第 3 章

SQL 基础

学习目的与要求

本章主要介绍 PL/SQL 中的基础储备知识，数据定义语言（DDL）、数据操纵语言（DML）和数据查询语言（DQL）。通过本章的学习，读者应了解基本表的创建、修改和删除的方法；熟悉表中数据的插入、修改和删除的方法；重点掌握对基本表中数据的各种查询方法，为 PL/SQL 编程奠定良好的基础。

本章主要内容

- SQL 概述
- 数据定义
- 数据操纵
- 数据查询

3.1 SQL 语言

SQL 是一种介于关系代数与关系演算之间的结构化查询语言（Structured Query Language），是一个通用的、功能极强的关系数据库语言。SQL 不仅具有丰富的查询功能，还具有数据定义和数据控制功能，是集 DQL（数据查询语言）、DDL（数据定义语言）、DML（数据操纵语言）、DCL（数据控制语言）于一体的关系数据语言。当前几乎所有的关系数据库管理系统都支持 SQL，许多软件厂商还对 SQL 基本命令集进行了不同程度的修改和扩充。

SQL 最早是 1974 年由 Boyce 和 Chamberlin 提出，并作为 IBM 公司研制的关系数据库管理系统原型 System R 的一部分付诸实施。它功能丰富，不仅具有数据定义、数据操纵、数据控制功能，还有着强大的查询功能，而且语言简洁，容易学习，易于使用。现在 SQL 已经成为关系数据库的标准语言，并且发展了 4 个主要标准，即 ANSI（美国国家标准机构）SQL；对 ANSI SQL 修改后在 1992 年采纳的标准，称为 SQL-92 或 SQL2；后来又出了

SQL-99 也称 SQL3 标准。SQL-99 从 SQL2 扩充而来，并增加了对象关系特征和许多其他的新功能。2003 年，推出了 SQL-2003。自 1986 年公布以来，SQL 随着数据库技术的发展而不断更新、丰富。

自 SQL 成为国际标准语言以后，各个数据库厂家纷纷推出各自的 SQL 软件或与 SQL 的接口软件。这就使大多数数据库均用 SQL 作为共同的数据存取语言和标准接口，使不同数据库系统之间的相互操作有了共同的基础，这个意义是十分重大的。

SQL 语言的应用更加广泛，Oracle、Sybase、Informix、Ingres、DB2、SQL Server、Rdb 等大型数据库管理系统都实现了 SQL 语言；Dbase、Foxpro、Access 等 PC 机数据库管理系统部分实现了 SQL 语言；可以在 HTML(Hypertext Markup Language，超文本标记语言)中嵌入 SQL 语句，通过 WWW 访问数据库；在 VC、VB、Delphi、PB 中也可嵌入 SQL 语句。目前，很多数据库产品都对 SQL 语句进行再开发与扩展，如 Oracle 提供的 PL/SQL(Procedure Language and SQL)就是对 SQL 的一种扩展。

3.1.1 SQL 的分类

SQL 语言的核心内容包括如下数据语言：

(1) 数据定义语言(Data Definition Language，DDL)，用于定义数据库的逻辑结构，包括基本表、视图及索引的定义。

(2) 数据操纵语言(Data Manipulation Language，DML)，用于对关系模式中的具体数据进行增、删、改等操作。

(3) 数据查询语言(Data Query Language，DQL)，用于实现各种不同的数据查询。

(4) 数据控制语言(Data Control Language，DCL)，用于数据访问权限的控制。

3.1.2 SQL 的特点

SQL 语言是一个综合的、通用的、功能极强的、易学易用的语言，所以能够被用户和业界广泛接受，并成为国际标准。其主要特点如下。

1. 综合统一

SQL 语言集数据定义语言(DDL)、数据操纵语言(DML)、数据查询语言(DQL)、数据控制语言(DCL)的功能于一体，语言风格统一，可以独立完成数据库生命周期中的全部活动，包括定义关系模式、插入数据、建立数据库、查询、更新、维护、数据库重构、数据库安全性控制等一系列操作要求，这些为数据库应用系统的开发提供了良好的环境。

2. 高度非过程化

SQL 语言是非过程化的语言，用户只需提出“做什么”，而不必指明“怎么做”，也不需要了解存取路径的选择，SQL 语言就可以将要求交给系统，自动完成全部工作。这不但大大减轻了用户负担，而且有利于提高数据独立性。

3. 面向集合的操作方式

非关系数据模型采用的是面向记录的操作方式，操作对象是一条记录。而 SQL 语言采用集合操作方式，不仅操作对象、查找结果可以是元组的集合，而且一次插入、删除、更新操

作的对象也可以是元组的集合。

4. 以同一种语法结构提供两种使用方式

SQL 语言既是独立的语言，又是嵌入式语言。作为独立的语言，它能够独立地被用于联机交互的使用方式中，用户可以在终端键盘上直接输入 SQL 命令对数据库进行操作。作为嵌入式语言，SQL 语句能够嵌入到高级语言（例如 C、COBOL、FORTRAN、C++、Java 等）程序中，供程序员设计程序时使用。现在很多数据库应用开发工具，都将 SQL 语言直接融入到自身的语言中，使用起来更加方便。尽管 SQL 的使用方式不同，但 SQL 语言的语法基本上是一致的。这种统一语法结构又提供两种不同使用方式的方法，为用户提供了极大的灵活性与方便性。

5. 语言简洁，易学易用

SQL 语言功能极强，但其语言十分简洁，完成数据定义、数据操纵、数据控制的核心功能只用了 9 个动词：CREATE、DROP、ALTER、SELECT、INSERT、UPDATE、DELETE、GRANT、REVOKE。而且 SQL 语言语法简单，接近英语口语，因此易学易用。

3.2 数据定义语言

通过 SQL 语言的数据定义功能，可以完成基本表、视图、索引的创建、修改和删除。但 SQL 不提倡修改视图和索引的定义，如果想修改视图和索引的定义，只能先将它们删除，然后再重建。SQL 常用的数据定义语句如表 3.1 所示。

表 3.1 SQL 的数据定义语句

操作对象	操作方式		
	创　建	删　除	修　改
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

3.2.1 基本表的定义

视频讲解

表是数据库中最基本的操作对象，是实际存放数据的地方。其他的数据库对象的创建及各种操作都是围绕表进行的，可以将表看作含列和行的表单。

1. 创建基本表的语法格式

SQL 语言使用 CREATE TABLE 语句定义基本表。其一般格式为：

```
CREATE TABLE <基本表名>
( <列名> <数据类型> [列级完整性约束]
[,<列名> <数据类型> [列级完整性约束] ]
…
[,表级完整性约束]);
```

说明：

(1) 其中，“< >”中的内容是必选项，“[]”中的内容是可选项。本书以下各章节也遵循这个规定。

(2) <基本表名>规定了所定义的基本表的名字，在一个用户中不允许有两个基本表的名字相同。<列名>规定了该列(属性)的名称。一个表中不能有两列的名字相同。

(3) 表名或列名命名规则：第一个字符必须是字母，后面可以跟字母、数字、3 个特殊符号(_、$、#)；表名或列名中不可以包含空格；表名和列名不区分大小写，但显示出来都是大写；保留字不能用作表名或列名。

(4) <数据类型>规定了该列的数据类型。

(5) <列级完整性约束>是指对某一列设置的约束条件。

(6) <表级完整性约束>规定了关系主键、外键和用户自定义完整性约束。

2. 数据类型

由于基本表的每个属性列都有自己的数据类型，所以首先介绍一下 SQL 所支持的数据类型。各个厂家的 SQL 所支持的数据类型不完全一致，这里只介绍 SQL-99 规定的主要数据类型。

1) 数值型

- INTEGER 定义数据类型为整数类型，它的精度(总有效位)由执行机构确定。INTEGER 可简写成 INT。
- SMALLINT 定义数据类型为短整数类型，它的精度由执行机构确定。
- NUMERIC(p,s)定义数据类型为数值型，并给定精度 p(总的有效位，不包含符号位及小数点)或标度 s(十进制小数点右边的位数)。
- FLOAT(p)定义数据类型为浮点数值型，p 为指定的精度。
- REAL 定义数据类型为浮点数值型，它的精度由执行机构确定。
- DOUBLE PRECISION 定义数据类型为双精度浮点类型，它的精度由执行机构确定。

2) 字符类型

- CHAR(n)定义指定长度的字符串，n 为字符数的固定长度。
- VARCHAR(n)定义可变长度的字符串，其最大长度为 n，n 不可省略。

3) 位串型

- BIT(n)定义数据类型为二进制位串，其长度为 n。
- BIT VARYING(n)定义可变长度的二进制位串，其最大长度为 n，n 不可省略。

4) 时间型

- DATE 用于定义日期，包含年、月、日，格式为 YYYY-MM-DD。
- TIME 用于定义时间，包含时、分、秒，其格式为 HH:MM:SS。

5) 布尔型

- BOOLEAN 定义布尔类型，其值可以是 TRUE(真)、FALSE(假)。

对于数值型数据，可以执行算术运算和比较运算，但对其他类型数据，只可以执行比较运算，不能执行算术运算。我们在这里只介绍了常用的一些数据类型，许多 SQL 产品还扩充了其他一些数据类型，用户在实际使用中应查阅数据库系统的参考手册。

3. 约束条件

在 SQL 语言中，约束是一些规则，约束在数据库中不占存储空间。根据约束所完成的功能不同，表达完整性约束的规则有主键约束、外键约束、属性约束几类。

1）主键约束(PRIMARY KEY)

主键约束体现了实体完整性。要求某一列的值既不能为空，也不能重复。

2）外键约束(FOREIGN KEY)

外键约束体现了参照完整性。外键的取值或者为空或者参考父表的主键。

3）属性约束

属性约束体现了用户定义的完整性。属性约束主要限制某一属性的取值范围。属性约束可分为以下几类：

- 非空约束(NOT NULL)。要求某一属性的值不允许为空值。
- 唯一约束(UNIQUE)。要求某一属性的值不允许重复。
- 检查约束(CHECK)。检查约束可以对某一个属性列的值加以限制。限制就是给某一列设定条件，只有满足条件的值才允许插入。

基本表的完整性约束可定义为两级：表级约束和列级约束。表级约束可以约束表中的任意一列或多列，而列级约束只能约束其所在的某一列。

上述 5 种约束条件均可作为列级完整性约束条件，但非空约束不可以作为表级完整性约束条件，其他 4 种可以作为表级完整性约束条件。

下面通过具体实例介绍基本表的创建方法。

例 3-1 建立样本数据库中的 STUDENT 表，要求所有约束条件均为列级完整性约束，且该表满足表 3.2 中所示的条件。

表 3.2 STUDENT(学生)表

字 段 名	字段类型	是否为空	说 明	字段描述
SNO	CHAR(8)	NOT NULL	主键	学生学号
SNAME	VARCHAR2(20)	UNIQUE	唯一约束	学生姓名
SEX	CHAR(4)	NOT NULL	非空约束	性别
AGE	INT		年龄大于 16 岁	年龄
DEPT	VARCHAR2(15)			学生所在的系别名称

SQL 语句如下所示：

```
CREATE TABLE STUDENT
(SNO CHAR(8) PRIMARY KEY,                    /* 主键约束 */
 SNAME VARCHAR2(20) UNIQUE ,                 /* 唯一约束 */
 SEX CHAR(4) NOT NULL,                       /* 非空约束 */
 AGE INT CHECK(Age > 16),                    /* 检查约束 */
 DEPT VARCHAR2(15));
```

例 3-2 建立样本数据库中的 COURSE 表，要求所有约束条件均为列级完整性约束，且该表满足表 3.3 中所示的条件。

表 3.3 COURSE(课程)表

字 段 名	字 段 类 型	是 否 为 空	说 明	字 段 描 述
CNO	CHAR(8)	NOT NULL	主键	课程编号
CNAME	VARCHAR2(10)			课程名称
TNAME	VARCHAR2(10)			授课教师名
CPNO	CHAR(8)		外键(参照课程表中的课程编号)	先修课程号
CREDIT	NUMBER			学分

SQL 语句如下所示：

```
CREATE TABLE COURSE
(CNO CHAR(8) PRIMARY KEY,                    /* 主键约束 */
 CNAME VARCHAR2(10),
 TNAME VARCHAR2(10),
 CPNO CHAR(8) REFERENCES COURSE(Cno),        /* 外键约束 */
 CREDIT NUMBER);
```

例 3-3 建立样本数据库中的 SC 表，要求所有约束条件均为列级完整性约束，且该表满足表 3.4 中所示的条件。

表 3.4 SC(选课)表

字 段 名	字 段 类 型	是 否 为 空	说 明	字 段 描 述
SNO	CHAR(8)	NOT NULL	外键(参照学生表中的学生编号)	学生学号
CNO	CHAR(8)	NOT NULL	外键(参照课程表中的课程编号)	课程编号
GRADE	NUMBER			选修成绩

其中，(SNO,CNO)属性组合为主键。

SQL 语句如下所示：

```
CREATE TABLE SC
(SNO CHAR(8),
 CNO CHAR(8),
 GRADE NUMBER,
 PRIMARY KEY(SNO,CNO),                       /* 主键约束 */
 FOREIGN KEY(SNO) REFERENCES STUDENT(SNO),   /* 外键约束 */
 FOREIGN KEY (CNO) REFERENCES COURSE(CNO) ); /* 外键约束 */
```

3.2.2 基本表的修改

视频讲解

随着应用环境和实际需求的变化，经常需要修改基本表的结构，包括修改属性列的数据类型及其精度，增加新的属性列或删除属性列，增加新的约束条件或删除原有的约束条件。SQL 语言通过 ALTER TABLE 命令对基本表的结构进行修改。其一般格式为：

```
ALTER TABLE <基本表名>
   [ADD <新列名> <数据类型> [列级完整性约束]]
   [DROP COLUMN <列名>]
   [MODIFY <列名> <新的数据类型>]
   [ADD CONSTRAINT <完整性约束>]
   [DROP CONSTRAINT <完整性约束>];
```

说明：

(1) ADD：为一个基本表增加新的属性列，但新的属性列的值必须允许为空(除非有默认值)。

(2) DROP COLUMN：删除基本表中原有的一列。

(3) MODIFY：修改基本表中原有属性列的数据类型。

(4) ADD CONSTRAINT 和 DROP CONSTRAINT：分别表示添加完整性约束和删除完整性约束。

例 3-4 向 STUDENT 表中增加一个身高 Height 属性列，数据类型为 INT。

SQL 语句如下所示：

```
ALTER TABLE STUDENT ADD Height INT;
```

新增加的属性列总是表的最后一列。不论表中是否已经有数据，新增加的列值为空。所以新增加的属性列不能有 NOT NULL 约束，否则就会产生矛盾。

例 3-5 将 STUDENT 表中的 Height 属性列的数据类型改为 REAL。

SQL 语句如下所示：

```
ALTER TABLE STUDENT MODIFY Height REAL;
```

修改原有的列定义有可能会破坏已有数据，所以在修改时需要注意：可以增加列值的宽度及小数点的长度，只有当某列所有行的值为空或整张表是空时，才能减少其列值宽度，或改变其列值的数据类型。

例 3-6 给 STUDENT 表中 Height 属性列增加一个 CHECK 约束，要求学生的身高要超过 140cm 才行。

SQL 语句如下所示：

```
ALTER TABLE STUDENT ADD CONSTRAINT Chk1 CHECK(Height > 140);
```

Chk1 是 Height 属性列上新增加的 CHECK 约束的名字。

例 3-7 删除 Height 属性列上的 CHECK 约束。

SQL 语句如下所示：

```
ALTER TABLE STUDENT DROP CONSTRAINT Chk1;
```

例 3-8 删除 STUDENT 表中新增加的 Height 属性列。

SQL 语句如下所示：

```
ALTER TABLE STUDENT DROP COLUMN Height;
```

3.2.3 基本表的删除

当数据库某个基本表不再使用时，可以使用 DROP TABLE 语句删除它。其一般格式为：

```
DROP TABLE <表名> [CASCADE CONSTRAINTS];
```

删除基本表时要注意以下几点：

(1) 表一旦被删除，则无法恢复。

(2) 如果表中有数据，则表的结构连同数据一起删除。

(3) 在表上的索引、约束条件、触发器以及表上的权限也一起被删除。

(4) 当删除表时，涉及该表的视图、存储过程、函数、包被设置为无效。

(5) 只有表的创建者或者拥有 DROP ANY TABLE 权限的用户才能删除表。

(6) 如果两张表之间有主外键约束条件，则必须先删除子表，然后再删除主表。

(7) 如果加上 CASCADE CONSTRAINTS，在删除基本表的同时，相关的依赖对象也一起被删除。

例 3-9 删除学生选课表 SC。

SQL 语句如下所示：

```
DROP TABLE SC;
```

基本表定义一旦被删除，表中的数据、表上建立的索引和视图都将自动被删除掉。因此执行删除基本表的操作一定要格外小心。删除表时要先删除从表，再删除主表。

3.2.4 实践环节：基本表的操作

某员工-部门数据库中包含雇员信息和部门信息两张基本表。各表的结构如附录 A 的表 A.4 和表 A.5 所示。

雇员信息表：Emp(Empno,Ename,Age,Sal,Deptno)，表中属性列依次是雇员编号、雇员姓名、年龄、月薪和部门号。

部门信息表：Dept(Deptno,Dname,Loc)，表中属性列依次是部门号、部门名称和部门地点。

(1) 请根据表的结构，用 SQL 语句分别创建部门信息表和雇员信息表。

(2) 向雇员表中增加性别属性列，列名为 Sex，数据类型为 CHAR(4)。

(3) 更改部门表中部门地点 Loc 的数据类型为 VARCHAR2(20)。

3.3 数据操纵语言

学生选课系统中的数据表定义完成后，用户需要用 DML 语句向表中插入数据。如果数据输入有误，则需要修改数据。如果不再需要某些数据，则需要删除数据。数据的插入、修改和删除都属于 SQL 语言的数据操纵功能。

3.3.1 插入数据


视频讲解

当基本表建立以后，就可以使用 INSERT 语句向表中插入数据了。向基本表中插入数据的语法格式如下：

```
INSERT INTO <基本表名> [(<列名 1>,<列名 2>,…,<列名 n>)]
VALUES(<列值 1>,<列值 2>,…,<列值 n>)
```

其中，<基本表名>指定要插入元组的表的名字；<列名 1>,<列名 2>,…,<列名 n>为要添加列值的列名序列；VALUES 后则一一对应要添加列的输入值。

注意：

(1) 向表中插入数据之前，表的结构必须已经创建。

(2) 插入的数据及列名之间用逗号分开。

(3) 在 INSERT 语句中列名是可以选择指定的，如果没有指定列名，则表示这些列按表中或视图中列的顺序和个数。

(4) 插入值的数据类型、个数、前后顺序必须与表中属性列的数据类型、个数、前后顺序匹配。

例 3-10 向学生表中插入一个新的学生记录。

方法一：省略所有列名

```
INSERT INTO STUDENT
VALUES ('05880111', '张晓三', '男', 23, '数学系');
```

方法二：指出所有列名

```
INSERT INTO STUDENT(SNO,SNAME,SEX,AGE,DEPT)
VALUES ('05880111', '张晓三', '男', 23, '数学系');
```

两种方法的作用是相同的。

例 3-11 向学生表中指定的属性列插入数据。

```
INSERT INTO STUDENT (SNO,SNAME,SEX)
VALUES ('05880112', '王晓五', '女');
```

其中，没有插入数据的属性列的值均为空值。

注意：在向表中插入数据时，所插入的数据应满足定义表时的约束条件。例如，如果再次向 STUDENT 表中插入学号为“05880112”的学生记录时，系统就会给出错误提示信息，违反了主键约束。如果再插入另一个新的学生“05880113”的学生记录，但不知道此同学的性别，插入的性别属性列的值为空值，此时系统也会给出错误提示信息，违反了定义表时对于“性别”字段的非空约束。

3.3.2 修改数据


视频讲解

如果表中的数据出现错误，可以利用 UPDATE 命令进行修改。UPDATE 语句用以修改满足指定条件的元组信息。UPDATE 语句一般语法格式为：

```
UPDATE <基本表名>
SET <列名 1> = <表达式> [,<列名 2> = <表达式>]…
[WHERE <条件>];
```

其中,UPDATE 关键字用于定位修改哪一张表,SET 关键字用于定位修改这张表中的哪些属性列,WHERE <条件>用于定位修改这些属性列中的哪些行。UPDATE 语句只能修改一个基本表中满足 WHERE <条件>的元组的某些列值,即其后只能有一个基本表名。这里,WHERE <条件>是可选的,如果省略不选,则表示要修改表中所有的元组。

1. 修改某一个元组的值

例 3-12 将 java 课程的学分改为 4 学分。

```
UPDATE COURSE
SET CREDIT = 4
WHERE CNAME = 'java';
```

2. 修改多个元组的值

例 3-13 将所有男同学的年龄增加 2 岁。

```
UPDATE STUDENT
SET AGE = AGE + 2
WHERE SEX = '男';
```

例 3-14 将所有课程的学分减 1。

```
UPDATE COURSE
SET CREDIT = CREDIT - 1;
```

3.3.3 删除数据

如果不再需要学生选课系统中的某些数据,此时应该删除这些数据,以释放其所占用的存储空间。DELETE 语句的一般语法格式为:

```
DELETE FROM <表名> [WHERE <条件>];
```

DELETE 语句的功能是从指定表中删除满足 WHERE <条件>的所有元组。DELETE 语句只删除表中的数据,而不能删除表的结构,所以表的定义仍然在数据字典中。如果省略 WHERE <条件>,表示删除表中全部的元组信息。

1. 删除某一个元组的值

例 3-15 删除学号为“20180010”的学生记录。

```
DELETE FROM STUDENT
WHERE SNO = '20180010';
```

2. 删除多个元组的值

例 3-16 删除学号为“20180002”的学生的选课记录。

```
DELETE FROM SC
WHERE SNO = '20180002';
```

每一个学生可能选修多门课程，所以 DELETE 语句会删除这个学生的多条选课记录。

例 3-17　删除所有学生的选课记录。

```
DELETE FROM SC;
```

3.3.4　实践环节：数据的操纵

根据 3.2.4 节中所创建的两张表——部门信息表 Dept(Deptno,Dname,Loc)和雇员信息表 Emp(Empno,Ename,Age,Sal,Deptno)，用 SQL 语句完成下列操作。

(1) 分别向雇员信息表和部门信息表中插入数据，各表中的数据如附录 A 的图 A.4 和图 A.5 所示。

(2) 给部门编号为 20 的所有员工月薪涨 500 元。

(3) 删除雇员表中年龄小于 18 岁的员工信息。

3.4　数据查询语言

3.4.1　SELECT 语句的一般格式

PL/SQL 语言中最重要、最核心的操作就是数据查询。关系代数的运算在关系数据库中主要由 SQL 数据查询来体现。SQL 语言提供 SELECT 语句进行数据库的查询，该语句具有灵活的使用方式和丰富的功能。其基本格式为：

```
SELECT [ALL|DISTINCT] <目标列表达式>[,<目标列表达式>]…
FROM <表名或视图名>[,<表名或视图名>]…
[WHERE <条件表达式>]
[GROUP BY <列名 1> [HAVING <组条件表达式>]]
[ORDER BY <列名 2> [ASC|DESC]];
```

其中：

(1) SELECT 子句说明要查询的数据。ALL 表示筛选出数据库表中满足条件的所有记录，一般情况下省略不写。DISTINCT 表示查询结果中无重复记录。

(2) FROM 子句说明要查询的数据来源。可以是数据库中的一个或多个表，或者是视图，各项之间用逗号分隔。

(3) WHERE 子句指定查询条件。查询条件中会涉及 PL/SQL 函数和 PL/SQL 操作符。

(4) GROUP BY 子句表示在查询时，可以按照某个或某些字段分组汇总，各分组选项之间用逗号分隔。HAVING 子句必须跟随 GROUP BY 一起使用，表示在分组汇总时，可以根据组条件表达式筛选出满足条件的组记录。

(5) ORDER BY 子句表示在显示结果时，按照指定字段进行排序。ASC 表示升序，DESC 表示降序，默认情况下是 ASC。

整个 SELECT 语句的含义是：根据 WHERE 子句的条件表达式，从 FROM 子句指定的表或视图中找出满足条件的元组，再按照 SELECT 子句中的目标列表达式，选出元组中的属性值形成结果表。如果有 GROUP BY 子句，则将结果按<列名 1 >的值进行分组，该属性列值相等的元组为一个组。通常会在每组中使用聚组函数。如果 GROUP BY 子句带有 HAVING 子句，则只有满足指定条件的组才能够输出。如果有 ORDER BY 子句，则结果表还需要按<列名 2 >的值的升序或者降序排列。查询子句的顺序是不可以前后调换的。

由于 SELECT 语句的形式多样，可以完成单表查询、多表连接查询、嵌套查询和集合查询等，想要熟练地掌握和运用 SELECT 语句，必须要下一番工夫。

下面我们以学生选课样例数据库系统为例，说明 SELECT 语句的各种用法。

3.4.2 单表查询

视频讲解

单表查询是指查询的数据只来自一张表，此时，SELECT 语句中的 FORM 子句只涉及一张表的查询。

1. 选择表中若干列

选择表中的全部列或部分列，这就是投影运算。

1）查询指定的列

例 3-18 查询全体学生的学号、姓名和年龄。

```
SELECT SNO,SNAME,AGE
FROM STUDENT;
```

查询结果如图 3.1 所示。

例 3-19 查询全部课程的课程名称和授课教师名。

```
SELECT CNAME,TNAME
FROM COURSE;
```

查询结果如图 3.2 所示。

SNO	SNAME	AGE
20180001	周一	17
20180002	吴二	20
20180003	张三	19
20180004	李四	22
20180005	王五	22
20180006	赵六	19
20180007	陈七	23
20180008	刘八	21
20180009	郑九	18
20180010	孙十	21

图 3.1 例 3-18 的 PL/SQL 程序运行效果

CNAME	TNAME
maths	李老师
english	赵老师
japanese	陈老师
database	张老师
java	王老师
jsp_design	刘老师

图 3.2 例 3-19 的 PL/SQL 程序运行效果

2）查询全部列

例 3-20 查询全部课程的详细记录。

```
SELECT *
FROM COURSE;
```

查询结果如图 3.3 所示。

3）查询经过计算的值

例 3-21 查询全体学生的姓名、性别及其出生年份。

```
SELECT SNAME,SEX,2018 - AGE
FROM STUDENT;
```

查询结果如图 3.4 所示。

CNO	CNAME	TNAME	CPNO	CREDIT
c1	maths	李老师		3
c2	english	赵老师		5
c3	japanese	陈老师		4
c4	database	张老师	c1	4
c5	java	王老师	c1	3
c6	jsp_design	刘老师	c5	2

图 3.3 例 3-20 的 PL/SQL 程序运行效果

SNAME	SEX	2018-AGE
周一	男	2001
吴二	女	1998
张三	女	1999
李四	男	1996
王五	男	1996
赵六	男	1999
陈七	女	1995
刘八	男	1997
郑九	女	2000
孙十	女	1997

图 3.4 例 3-21 的 PL/SQL 程序运行效果

4）指定别名来改变查询结果的列标题

从前面的查询结果中，我们可以看到，显示的每一个属性列的标题是列名，有时候列名就是拼音代码，意义不是很清楚，为了解决这个问题，我们可以给属性列提供一个别名。方法就是：在列名的后面加上一个空格或者“as”，然后写上它的别名。在查询结果显示时就用别名代替列名了。

例 3-22 查询全体学生的姓名、性别及其出生年份。

```
SELECT SNAME,SEX,2018 - AGE 出生年份
FROM STUDENT;
```

查询结果如图 3.5 所示。

2. 选择表中若干行

选择表中若干行，这就是选择运算。

1）消除取值重复的行

例 3-23 查询学生表中的所有院系。

```
SELECT DEPT
FROM STUDENT;
```

查询结果如图 3.6 所示。

SNAME	SEX	出生年份
周一	男	2001
吴二	女	1998
张三	女	1999
李四	男	1996
王五	男	1996
赵六	男	1999
陈七	女	1995
刘八	男	1997
郑九	女	2000
孙十	女	1997

图 3.5 例 3-22 的 PL/SQL 程序运行效果

DEPT
计算机系
信息系
计算机系
信息系
数学系
数学系
日语系
日语系
管理系
管理系

图 3.6 例 3-23 的 PL/SQL 程序运行效果

由于多名同学属于同一个院系，所以查询的结果中包含了许多重复的行。如果想去掉重复的行，必须指定 DISTINCT 关键字。

```
SELECT DISTINCT dept
FROM STUDENT;
```

图 3.7 例 3-23 的 PL/SQL 程序运行效果

查询结果如图 3.7 所示。

2）查询满足条件的元组

查询满足指定条件的元组可以通过 WHERE 子句来实现。使用 WHERE 子句时，应该注意以下几点：

（1）如果该列数据类型为字符型，需要使用单引号把字符串括起来。例如 WHERE Cname='java'。单引号内的字符串大小写是有区别的。

（2）如果该列数据类型为日期型，需要使用单引号把日期括起来。

（3）如果该列数据类型为数字型，则不必用单引号。例如 WHERE Age>20。

（4）WHERE 子句中可以使用列名或表达式，但不能使用它的别名。

WHERE 子句常用的查询条件如表 3.5 所示。

表 3.5 常用的查询条件

查 询 条 件	谓 词
比较	=，>，<，>=，<=，!=，< >，!>，!<，NOT 等比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件	AND，OR

① 比较大小。

例 3-24 查询数学系全体学生的姓名。

```
SELECT SNAME
FROM STUDENT
WHERE DEPT = '数学系';
```

图 3.8 例 3-24 的 PL/SQL 程序运行效果

查询结果如图 3.8 所示。

例 3-25 查询年龄超过 20 岁的学生姓名及其年龄。

```
SELECT SNAME,AGE
FROM STUDENT
WHERE AGE>20;
```

查询结果如图 3.9 所示。

例 3-26 查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT SNO
FROM SC
WHERE GRADE<60;
```

查询结果如图 3.10 所示。

SNAME	AGE
李四	22
王五	22
陈七	23
刘八	21
孙十	21

图 3.9　例 3-25 的 PL/SQL 程序运行效果

SNO
20180002
20180009
20180007
20180005

图 3.10　例 3-26 的 PL/SQL 程序运行效果

语句中使用了 DISTINCT 关键字,目的是当某一个学生有多门课程不及格时,他的学号只显示一次。

② 确定范围(谓词 BETWEEN AND)。

例 3-27　查询年龄在 16～20 岁(包括 16 岁和 20 岁)的学生姓名和年龄。

```
SELECT SNAME,AGE
FROM STUDENT
WHERE AGE BETWEEN 16 AND 20;
```

查询结果如图 3.11 所示。

例 3-28　查询年龄不在 16～20 岁的学生姓名和年龄。

```
SELECT SNAME,AGE
FROM STUDENT
WHERE AGE NOT BETWEEN 16 AND 20;
```

查询结果如图 3.12 所示。

SNAME	AGE
周一	17
吴二	20
张三	19
赵六	19
郑九	18

图 3.11　例 3-27 的 PL/SQL 程序运行效果

SNAME	AGE
李四	22
王五	22
陈七	23
刘八	21
孙十	21

图 3.12　例 3-28 的 PL/SQL 程序运行效果

③ 确定集合(谓词 IN)。

例 3-29　查询计算机系、日语系和管理系的学生姓名和性别。

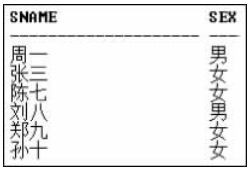
```
SELECT SNAME,SEX
FROM STUDENT
WHERE DEPT IN ('计算机系','日语系','管理系');
```

查询结果如图 3.13 所示。

例 3-30　查询既不是计算机系、日语系,也不是管理系的学生姓名和性别。

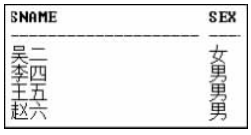
```
SELECT SNAME,SEX
FROM STUDENT
WHERE DEPT NOT IN ('计算机系','日语系','管理系');
```

查询结果如图 3.14 所示。



SNAME	SEX
周一	男
张三	女
陈七	女
刘八	男
郑九	女
孙十	女

图 3.13 例 3-29 的 PL/SQL 程序运行效果



SNAME	SEX
吴二	女
李四	男
王五	男
赵六	男

图 3.14 例 3-30 的 PL/SQL 程序运行效果

④ 字符匹配(谓词 LIKE)。

谓词 LIKE 可以用来进行字符串的匹配。基本格式为:

```
[NOT] LIKE '<匹配串>' [ESCAPE '<换码字符>']
```

其含义是查找指定的属性列值与<匹配串>相匹配的元组。<匹配串>可以是一个完整的字符串,也可以含有通配符%和_。其中,%(百分号)代表任意长度(长度可以为 0)的字符串;_(下横线)代表任意单个字符。

例 3-31 查询所有姓张的学生姓名、年龄和系别名称。

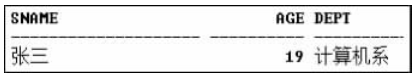
```
SELECT SNAME,AGE,DEPT
FROM STUDENT
WHERE SNAME LIKE '张%';
```

查询结果如图 3.15 所示。

例 3-32 查询姓名中第二个汉字是“七”的学生姓名和年龄。

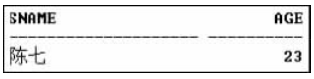
```
SELECT SNAME,AGE
FROM STUDENT
WHERE SNAME LIKE '_七%';
```

查询结果如图 3.16 所示。



SNAME	AGE	DEPT
张三	19	计算机系

图 3.15 例 3-31 的 PL/SQL 程序运行效果



SNAME	AGE
陈七	23

图 3.16 例 3-32 的 PL/SQL 程序运行效果

如果用户查询的匹配字符串本身含有%或_,这时就要使用 ESCAPE‘<换码字符>’短语对通配符进行转义。

例 3-33 查询以“jsp_”开头,且倒数第二个字符为 g 的课程的详细信息。

```
SELECT *
FROM COURSE
WHERE CNAME LIKE 'jsp\_%g_' ESCAPE'\';
```

查询结果如图 3.17 所示。

⑤ 涉及空值的查询。

例 3-34 查询选修了课程,但没有成绩的学生学号和相应的课程号。

```
SELECT SNO,CNO
```

```
FROM SC
WHERE GRADE IS NULL;
```

查询结果如图 3.18 所示。

CNO	CNAME	TNAME	CPNO	CREDIT
c6	jsp_design	刘老师	c5	2

图 3.17　例 3-33 的 PL/SQL 程序运行效果

SNO	CNO
20180007	c4
20180008	c1

图 3.18　例 3-34 的 PL/SQL 程序运行效果

注意：这里“IS”不能用等号(=)代替。

例 3-35　查询选修了课程，并且有成绩的学生学号和相应的课程号。

```
SELECT SNO,CNO
FROM SC
WHERE GRADE IS NOT NULL;
```

查询结果如图 3.19 所示。

SNO	CNO
20180001	c1
20180001	c2
20180001	c3
20180001	c4
20180002	c1
20180002	c3
20180002	c5
20180003	c1
20180003	c2
20180003	c3
20180003	c5
20180004	c2
20180004	c3
20180005	c1
20180005	c5
20180006	c6
20180007	c1
20180007	c6
20180009	c1
20180009	c2
20180009	c3
20180009	c4
20180009	c5
20180009	c6

图 3.19　例 3-35 的 PL/SQL 程序运行效果

⑥ 多重条件查询。

逻辑运算符 AND 和 OR 可用来联结多个查询条件。AND 的优先级高于 OR，但用户可以通过括号来改变优先级。

例 3-36　查询日语系女同学的姓名和年龄。

```
SELECT SNAME,AGE
FROM STUDENT
WHERE DEPT = '日语系' AND SEX = '女';
```

查询结果如图 3.20 所示。

例 3-37　查询管理系或年龄在 20 岁以下的学生姓名。

```
SELECT SNAME
FROM STUDENT
WHERE DEPT = '管理系' OR AGE < 20;
```

查询结果如图 3.21 所示。

SNAME	AGE
陈七	23

图 3.20　例 3-36 的 PL/SQL 程序运行效果

SNAME
周一
张三
赵六
郑九
孙十

图 3.21　例 3-37 的 PL/SQL 程序运行效果

⑦ 对查询结果进行排序。

ORDER BY 子句可指定按照一个或多个属性列的升序(ASC)或者降序(DESC)重新排列查询结果。省略不写，默认为升序排列。由于是控制输出结果，因此 ORDER BY 子句只能用于最终的查询结果。

例 3-38 查询选修 c3 课程的学生学号及成绩，查询结果按照成绩的降序排列。

```
SELECT SNO, GRADE
FROM SC
WHERE CNO = 'c3'
ORDER BY GRADE DESC;
```

SNO	GRADE
20180004	97
20180001	82
20180009	80
20180003	66
20180002	61

图 3.22 例 3-38 的 PL/SQL 程序运行效果

查询结果如图 3.22 所示。

例 3-39 查询所有学生的基本信息，查询结果按学生年龄的升序排列，年龄相同时则按学号降序排列。

```
SELECT *
FROM STUDENT
ORDER BY AGE ASC, SNO DESC;
```

查询结果如图 3.23 所示。

SNO	SNAME	SEX	AGE	DEPT
20180001	周一	男	17	计算机系
20180009	郑九	女	18	管理系
20180006	赵六	男	19	数学系
20180003	张三	女	19	计算机系
20180002	吴二	女	20	信息系
20180010	孙十	女	21	管理系
20180008	刘八	男	21	日语系
20180005	王五	男	22	数学系
20180004	李四	男	22	信息系
20180007	陈七	女	23	日语系

图 3.23 例 3-39 的 PL/SQL 程序运行效果

3. 使用聚组函数

为了进一步方便用户，增强检索功能，SQL 提供了许多聚组函数，主要有：

(1) COUNT ([DISTINCT|ALL] *) 统计元组个数
COUNT ([DISTINCT|ALL] <列名>) 统计某一列中值的个数
(2) SUM ([DISTINCT|ALL] <列名>) 计算一列值的总和(此列必须是数值型)
(3) AVG ([DISTINCT|ALL] <列名>) 计算一列值的平均值(此列必须是数值型)
(4) MAX ([DISTINCT|ALL] <列名>) 求一列值中的最大值
(5) MIN ([DISTINCT|ALL] <列名>) 求一列值中的最小值

如果指定 DISTINCT 短语，则表示在查询时要取消指定列中的重复值。如果不指定 DISTINCT 短语或指定 ALL 短语(ALL 为默认值)，则表示不取消重复值。在聚组函数遇到空值时，除 COUNT(*)外，都跳过空值而只处理非空值。

例 3-40 查询学生表中的总人数。

```
SELECT COUNT (*)
FROM STUDENT;
```

查询结果如图 3.24 所示。

例 3-41 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT SNO)
FROM SC;
```

查询结果如图 3.25 所示。

COUNT(*)
10

图 3.24 例 3-40 的 PL/SQL 程序运行效果

COUNT(DISTINCTSNO)
9

图 3.25 例 3-41 的 PL/SQL 程序运行效果

由于存在一个同学选修多门课程的情况，为了避免重复计算学生人数，所以必须加 DISTINCT 关键字，表示在统计人数时，取消指定列中的重复值。

例 3-42 查询选修 c3 课程的平均成绩、最高成绩和最低成绩。

```
SELECT AVG(GRADE), MAX(GRADE), MIN(GRADE)
FROM SC
WHERE CNO = 'c3';
```

查询结果如图 3.26 所示。

例 3-43 查询学号为 20180001 学生选修课程的成绩总和。

```
SELECT SUM(GRADE)
FROM SC
WHERE SNO = '20180001';
```

查询结果如图 3.27 所示。

AVG(GRADE)	MAX(GRADE)	MIN(GRADE)
77.2	97	61

图 3.26 例 3-42 的 PL/SQL 程序运行效果

SUM(GRADE)
340

图 3.27 例 3-43 的 PL/SQL 程序运行效果

3.4.3 分组查询

视频讲解

在 SELECT 语句中可以使用 GROUP BY 子句将查询结果按照某一列或多列的值分组，值相等的为一组，然后使用聚组函数返回每一个组的汇总信息。而且，还可以使用 HAVING 子句限制返回的结果集。

例 3-44 查询选课表中每门课程的课程号及这门课程的选修人数。

```
SELECT CNO, COUNT(SNO)
FROM SC
GROUP BY CNO;
```

查询结果如图 3.28 所示。

该 SELECT 语句对 SC 表按 Cno 的值进行分组，所有相同 Cno 值的元组为一组，然后

对每一组用聚组函数 COUNT 来计算，统计该组的学生人数。在分组查询中 HAVING 子句用于分完组后，对每一组进行条件判断，只有满足条件的分组才被选出来，这种条件判断一般与 GROUP BY 子句有关。

例 3-45 查询选修 3 门及其以上课程的学生学号。

```
SELECT SNO
FROM SC
GROUP BY SNO
HAVING COUNT(Cno)>=3;
```

查询结果如图 3.29 所示。

CNO	COUNT(SNO)
c1	7
c3	5
c4	3
c6	3
c5	4
c2	4

图 3.28 例 3-44 的 PL/SQL 程序运行效果

SNO
20180001
20180002
20180003
20180007
20180009

图 3.29 例 3-45 的 PL/SQL 程序运行效果

使用 GROUP BY 和 HAVING 子句时需要注意以下几点：

(1) 带有 GROUP BY 子句的查询语句中，在 SELECT 子句中指定的列要么是 GROUP BY 子句中指定的列，要么包含聚组函数，否则出错。

(2) 可以使用多个属性列进行分组。

(3) 聚组函数只能出现在 SELECT、HAVING、ORDER BY 子句中。在 WHERE 子句中是不能使用聚组函数的。

在一个 SELECT 语句中可以有 WHERE 子句和 HAVING 子句，这两个子句都可以用于限制查询的结果。那么，WHERE 子句与 HAVING 子句的区别是：

(1) WHERE 子句的作用是在分组之前过滤数据。WHERE 条件中不能包含聚组函数。使用 WHERE 条件选择满足条件的行。

(2) HAVING 子句的作用是在分组之后过滤数据。HAVING 条件中经常包含聚组函数。使用 HAVING 条件选择满足条件的组。使用 HAVING 子句时必须首先使用 GROUP BY 进行分组。

3.4.4 连接查询

视频讲解

在数据库中通常存在着多个相互关联的表，用户常常需要同时从多个表中找出自己想要的数据，这就涉及多个数据表的查询。连接查询是指通过两个或两个以上的关系表或视图的连接操作来实现的查询。连接查询是关系数据库中最主要的查询，包括等值连接、非等值连接、自然连接、自身连接、外连接和复合条件连接等。

连接查询中用来连接两个表的条件称为连接条件或连接谓词，其格式为：

[<表名 1>.]<列名 1><比较运算符>[<表名 2>.]<列名 2>

其中，比较运算符主要有＝、＞、＜、＞＝、＜＝、!＝。

此外，连接谓词还可以使用下面形式：

```
[<表名 1>.]<列名 1> BETWEEN [<表名 2>.]<列名 2> AND [<表名 2>.]<列名 3>
```

连接条件中的列名称为连接字段。连接条件中的各连接字段的数据类型必须是能够比较的，但名字不必相同。

1. 等值连接

当连接运算符为“＝”时，称为等值连接。使用其他运算符时，称为非等值连接。

例 3-46 查询每个同学基本信息及其选修课程的情况。

```
SELECT STUDENT. *, SC. *
FROM STUDENT, SC
WHERE STUDENT.SNO = SC.SNO;
```

查询结果如图 3.30 所示。

SNO	SNAME	SEX	AGE	DEPT	SNO	CNO	GRADE
20180001	周一	男	17	计算机系	20180001	c1	75
20180001	周一	男	17	计算机系	20180001	c2	95
20180001	周一	男	17	计算机系	20180001	c3	82
20180001	周一	男	17	计算机系	20180001	c4	88
20180002	吴二	女	20	信息系	20180002	c1	89
20180002	吴二	女	20	信息系	20180002	c3	61
20180002	吴二	女	20	信息系	20180002	c5	55
20180003	张三三	女	19	计算机系	20180003	c1	72
20180003	张三三	女	19	计算机系	20180003	c2	45
20180003	张三三	女	19	计算机系	20180003	c3	66
20180003	张三三	女	19	计算机系	20180003	c5	86
20180004	李四	男	22	信息系	20180004	c2	85
20180004	李四	男	22	信息系	20180004	c3	97
20180005	王五	男	22	数学系	20180005	c1	52
20180005	王五	男	22	数学系	20180005	c5	56
20180006	赵六	男	19	数学系	20180006	c6	74
20180007	陈七	女	23	日语系	20180007	c1	57
20180007	陈七	女	23	日语系	20180007	c4	
20180007	陈七	女	23	日语系	20180007	c6	80
20180008	刘八	男	21	日语系	20180008	c1	
20180009	郑九	女	18	管理系	20180009	c1	86
20180009	郑九	女	18	管理系	20180009	c2	67
20180009	郑九	女	18	管理系	20180009	c3	80
20180009	郑九	女	18	管理系	20180009	c4	72
20180009	郑九	女	18	管理系	20180009	c5	36
20180009	郑九	女	18	管理系	20180009	c6	52

图 3.30 例 3-46 的 PL/SQL 程序运行效果

说明：

(1) STUDENT.SNO ＝ SC.SNO 是两个关系表的连接条件，STUDENT 表和 SC 表中的记录只有满足这个条件才能连接。

(2) 在 STUDENT 表和 SC 表中存在相同的属性名 SNO，因此存在属性的二义性问题。SQL 通过在属性前面加上关系名及一个小圆点来解决这个问题，表示该属性来自这个关系。

2. 自然连接

如果是按照两个表中的相同属性进行等值连接，并且在结果中去掉了重复的属性列，我们称之为自然连接。

例 3-47 用自然连接来完成查询每个同学基本信息及其选修课程的情况。

```
SELECT STUDENT.SNO, SNAME,SEX, AGE, DEPT,CNO,GRADE
```

```
FROM STUDENT, SC
WHERE STUDENT.SNO = SC.SNO;
```

查询结果如图 3.31 所示。

SNO	SNAME	SEX	AGE	DEPT	CNO	GRADE
20180001	周一	男	17	计算机系	c1	75
20180001	周一	男	17	计算机系	c2	95
20180001	周一	男	17	计算机系	c3	82
20180001	周一	男	17	计算机系	c4	88
20180002	吴二	女	20	信息系	c1	89
20180002	吴二	女	20	信息系	c3	61
20180002	吴二	女	20	信息系	c5	55
20180003	张三	女	19	计算机系	c1	72
20180003	张三	女	19	计算机系	c2	45
20180003	张三	女	19	计算机系	c3	66
20180003	张三	女	19	计算机系	c5	86
20180004	李四	男	22	信息系	c2	85
20180004	李四	男	22	信息系	c3	97
20180005	王五	男	22	数学系	c1	52
20180005	王五	男	22	数学系	c5	56
20180006	赵六	男	19	数学系	c6	74
20180007	陈七	女	23	日语系	c1	57
20180007	陈七	女	23	日语系	c4	
20180007	陈七	女	23	日语系	c6	80
20180008	刘八	男	21	日语系	c1	
20180009	郑九	女	18	管理系	c1	86
20180009	郑九	女	18	管理系	c2	67
20180009	郑九	女	18	管理系	c3	80
20180009	郑九	女	18	管理系	c4	72
20180009	郑九	女	18	管理系	c5	36
20180009	郑九	女	18	管理系	c6	52

图 3.31 例 3-47 的 PL/SQL 程序运行效果

3. 复合条件连接

上面例题中，在 WHERE 子句里除了连接条件外，还可以有多个限制条件。连接条件用于多个表之间的连接，限制条件用于限制所选取的记录要满足什么条件，这种连接称为复合条件连接。

例 3-48 查询选修课程号为 c1，并且成绩不及格的学生学号、姓名和系别名称。

```
SELECT STUDENT.SNO, SNAME, DEPT
FROM STUDENT, SC
WHERE STUDENT.SNO = SC.SNO          /* 连接条件 */
and CNO = 'c1'                      /* 限制条件 */
and GRADE < 60;                     /* 限制条件 */
```

SNO	SNAME	DEPT
20180005	王五	数学系
20180007	陈七	日语系

图 3.32 例 3-48 的 PL/SQL 程序运行效果

查询结果如图 3.32 所示。

连接操作除了可以是两个表的连接外，还可以是两个以上的表的连接，把它称为多表连接。

例 3-49 查询计算机系选修 maths 课程的学生姓名、授课教师名以及这门课程的成绩。

```
SELECT SNAME, TNAME, GRADE
FROM STUDENT, COURSE, SC
WHERE STUDENT.SNO = SC.SNO          /* 连接条件 */
AND COURSE.CNO = SC.CNO             /* 连接条件 */
and DEPT = '计算机系'               /* 限制条件 */
and CNAME = 'maths';                /* 限制条件 */
```

查询结果如图 3.33 所示。

如果是多个表之间连接，那么 WHERE 子句中就有多个连接条件。n 个表之间的连接至少有 n－1 个连接条件。

4. 自身连接

连接操作不仅可以在两个表之间进行，也可以是一个表与其自身进行连接，称为表的自身连接。自身连接要求必须给表取别名，把它们当作两个不同的表来处理。

例 3-50 在 SC 表中查询至少选修课程号为 c1 和 c2 的学生学号。

在 SC 表中，每一条记录只是显示一个学生选修一门课程的情况，在这里，一条记录不能同时显示选修两门课程的情况，因此就要将 SC 表与其自身连接。为 SC 表取两个别名，一个是 FIRST，另一个是 SECOND，完成查询的语句为：

```
SELECT FIRST.SNO
FROM SC FIRST,SC SECOND
WHERE FIRST.SNO = SECOND.SNO          /*连接条件*/
and FIRST.CNO = 'c1'                  /*限制条件*/
and SECOND.CNO = 'c2';                /*限制条件*/
```

查询结果如图 3.34 所示。

SNAME	TNAME	GRADE
周一	李老师	75
张三	李老师	72

图 3.33 例 3-49 的 PL/SQL 程序运行效果

SNO
20180001
20180009
20180003

图 3.34 例 3-50 的 PL/SQL 程序运行效果

上面例题中，连接条件用来实现每一条记录是同一个学生的选课信息，限制条件用来实现选修的课程至少有 c1 和 c2。

5. 外连接

在通常的连接操作中，只有满足连接条件的元组才能作为结果输出。如例 3-46 和例 3-47 的结果中没有 20180010 学生的信息，原因在于他没有选课，在 SC 表中没有相应的元组。如果想以 Student 表为主体列出每个学生的基本情况及其选课情况，若某个学生没有选课，只输出学生的基本信息，其选课信息可以为空值，此时就需要使用外连接了。

外连接的表示方法为：在连接条件的某一边加上操作符(＋)(有的数据库系统中用＊)。(＋)号放在连接条件中信息不完整的那一边。外连接运算符(＋)若出现在连接条件的右边，则称为左外连接；若出现在连接条件的左边，则称为右外连接。

例 3-51 以 Student 表为主体列出每个学生的基本情况及其选课情况，若某个学生没有选课，只输出学生的基本信息，其选课信息为空值。

```
SELECT STUDENT.SNO,SNAME,SEX,AGE,DEPT,CNO,GRADE
FROM STUDENT,SC
WHERE STUDENT.SNO = SC.SNO(+);
```

查询结果如图 3.35 所示。

SNO	SNAME	SEX	AGE	DEPT	CNO	GRADE
20180001	周一	男	17	计算机系	c1	75
20180001	周一	男	17	计算机系	c2	95
20180001	周一	男	17	计算机系	c3	82
20180001	周一	男	17	计算机系	c4	88
20180002	吴二	女	20	信息系	c1	89
20180002	吴二	女	20	信息系	c3	61
20180002	吴二	女	20	信息系	c5	55
20180003	张三三	女	19	计算机系	c1	72
20180003	张三三	女	19	计算机系	c2	45
20180003	张三三	女	19	计算机系	c3	66
20180003	张三三	女	19	计算机系	c5	86
20180004	李四	男	22	信息系	c2	85
20180004	李四	男	22	信息系	c3	97
20180005	王五	男	22	数学系	c1	52
20180005	王五	男	22	数学系	c5	56
20180006	赵六	男	19	数学系	c6	74
20180007	陈七	女	23	日语系	c1	57
20180007	陈七	女	23	日语系	c4	
20180007	陈七	女	23	日语系	c6	80
20180008	刘八	男	21	日语系	c1	
20180009	郑九	女	18	管理系	c1	86
20180009	郑九	女	18	管理系	c2	67
20180009	郑九	女	18	管理系	c3	80
20180009	郑九	女	18	管理系	c4	72
20180009	郑九	女	18	管理系	c5	36
20180009	郑九	女	18	管理系	c6	52
20180010	孙十	女	21	管理系		

图 3.35 例 3-51 的 PL/SQL 程序运行效果

3.4.5 嵌套查询

在 SQL 语言中，一个 SELECT-FROM-WHERE 语句称为一个查询块。将一个查询块嵌套在另一个查询块的 WHERE 子句或 HAVING 子句的条件中的查询称为嵌套查询。这也是涉及多表的查询，其中外层查询称为父查询，内层查询称为子查询。

子查询中还可以嵌套其他子查询，即允许多层嵌套查询，其执行过程是由内向外的，每一个子查询是在上一级查询处理之前完成的。这样上一级的查询就可以利用已完成的子查询的结果，可以将一系列简单的查询组合成复杂的查询，从而一些原来无法实现的查询也因为有了多层嵌套的子查询而迎刃而解。

使用子查询的原则如下：

(1) 子查询必须用括号括起来。

(2) 子查询不能包含 ORDER BY 子句。

(3) 子查询可以在许多 SQL 语句中使用，如 SELECT、INSERT、UPDATE、DELETE 语句中。

1. 不相关子查询

查询条件不依赖于父查询的子查询称为不相关子查询。它的执行过程为：先执行子查询，将子查询的结果作为外层父查询的条件，然后执行父查询。

不相关子查询的特点如下：

(1) 先执行子查询，后执行父查询。

(2) 子查询能够独立执行，不依赖于外层父查询。

(3) 子查询只执行一次。

1）带有 IN 谓词的子查询

当子查询的结果是一个集合时，经常使用带 IN 谓词的子查询。

例 3-52 查询选修课程号为 c2 的学生姓名。

方法一：采用前面学习的多表连接查询来完成。

```
SELECT SNAME
FROM STUDENT,SC
WHERE STUDENT.SNO = SC.SNO AND CNO = 'c2';
```

方法二：采用子查询来完成。

```
SELECT SNAME
FROM STUDENT
WHERE SNO IN
        ( SELECT SNO
          FROM SC
          WHERE CNO = 'c2');
```

查询结果如图 3.36 所示。

查询选修 c2 课程的学生学号是一个子查询，查询学生的姓名是父查询。由于可能有多个同学选修了 c2 课程，所以子查询是一个集合，采用 IN 谓词。上述查询的执行过程是：先执行子查询，得到选修 c2 课程的学生学号的集合，然后将该集合作为外层父查询的条件，执行父查询，从而得到集合中学号对应的学生姓名。

例 3-53 查询既没有选修课程号 c1，也没有选修课程号 c2 的学生学号。

```
SELECT SNO
FROM SC
WHERE SNO NOT IN
             (SELECT SNO
              FROM SC
              WHERE CNO = 'c1')
    and SNO NOT IN
             (SELECT SNO
              FROM SC
              WHERE CNO = 'c2');
```

查询结果如图 3.37 所示。

图 3.36 例 3-52 的 PL/SQL 程序运行效果

图 3.37 例 3-53 的 PL/SQL 程序运行效果

例 3-54 查询选修了课程名为 java 的学生学号和姓名。

```
SELECT SNO,SNAME
FROM STUDENT
```

```
WHERE SNO IN
          (SELECT SNO
           FROM SC
           WHERE CNO IN
          (SELECT CNO
           FROM COURSE
           WHERE CNAME = 'java'));
```

查询结果如图 3.38 所示。

此例也可以采用多表连接方法来实现。

```
SELECT STUDENT.SNO,SNAME
FROM STUDENT,SC,COURSE
WHERE STUDENT.SNO = SC.SNO AND COURSE.CNO = SC.CNO
AND CNAME = 'java';
```

查询结果和图 3.38 相同。

2）带有比较运算符的子查询

带有比较运算符的子查询是指父查询与子查询之间用比较运算符进行连接。只有当内层查询返回的是单值时，才可以用＞、＜、＝、＞＝、＜＝、!＝或＜＞等比较运算符。

例 3-55 查询与学号“20180001”学生在同一系别的学生学号和姓名。

```
SELECT SNO,SNAME
FROM STUDENT
WHERE DEPT = ( SELECT DEPT
               FROM STUDENT
               WHERE SNO = '20180001');
```

查询结果如图 3.39 所示。

SNO	SNAME
20180002	吴二
20180003	张三
20180005	王五
20180009	郑九

图 3.38 例 3-54 的 PL/SQL 程序运行效果

SNO	SNAME
20180001	周一
20180003	张三

图 3.39 例 3-55 的 PL/SQL 程序运行效果

也可以用前面学习的 IN 谓词来实现。

```
SELECT SNO,SNAME
FROM STUDENT
WHERE DEPT IN ( SELECT DEPT
                FROM STUDENT
                WHERE SNO = '20180001');
```

注意：当子查询的结果是单个值时，谓词 IN 和“＝”的作用是等价的；当子查询的结果是多个值时，只能用谓词 IN，而不能用“＝”了。

3）带有 ANY 谓词或 ALL 谓词的子查询

使用 ANY 或 ALL 谓词前必须同时使用比较运算符，含义如表 3.5 所示。

表 3.5　ANY 和 ALL 谓词的使用含义

ANY 或 ALL 谓词前的比较运算符	含　义
>ANY	大于子查询结果集中的某个值
>ALL	大于子查询结果集中的所有值
<ANY	小于子查询结果集中的某个值
<ALL	小于子查询结果集中的所有值
>=ANY	大于等于子查询结果集中的某个值
>=ALL	大于等于子查询结果集中的所有值
<=ANY	小于等于子查询结果集中的某个值
<=ALL	小于等于子查询结果集中的所有值
=ANY	等于子查询结果集中的某个值
=ALL	等于子查询结果集中的所有值(无意义)
<>ANY	不等于子查询结果集中的某个值(无意义)
<>ALL	不等于子查询结果集中的任何一个值

注意：<>ALL 等价于 NOT IN；=ANY 等价于 IN；=ALL、<>ANY 没有意义。

例 3-56　查询选修课程号为 c2 的学生姓名。(与例 3-52 相同,IN 与=ANY 等价。)

```
SELECT SNAME
FROM STUDENT
WHERE SNO  = ANY
                ( SELECT SNO
                  FROM SC
                  WHERE CNO = 'c2');
```

查询结果如图 3.40 所示。

若此题换成查询没有选修课程号为 c2 的学生姓名,则只需将=ANY 换成<>ALL。因为<>ALL 与 NOT IN 等价。

例 3-57　查询比所有男同学年龄都大的女同学的学号、姓名和年龄。

```
SELECT SNO,SNAME,AGE
FROM STUDENT
WHERE SEX = '女' and AGE > all
                          (SELECT AGE
                           FROM STUDENT
                           WHERE SEX = '男');
```

查询结果如图 3.41 所示。

SNAME
周一
张三
李四
郑九

图 3.40　例 3-56 的 PL/SQL 程序运行效果

SNO	SNAME	AGE
20180007	陈七	23

图 3.41　例 3-57 的 PL/SQL 程序运行效果

用聚组函数实现子查询通常比直接用 ANY 或 ALL 谓词查询效率高,ANY 或 ALL 谓词与聚组函数的对应关系如表 3.6 所示。

表 3.6 ANY 或 ALL 谓词与聚组函数的对应关系

比较运算符	ANY	ALL
=	IN	无意义
< >	无意义	NOT IN
<	< MAX	< MIN
<=	<= MAX	<= MIN
>	> MIN	> MAX
>=	>= MIN	>= MAX

例 3-58 查询其他系中比数学系某一学生年龄大的学生姓名和年龄。

方法一：

```
SELECT SNAME,AGE
FROM STUDENT
WHERE DEPG<>'数学系'
and Age>ANY ( SELECT AGE
              FROM STUDENT
              WHERE DEPT='数学系');
```

查询结果如图 3.42 所示。

方法二：

```
SELECT SNAME,AGE
FROM STUDENT
WHERE DEPT<>'数学系'
and AGE> (SELECT MIN(AGE)
          FROM STUDENT
          WHERE DEPT='数学系');
```

查询结果和图 3.42 相同。

例 3-59 查询其他系中比数学系所有学生年龄都大的学生姓名和年龄。

方法一：

```
SELECT SNAME,AGE
FROM STUDENT
WHERE DEPT<>'数学系'
and AGE>ALL ( SELECT AGE
              FROM STUDENT
              WHERE DEPT='数学系');
```

查询结果如图 3.43 所示。

SNAME	AGE
吴二	20
李四	22
陈七	23
刘八	21
孙十	21

图 3.42 例 3-58 的 PL/SQL 程序运行效果

SNAME	AGE
陈七	23

图 3.43 例 3-59 的 PL/SQL 程序运行效果

方法二：

```
SELECT SNAME,AGE
FROM STUDENT
WHERE DEPT<>'数学系'
and  AGE > (SELECT MAX(AGE)
            FROM STUDENT
            WHERE DEPT = '数学系');
```

查询结果和图 3.43 相同。

2. 相关子查询

前面我们介绍的子查询都是不相关子查询，不相关子查询比较简单，在整个过程中子查询只执行一次，并且把结果用于父查询，即子查询不依赖于外层父查询。而更复杂的情况是子查询要多次执行，子查询的查询条件依赖于外层父查询的某个属性值，称这类查询为相关子查询。

相关子查询的特点如下：

(1) 执行父查询，后执行子查询。

(2) 子查询不能独立运行，子查询的条件依赖外层父查询中取的值。

(3) 子查询多次运行。

1) 带有比较运算符的相关子查询

例 3-60 查询所有课程成绩均及格的学生学号和姓名。

```
SELECT SNO,SNAME
FROM STUDENT
WHERE 60 <= (SELECT MIN(GRADE)
             FROM SC
             WHERE STUDENT.SNO = SC.SNO);
```

查询结果如图 3.44 所示。

2) 有 EXISTS 谓词的子查询

在相关子查询中经常使用 EXISTS 谓词。带有 EXISTS 谓词的子查询不返回任何数据，只产生逻辑真值“true”或逻辑假值“false”。

若内层查询结果非空，则外层的 WHERE 子句返回真值。

若内层查询结果为空，则外层的 WHERE 子句返回假值。

由 EXISTS 引出的子查询，其目标列表达式通常都用 * ，因为带 EXISTS 的子查询只返回真值或假值，给出列名无实际意义。

例 3-61 查询选修课程号为 c2 的学生姓名。(与例 3-52、例 3-56 相同。)

```
SELECT SNAME
FROM STUDENT
WHERE EXISTS
             (SELECT *
              FROM SC
              WHERE STUDENT.SNO = SC.SNO AND CNO = 'c2');
```

查询结果如图 3.45 所示。

SNO	SNAME
20180001	周一
20180003	张三
20180004	李四
20180006	赵六

图 3.44 例 3-60 的 PL/SQL 程序运行效果

SNAME
周一
张三
李四
郑九

图 3.45 例 3-61 的 PL/SQL 程序运行效果

执行过程：首先取外层查询中 Student 表的第一行元组，根据它与内层查询相关属性值(Sno)来处理内层查询，若内层查询结果非空，则 EXISTS 为真，就把 Student 表的第一行元组中 Sname 值取出放入查询结果的结果集中；然后取 Student 表的第二行、第三行、……重复上述过程，直到 Student 表中所有行全部被检索完为止。

与 EXISTS 谓词相对应的是 NOT EXISTS 谓词。

若内层查询结果非空，则外层的 WHERE 子句返回假值。

若内层查询结果为空，则外层的 WHERE 子句返回真值。

例 3-62 查询没有选修课程号为 c2 的学生姓名。

```
SELECT SNAME
FROM STUDENT
WHERE NOT EXISTS
              (SELECT *
               FROM SC
               WHERE STUDENT.SNO = SC.SNO AND CNO = 'c2');
```

查询结果如图 3.46 所示。

例 3-63 查询没有选课的学生学号和姓名。

```
SELECT SNO,SNAME
FROM STUDENT
WHERE NOT EXISTS
              ( SELECT *
                FROM SC
                WHERE STUDENT.SNO = SC.SNO);
```

查询结果如图 3.47 所示。

SNAME
吴二
王五
赵六
陈七
刘八
孙十

图 3.46 例 3-62 的 PL/SQL 程序运行效果

SNO	SNAME
20180010	孙十

图 3.47 例 3-63 的 PL/SQL 程序运行效果

例 3-64 假设全体同学都选修了相应的课程且有成绩，那么查询所有课程成绩均大于 80 分的学生学号和姓名。

```
SELECT SNO,SNAME
FROM STUDENT
```

```
WHERE NOT EXISTS
              (SELECT *
               FROM SC
               WHERE STUDENT.SNO = SC.SNO AND GRADE <= 80);
```

查询结果如图 3.48 所示。

如果没有假设学生都选修了课程并且有成绩，那么在此查询结果集中将会出现没有选修任何课程的学生姓名和选修了课程但没有成绩的学生姓名，很显然，这些不是我们所想要的结果。

例 3-65 查询选修全部课程的学生姓名。（相当于查询这样的学生，没有一门课程是他不选的。）

```
SELECT SNAME
FROM STUDENT
WHERE NOT EXISTS
              ( SELECT *
                FROM COURSE
                WHERE NOT EXISTS
                          ( SELECT *
                            FROM SC
                            WHERE STUDENT.SNO = SC.SNO and
                            SC.CNO =  COURSE.CNO));
```

查询结果如图 3.49 所示。

SNO	SNAME
20180004	李四
20180008	刘八
20180010	孙十

图 3.48 例 3-64 的 PL/SQL 程序运行效果

SNAME
郑九

图 3.49 例 3-65 的 PL/SQL 程序运行效果

3.4.6 实践环节：数据的查询

根据 3.2.4 节中所创建的两张表——部门信息表 Dept(Deptno,Dname,Loc)和雇员信息表 Emp(Empno,Ename,Age,Sal,Deptno)，以及 3.3.4 节中所插入的数据，用 SQL 语句完成下列操作。

(1) 查询所有部门的详细记录。

(2) 查询所有雇员的姓名和年薪(年薪＝月薪×12)。

(3) 查询姓李的员工姓名。

(4) 查询年龄大于 40 岁并且月薪小于 1000 的员工编号和姓名。

(5) 查询月薪为空的刚入职的员工姓名。

(6) 查询部门编号为 10 的员工最高工资和最低工资。

(7) 查询每一个部门的员工人数。

(8) 查询平均工资超过 5000 的部门编号。

(9) 查询所有员工的姓名、部门号及月薪,查询结果按照部门号升序排列,同一部门按照月薪降序排列。

(10) 查询研发部年龄超过35岁且月薪超过8000元的员工姓名。

3.5 小结

SQL称为结构化查询语言,在许多关系数据库管理系统中均可使用,其功能并非仅局限于查询,它集数据定义、数据查询、数据操纵、数据控制功能于一体。

SQL语言的数据定义功能包括基本表、索引、视图的创建、修改和删除操作。

SQL语言的数据操纵功能包括数据插入、修改和删除操作。

SQL语言的数据查询功能是最丰富的,也是最复杂的。它是本章要求重点掌握的内容,包括单表查询、连接查询、嵌套查询、集合查询等。查询语句中可以使用聚组函数完成相关计算,可以使用分组子句将查询结果按某一属性列的值分组,可以使用排序子句将查询结果按指定的属性列进行排序输出。

习题3

一、选择题

1. SQL语言通常称为(　　)。

A. 结构化操纵语言　　B. 结构化控制语言

C. 结构化定义语言　　D. 结构化查询语言

2. 下列SQL语句命令,属于数据定义语言的是(　　)。

A. SELECT　　B. CREATE

C. GRANT　　D. DELETE

3. 以下操作不属于数据更新的(　　)。

A. 插入　　B. 删除　　C. 修改　　D. 查询

4. 在创建基本表的过程中,下列说法正确的是(　　)。

A. 在一个数据库中,两个基本表的名字可以相同

B. 表名和属性列的名字不区分大小写

C. 在给表命名时,第一个字符必须是字母或数字

D. 在给表中的属性列命名时,第一个字符必须是字母或数字

5. Oracle提供了5种约束条件保证数据的完整性和参考完整性,其中主键约束包含该键上的每一列的两种约束是(　　)。

A. 非空约束和检查约束　　B. 非空约束和唯一约束

C. 唯一约束和检查约束　　D. 外键约束和检查约束

6. 下列说法错误的是(　　)。

A. 主键约束既可以作为列级完整性约束条件,也可以作为表级完整性约束条件

B. 唯一约束可以作为列级完整性约束条件,但不可以作为表级完整性约束条件

C. 非空约束可以作为列级完整性约束条件,但不可以作为表级完整性约束条件

D. 外键约束既可以作为列级完整性约束条件,也可以作为表级完整性约束条件

7. 在 SQL 语句中,对分组情况满足的条件进行判断的语句是(　　)。

A. HAVING　　B. ORDER BY　　C. WHERE　　D. GROUP BY

8. 连接两个表 M 与 N 中的数据,形成一个结果集,并在会话中显示这个结果。表 M 与表 N 有一个共享列,在两个表中都称为 W。即使表 N 中没有相应数值,下列选项中的 WHERE 子句可以显示表 M 中 W 列为 2 的数据的是(　　)。

A. WHERE M. W=2 and M. W=N. W;

B. WHERE M. W=2;

C. WHERE M. W=2 and M. W(+)=N. W(+);

D. WHERE M. W=2 and M. W=N. W(+);

9. 查询所有成绩均大于 80 分的学生姓名,应执行(　　)语句。

A. SELECT SNAME FROM STUDENT WHERE EXISTS(SELECT * FROM SC WHERE STUDENT. SNO=SC. SNO and GRADE>80);

B. SELECT SNAME FROM STUDENT WHERE EXISTS(SELECT * FROM SC WHERE STUDENT. SNO=SC. SNO and GRADE<=80);

C. SELECT SNAME FROM STUDENT WHERE NOT EXISTS(SELECT * FROM SC WHERE STUDENT. SNO=SC. SNO and GRADE<=80);

D. SELECT SNAME FROM STUDENT WHERE NOT EXISTS(SELECT * FROM SC WHERE STUDENT. SNO=SC. SNO and GRADE>80);

10. 下列描述不正确的是(　　)。

A. 向表中插入数据之前,要先有表的结构

B. 插入的数据及列名之间用逗号分开

C. 在 INSERT 语句中列名是可以选择指定的,如果没有指定列名,则表示这些列按表中或视图中列的顺序和个数

D. 插入值的数据类型、个数、前后顺序不用与表中属性列的数据类型、个数、前后顺序匹配

11. 将 java 课程的学分改为 4 学分,正确的语句是(　　)。

A. UPDATE COURSE SET CREDIT=4 WHERE CNAME='java';

B. UPDATE COURSE SET CREDIT=4 WHERE CNAME='JAVA';

C. UPDATE COURSE VALUES CREDIT=4 WHERE CNAME='java';

D. UPDATE COURSE VALUES CREDIT=4 WHERE CNAME='JAVA';

12. 下列有关 UPDATE 语句描述错误的是(　　)。

A. UPDATE 关键字用于定位修改哪一张表

B. SET 关键字用于定位修改这张表中的哪些属性列

C. WHERE <条件>用于定位修改这些属性列当中的哪些行

D. 在 UPDATE 语句中不可以嵌套子查询

13. 下列有关 DELETE 语句描述错误的是(　　)。

A. DELETE 语句的功能是从指定表中删除满足 WHERE <条件>的所有元组

B. DELETE 语句既可以删除表中的数据,也可以删除表的结构

C. 在 DELETE 语句中可以嵌套子查询

D. 在删除表中的数据时,应满足定义表时设定的约束条件

二、上机实验题

某数据库中包含科室表和医生表。

科室表:dept_table(deptno,dname,loc),表中属性列依次是科室编号、科室名称、科室所在地点。科室表结构如下:

列　名	数据类型	长　度	完整性约束
deptno	CHAR	8	主键
dname	VARCHAR	15	唯一
loc	VARCHAR	9	无

医生表:doctor_table(docno,docname,age,sal,deptno),表中属性列依次是医生编号、医生姓名、年龄、工资、所在科室编号。医生表结构如下:

列　名	数据类型	长　度	完整性约束
docno	CHAR	8	主键
docname	VARCHAR	10	非空
age	INT	无	年龄在 18～55 岁
sal	NUMBER	无	无
deptno	CHAR	8	外键(参照 dept_table 表中的 deptno)

(1) 用 SQL 语句实现以下基本表的创建。

① 科室表(dept_table)的创建。

② 医生表(doctor_table)的创建。

(2) 根据各表结构,用 SQL 语句完成下列操作。

① 将医生表 doctor_table 中的 docname 列的数据类型修改为 CHAR(20)。

② doctor_table 表中添加一个名为 chk_sal 的约束,从而保证医生的工资必须大于 800。

③ 将张三医生的工资调高 20%。

④ 删除科室号为 10,工资低于 2000 的医生信息。

⑤ 查询各个科室的科室名称及其医生人数,查询结果按照医生人数的升序排列,如果人数相同,按照科室名称降序排列。

⑥ 查询各科室中至少有 2 个人工资在 2500 以上的科室号和医生人数。

⑦ 查询平均工资超过 3000 的科室编号。

⑧ 查询孙七医生所在的科室名称及科室地点。

⑨ 查询姓赵的医生的姓名、年龄和所在科室名称。

⑩ 建一个“口腔科”科室的视图 dept_10,包括医生编号、医生姓名及工资。