使用的是 Windows 10 操作系统（属于桌面类系统），所以选中“桌面类”单选按钮，然后单击“下一步”按钮。

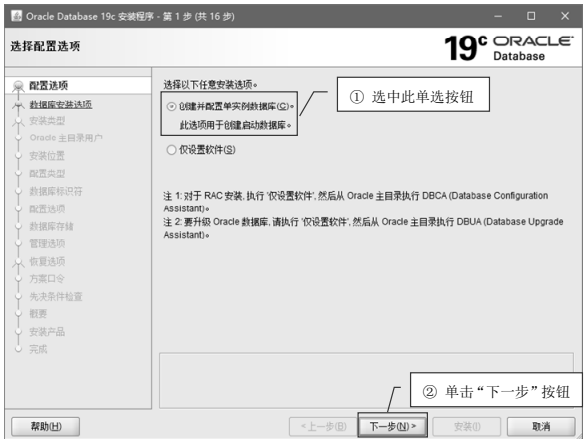


图 1.6　“选择配置选项”界面

图 1.7　“选择系统类”界面

（4）单击“下一步”按钮后，会打开“指定 Oracle 主目录用户”界面。在该界面中，需要指定 Oracle 主目录用户，这里选中“创建新 Windows 用户”单选按钮创建一个新用户，然后单击“下一步”

按钮，如图 1.8 所示。

（5）单击“下一步”按钮后，会打开“典型安装配置”界面。在该界面中首先设置文件目录，默认情况下，安装系统会自动搜索出剩余磁盘空间最大的磁盘作为默认安装盘，当然也可以自定义安装磁盘；接着选择数据库版本，通常选择“企业版”即可；然后输入“全局数据库名”和登录密码（需要记住，该密码是 system、sys、sysman、dbsnmp 这 4 个管理账户共同使用的初始密码。另外，用户 scott 的初始密码为 tiger），其中“全局数据库名”也就是数据库实例名称，它具有唯一性，不允许出现两个重复的“全局数据库名”；再取消选中“创建为容器数据库”复选框；最后单击“下一步”按钮，如图 1.9 所示。

图 1.8 “指定 Oracle 主目录用户”界面

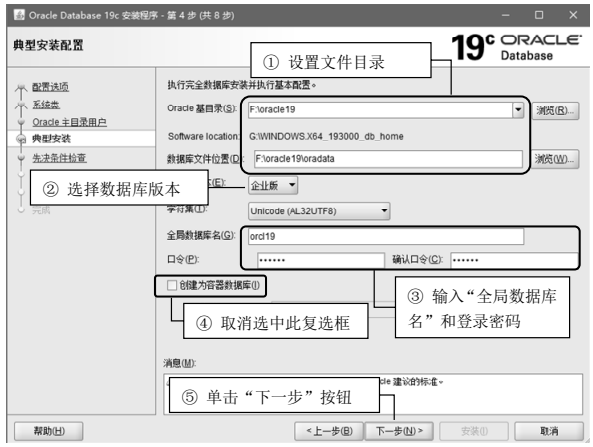


图 1.9 “典型安装配置”界面

一般将全局数据库名设置为 orcl，因为笔者电脑中已有名为 orcl 的全局数据库名，为了避免重名，将 Oracle 19c 的全局数据库名设置为 orcl19。

在“口令”和“确认口令”后输入一样的密码，即为 system 账户的密码。此为本书中设置的密码，读者可自行设置此密码。由于此口令过于简单，单击“下一步”按钮之后，会出现如图 1.10 所示的确认口令界面，在此界面单击“是”按钮。

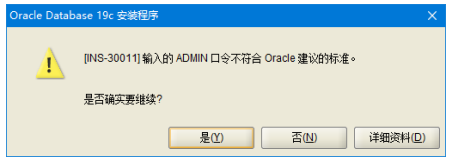


图 1.10 确认口令

（6）接下来会打开“执行先决条件检查”界面，该界面用来检查安装本产品所需要的最低配置，检查结果如图 1.11 所示。

（7）检查完毕后，弹出如图 1.12 所示的“概要”界面，在该界面中会显示安装产品的概要信息，若在步骤（6）中检查出某些系统配置不符合 Oracle 安装的最低要求，则会在该界面的列表中显示出来，以供用户参考，然后单击“安装”按钮即可。

（8）单击“安装”按钮后，会打开“安装产品”界面，在该界面中会显示产品的安装进度，过程比较缓慢，请耐心等待，如图 1.13 所示。

（9）当“安装产品”界面中的进度条到达 100%后，会出现如图 1.14 所示的“完成”界面，表示 Oracle 19c 已经安装成功，单击“关闭”按钮即可退出安装程序。

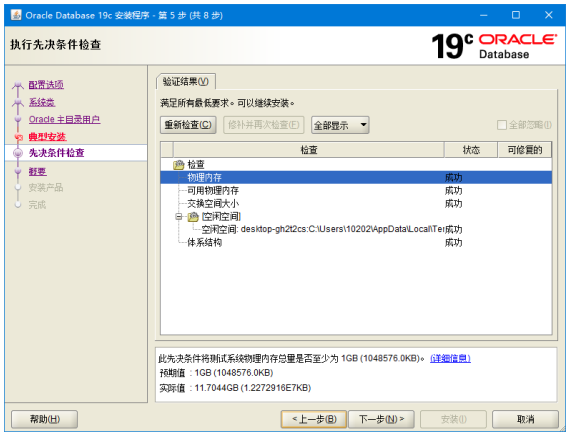


图 1.11　“执行先决条件检查”界面

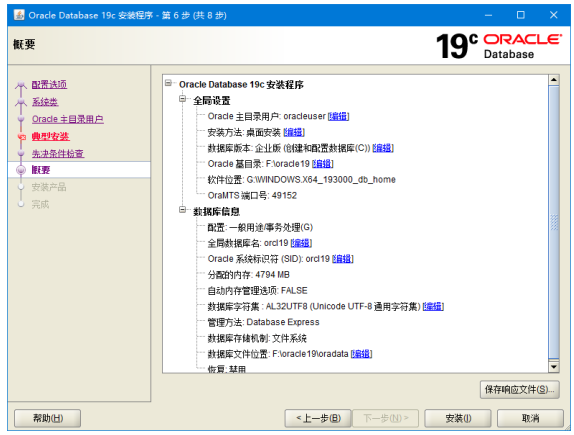


图 1.12　“概要”界面

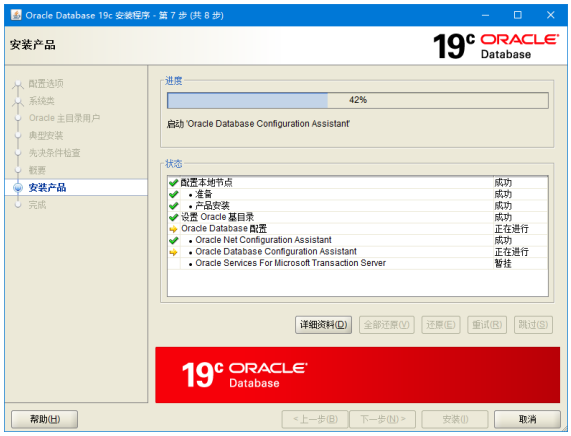


图 1.13　“安装产品”界面

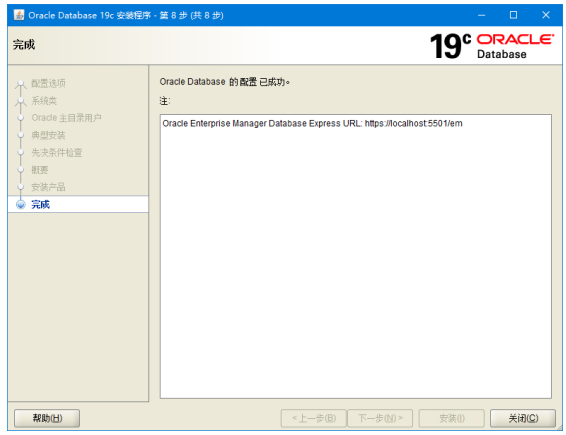


图 1.14　“完成”界面

1.3.2　Oracle 19c 的卸载

想要在 Windows 10 上卸载 Oracle 19c 数据库，必须手动删除所有以 Oracle 开头的注册表现、文件和文件夹，具体操作步骤如下。

（1）右击 Windows 10 系统的“此电脑”，在快捷菜单中选择“管理”命令，打开“计算机管理”界面。在左侧的列表项中，单击“服务和应用程序”前的“>”展开符号，在其下面选择“服务”选项，打开“服务”窗口，如图 1.15 所示。在 Oracle 19c 相关服务上右击，在弹出的快捷菜单中选择“停止”命令。

（2）删除以 Oracle 开头的注册表项。右击开始菜单，选择“运行”选项，然后输入 regedit，单击“确定”按钮，打开“注册表编辑器”窗口。

① 在 HKEY_LOCAL_MACHINE\SOFTWARE\Oracle 上右击，然后在弹出的快捷菜单中选择“删除”命令，如图 1.16 所示。

② 在 HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle 上右击，然后在弹出的快捷菜单中选择“删除”命令，如图 1.17 所示。

③ 在 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Oracle*上右击，然后在弹出的快捷菜单中选择“删除”命令，如图 1.18 所示。

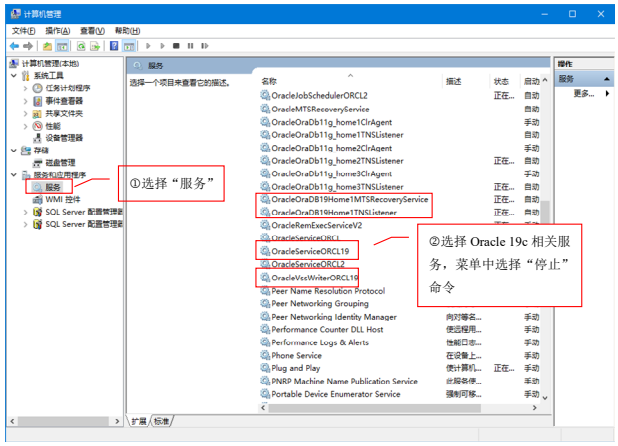


图 1.15　停止 Oracle 19c 相关所有的后台服务

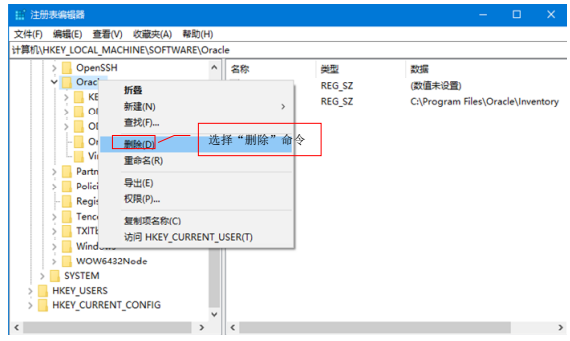


图 1.16　删除 HKEY_LOCAL_MACHINE\SOFTWARE\Oracle 下的注册表

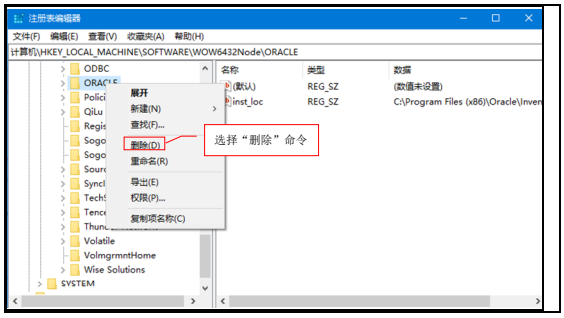


图 1.17　删除 HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle 下的注册表

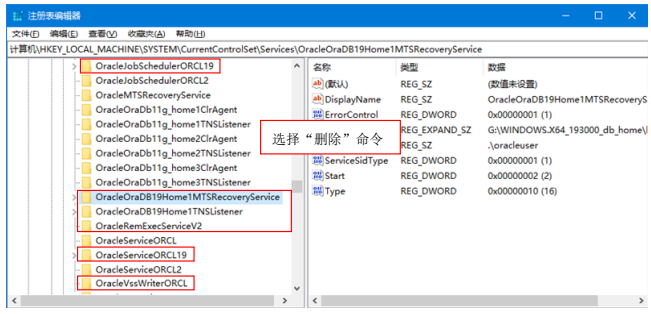


图 1.18　删除 Oracle 19c 相关的注册表

删除完注册表之后，重新启动 Windows 系统。

（3）删除以 Oracle 开头的文件夹 F:\oracle19、C:\Users\oracleuser 和 C:\Program Files\Oracle。

完成以上步骤后，Oracle 19c 就从 Windows 系统中卸载了。

1.4　Oracle 19c 的管理工具

Oracle 19c 提供了多种数据库管理工具，这里主要介绍常用的 SQL*Plus、Oracle 企业管理器（Oracle enterprise manager，OEM）和数据库配置助手（database configuration assistant）。

1.4.1　SQL*Plus 工具

在 Oracle 19c 数据库系统中，用户对数据库的操作主要是通过 SQL*Plus 来完成的。SQL*Plus 作为 Oracle 的客户端工具，既可以建立位于数据库服务器上的数据连接，也可以建立位于网络中的数据连接。

1. 启动 SQL*Plus

下面将介绍如何启动 SQL*Plus 和如何使用 SQL*Plus 连接到数据库。

（1）选择“开始”\Oracle-OraDb19c_home1\SQLPlus 命令，打开如图 1.19 所示的 SQL*Plus 启动界面。

（2）在命令提示符的位置输入登录用户名（如 system 或 sys 等系统管理账户）和登录密码（密码是在安装或创建数据库时指定的），若输入的用户名和密码正确，则 SQL*Plus 将可连接到数据库，如图 1.20 所示。

图 1.19　SQL*Plus 启动界面

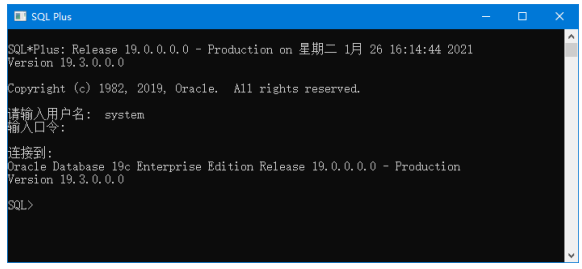


图 1.20　使用 SQL*Plus 连接到数据库

另外，还可以通过在“运行”中输入 cmd 命令来启动命令行窗口，然后在该窗口中输入 SQL*Plus 命令来连接到数据库，如图 1.21 和图 1.22 所示。使用 SQL*Plus 命令连接到数据库实例的语法格式如下。

```
SQLPLUS username[/password][@connect_identifier] [AS SYSOPER|SYSDBA]
```

- ☑ username：表示登录用户名。
- ☑ /password：表示登录密码。
- ☑ @connect_identifier：表示连接的全局数据库名，若连接本机上的默认数据库，则可以省略。

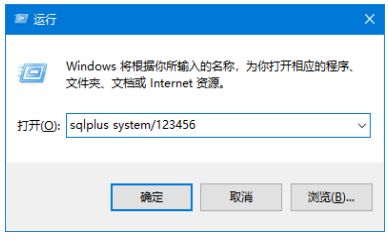


图 1.21　使用 SQL*Plus 命令连接到数据库实例

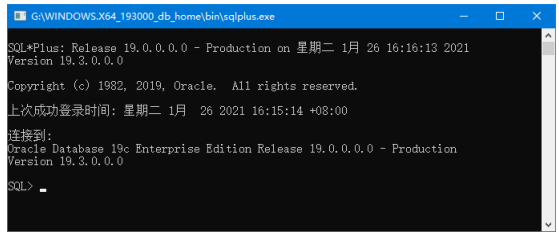


图 1.22　通过命令启动的 SQL*Plus 命令行窗口

说明

在输入 Oracle 数据库命令时，其关键字不区分大小写（例如，输入 sqlplus 或 SQLPLUS 都可以），但参数区分大小写。

2. 使用 SQL*Plus 连接 scott 用户

scott 用户是 Oracle 数据库系统中非常常用的用户，用户名为 scott，密码为 tiger。scott 用户中包含员工信息表 emp、部门信息表 dept、奖金表 bonus 和工资等级表 salgrade，本书中的大多数实例操作的就是这 4 张表。

但是在 Oracle 19c 中并不存在 scott 用户，所以需要自行创建，下面演示如何创建 scott 用户，并创建 scott 用户中的数据表。

（1）打开 SQL*Plus 之后，在“请输入用户名：”后输入 scott，在“输入口令：”后输入 tiger，按 Enter 键之后，结果如图 1.23 所示。。

（2）通过图 1.23 可知，不能连接 scott 用户，所以首先以 sysdba 的身份连接数据库（用户名为“sqlplus /as sysdba”，输入口令后直接按 Enter 键，即可连接 sys 数据库），然后创建 scott 用户，命令如下。

```
sqlplus /as sysdba
create user scott identified by tiger;
```

执行结果如图 1.24 所示。

图 1.23　不能连接数据库

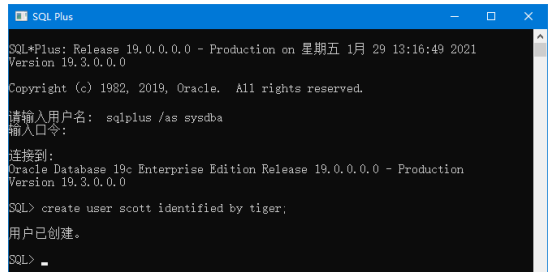


图 1.24　创建 scott 用户

（3）设置用户使用的表空间，命令如下。

```
ALTER USER scott DEFAULT TABLESPACE USERS;
ALTER USER scott TEMPORARY TABLESPACE TEMP;
```

执行结果如图 1.25 所示。

（4）为 scott 用户赋予权限，并使用此用户登录，命令如下。

```
GRANT dba TO scott;
CONNECT scott/tiger;
```

执行结果如图 1.26 所示。

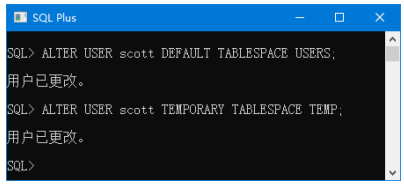


图 1.25　设置用户使用的表空间

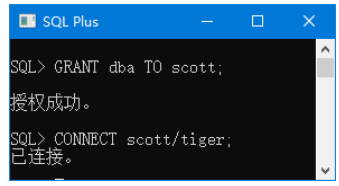


图 1.26　为 scott 用户赋予权限，并使用此用户登录

（5）输入以下代码，创建部门信息表 dept、员工信息表 emp、奖金表 bonus 和工资等级表 salgrade，并插入测试数据。

```
-- 创建数据表
CREATE TABLE dept (
    deptno    NUMBER(2) CONSTRAINT PK_DEPT PRIMARY KEY,
    dname     VARCHAR2(14) ,
    loc       VARCHAR2(13)
```

```
) ;
CREATE TABLE emp (
    empno     NUMBER(4) CONSTRAINT PK_EMP PRIMARY KEY,
    ename     VARCHAR2(10),
    job         VARCHAR2(9),
    mgr         NUMBER(4),
    hiredate     DATE,
    sal         NUMBER(7,2),
    comm     NUMBER(7,2),
    deptno     NUMBER(2) CONSTRAINT FK_DEPTNO REFERENCES DEPT
);
CREATE TABLE bonus (
    enamE     VARCHAR2(10)     ,
    job         VARCHAR2(9)   ,
    sal         NUMBER,
    comm     NUMBER
) ;
CREATE TABLE salgrade (
    grade         NUMBER,
    losal         NUMBER,
    hisal         NUMBER
);
-- 插入测试数据 —— dept
INSERT INTO dept VALUES     (10,'ACCOUNTING','NEW YORK');
INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');
INSERT INTO dept VALUES     (30,'SALES','CHICAGO');
INSERT INTO dept VALUES     (40,'OPERATIONS','BOSTON');
-- 插入测试数据 —— emp
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,
to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,
to_date('22-2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,
to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,
to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,
to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,
to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,
to_date('13-07-87','dd-mm-yyyy')-85,3000,NULL,20);
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,
to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,
to_date('13-07-87','dd-mm-yyyy')-51,1100,NULL,20);
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);
```

```
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,
to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);
-- 插入测试数据 —— salgrade
INSERT INTO salgrade VALUES (1,700,1200);
INSERT INTO salgrade VALUES (2,1201,1400);
INSERT INTO salgrade VALUES (3,1401,2000);
INSERT INTO salgrade VALUES (4,2001,3000);
INSERT INTO salgrade VALUES (5,3001,9999);
-- 事务提交
COMMIT;
```

以上代码执行完毕之后，为了验证是否成功连接到系统的 scott 用户，可以通过在 SQL *Plus 中查询部门表的所有信息（dept 表）来进行验证。使用 scott 用户连接 Oracle 后，在提示符“SQL>”后输入如下语句。

```
SELECT * FROM dept;
```

执行结果如图 1.27 所示。

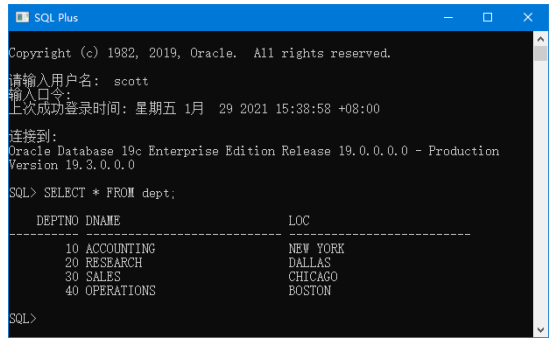


图 1.27　查看 dept 表中数据

1.4.2　Oracle 企业管理器

Oracle 企业管理器（Oracle enterprise manager，OEM）是基于 Web 界面的 Oracle 数据库管理工具。启动 Oracle 19c 的 OEM 只需要在浏览器中输入其 URL 地址——通常为 https://localhost:5501/em（此地址在图 1.14 中可以看到），然后连接主页即可。

如果是第一次使用 OEM，启动 Oracle 19c 的 OEM 后，需要安装“信任证书”或者直接选择“高级”/“继续前往 localhost（不安全）”即可；然后就会出现 OEM 的登录页面，用户需要输入登录用户名（如 system、sys、scott 等）和登录口令，如图 1.28 所示。

在输入用户名和口令后，单击 Log in 按钮，若用户名和口令都正确，就会进入 OEM 的主界面中，如图 1.29 所示。

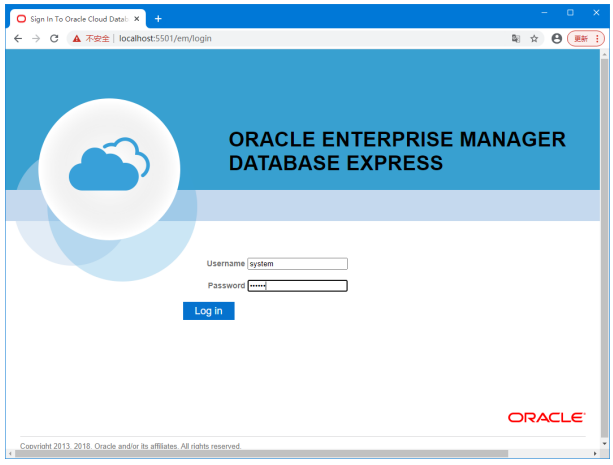


图 1.28　登录 OEM

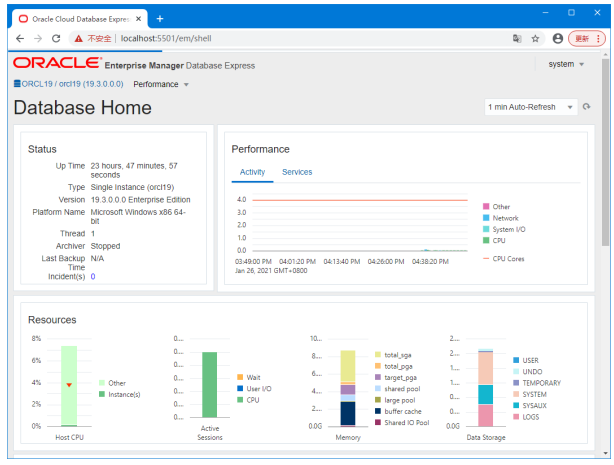


图 1.29　OEM 主界面

OEM 以图形的方式提供用户对数据库的操作，虽然操作起来比较方便简单，不需要使用大量的命令，但这对于初学者来说减少了学习操作 Oracle 数据库命令的机会，而且不利于读者深刻地理解 Oracle 数据库。因此，建议读者强制自己使用 SQL*Plus 工具。另外，本书实例的讲解也主要在 SQL*Plus 中完成，以帮助读者更好地学习 SQL*Plus 命令。

1.4.3　数据库配置助手

在安装 Oracle 19c 数据库管理系统的过程中，若选中“仅设置软件”单选按钮（见图 1.6），则系统安装完毕后，需要手动创建数据库才能够实现对 Oracle 数据库的各种操作。在 Oracle 19c 中，可以通过数据库配置助手（database configuration assistant，DBCA）来实现创建和配置数据库。

选择“开始”\Oracle-OraDb19c_home1\Database Configuration Assistant 命令，将打开如图 1.30 所示的界面。

然后，用户只需要按照数据库配置助手向导的提示逐步进行设置，就可以实现创建和配置数据库。

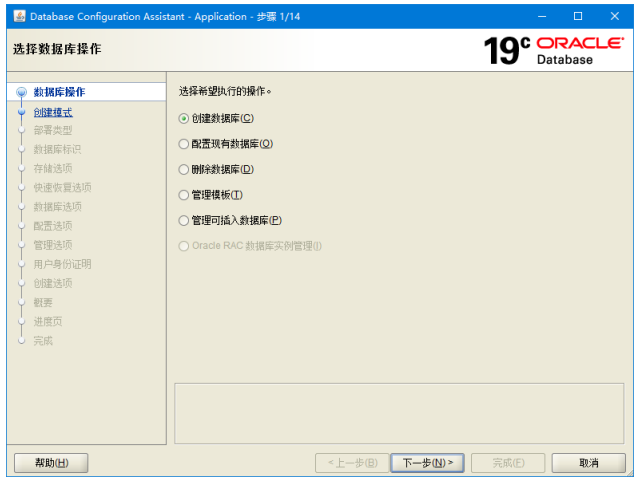


图 1.30　启动数据库配置助手

互动练习：在 scott 模式下，使用 SELECT 语句查询 dept 表中的所有记录。

1.5　启动与关闭数据库实例

1.5.1　启动数据库实例

Oracle 数据库实例的启动过程分为 3 个步骤，分别是启动实例、加载数据库和打开数据库。用户可以根据实际情况的需要，以不同的模式启动数据库。启动数据库使用 STARTUP 命令，语法格式如下。

```
STARTUP [NOMOUNT | MOUNT | OPEN | FORCE] [RESTRICT] [PFILE=filename]
```

- ☑ NOMOUNT：表示启动实例但不加载数据库。
- ☑ MOUNT：表示启动实例，加载数据库，并保持数据库的关闭状态。
- ☑ OPEN：表示启动实例，加载并打开数据库，这个是默认选项。
- ☑ FORCE：表示终止实例并重新启动数据库。
- ☑ RESTRICT：用于指定以受限制的会话方式启动数据库。
- ☑ PFILE：用于指定启动实例时所使用的文本参数文件，filename 就是文件名。

Oracle 数据库实例在启动时必须读取一个初始化参数文件，以便从中获得有关实例启动的参数配置信息。若在 STARTUP 语句中没有指定 PFILE 参数，则 Oracle 首先读取默认位置的服务器初始化参数文件 spfile，若没有找到默认的服务器初始化参数文件，则将读取默认位置的文本初始化参数文件。下面将分别讲解 STARTUP 语法中列举出的几种启动模式。

1. NOMOUNT 模式

NOMOUNT 启动模式只会创建实例（即创建 Oracle 实例的各种内存结构和服务进程），并不加载数据库，也不会打开任何数据文件。

【例 1.1】启动数据库实例到 NOMOUNT 模式下，代码及运行结果如下。**（实例位置：资源包\TM\sl\1\1）**

```
SQL> connect system/1qaz2wsx as sysdba;
已连接。
SQL> shutdown immediate
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
SQL> startup nomount
ORACLE 例程已经启动。
Total System Global Area      535662592 bytes
Fixed Size                      1375792 bytes
Variable Size                 226492880 bytes
Database Buffers              301989888 bytes
Redo Buffers                    5804032 bytes
```

在上述示例代码中，首先用户要以 sysdba 的身份登录，然后才具有关闭和启动数据实例的权限。这意味着在使用 SHUTDOWN 命令关闭数据库实例之后，可以使用 STARTUP NOMOUNT 命令启动数据库实例。

说明

通常在创建新数据库或重建控制文件时，使用 NOMOUNT 模式启动数据库实例。

2. MOUNT 模式

MOUNT 模式将启动实例，加载数据库并保持数据库的关闭状态。

【例 1.2】启动数据库实例到 MOUNT 模式下，代码及运行结果如下。**（实例位置：资源包\TM\sl\1\2）**

```
SQL> shutdown immediate
```

```
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
SQL> startup mount
ORACLE 例程已经启动。
Total System Global Area      535662592 bytes
Fixed Size                      1375792 bytes
Variable Size                 226492880 bytes
Database Buffers              301989888 bytes
Redo Buffers                    5804032 bytes
数据库装载完毕。
```

上述代码中，首先使用 SHUTDOWN 命令关闭数据库实例，然后使用 STARTUP MOUNT 命令启动数据库实例。

说明

MOUNT 模式通常在进行数据库维护时使用，如执行数据库完全恢复操作、更改数据库的归档模式等。

3. OPEN 模式

OPEN 模式将启动实例，加载并打开数据库。这是常规的启动模式，用户要对数据库进行多种操作，就必须使用 OPEN 模式启动数据库实例。

【例 1.3】启动数据库实例到 OPEN 模式下，代码及运行结果如下。

```
SQL> startup
ORACLE 例程已经启动。
Total System Global Area      535662592 bytes
Fixed Size                      1375792 bytes
Variable Size                 226492880 bytes
Database Buffers              301989888 bytes
Redo Buffers                    5804032 bytes
数据库装载完毕。
数据库已经打开。
```

在上述代码中，startup 命令的后面不带有任何参数，就表示以 OPEN 模式启动数据库实例。

4. FORCE 模式

FORCE 模式将终止实例并重新启动数据库，这种启动模式具有一定的强制性。例如，在其他启动模式失效时，可以尝试使用这种启动模式。

【例 1.4】启动数据库实例到 FORCE 模式下，代码及运行结果如下。

```
SQL> startup force
ORACLE 例程已经启动。
Total System Global Area      535662592 bytes
Fixed Size                      1375792 bytes
Variable Size                 226492880 bytes
Database Buffers              301989888 bytes
```

```
Redo Buffers                5804032 bytes
数据库装载完毕。
数据库已经打开。
```

1.5.2 关闭数据库实例

与启动数据库实例相同，关闭数据库实例也分为 3 个步骤，分别是关闭数据库、卸载数据库和关闭 Oracle 实例。在 SQL*Plus 中，可以使用 SHUTDOWN 语句关闭数据库，其具体语法格式如下。

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT]
```

☑ NORMAL：表示以正常方式关闭数据库。
☑ TRANSACTIONAL：表示在当前所有的活动事务被提交完毕之后关闭数据库。
☑ IMMEDIATE：表示在尽可能短的时间内立即关闭数据库。
☑ ABORT：表示以终止方式来关闭数据库。

下面将分别讲解在 SHUTDOWN 语法中列举出的 4 种关闭数据库实例的方式。

1. NORMAL 方式

NORMAL 方式称作正常关闭方式，如果对关闭数据库的时间没有限制，通常会使用这种方式来关闭数据库。

【例 1.5】使用 NORMAL 方式关闭数据库，代码及运行结果如下。

```
SQL> shutdown normal
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
```

从上述代码中可以看出，Oracle 在执行 SHUTDOWN 命令后，所返回的响应信息就是关闭数据库实例的过程。当以正常方式关闭数据库时，Oracle 将执行如下操作。

☑ 阻止任何用户建立新的连接。
☑ 等待当前所有正在连接的用户主动断开连接。
☑ 当所有的用户都断开连接后，将立即关闭数据库。

2. TRANSACTIONAL 方式

TRANSACTIONAL 方式称作事务关闭方式，它的首要任务是能够保证当前所有的活动事务都可以被提交，并在尽可能短的时间内关闭数据库。

【例 1.6】使用 TRANSACTIONAL 方式关闭数据库，代码及运行结果如下。

```
SQL> shutdown transactional
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
```

以事务方式关闭数据库时，Oracle 将执行如下操作。

☑ 阻止用户建立新连接和开始新事务。

☑　等待所有活动事务提交后，再断开用户连接。
☑　当所有的活动事务提交完毕、所有的用户都断开连接后，将关闭数据库。

3. IMMEDIATE 方式

与 IMMEDIATE 单词的含义一样，IMMEDIATE 方式称作立即关闭方式，这种方式将在尽可能短的时间内关闭数据库。

【例 1.7】使用 IMMEDIATE 方式关闭数据库，代码及运行结果如下。

```
SQL> shutdown immediate
数据库已经关闭。
已经卸载数据库。
ORACLE 例程已经关闭。
```

在这种关闭方式下，Oracle 不但会立即中断当前用户的连接，而且会强行终止用户的当前活动事务，将未完成的事务回退。以立即关闭方式关闭数据库时，Oracle 将执行如下操作。

☑　阻止用户建立新连接和开始新事务。
☑　将未提交的活动事务回退。
☑　关闭数据库。

4. ABORT 方式

ABORT 方式称作终止关闭方式，具有一定的强制性和破坏性。使用这种方式会强制中断数据库的操作，这样可能会丢失一部分数据信息，影响数据库的完整性。除了由于使用其他 3 种方式无法关闭数据库而使用它之外，应该尽量避免使用这种方式。

【例 1.8】使用 ABORT 方式关闭数据库，代码及运行结果如下。

```
SQL> shutdown abort
ORACLE 例程已经关闭。
```

以立即关闭方式关闭数据库时，Oracle 将执行如下操作。

☑　阻止用户建立新连接和开始新事务。
☑　取消未提交的活动事务，而不是回退。
☑　立即终止正在执行的任何 SQL 语句。
☑　立即关闭数据库。

互动练习：在 scott 模式下，使用 DISTINCT 关键字显示 emp 表中的不重复记录。

1.6　实践与练习

1. 尝试通过数据库配置助手（Database Configuration Assistant）创建一个名称为 mr 的数据库。
2. 尝试启动数据库到 OPEN 模式，然后使用 TRANSACTIONAL 方式关闭数据库。