第3章

关系数据库标准语言SQL

视频

3.1 SQL 概述

3.1.1 SQL 简介

结构化查询语言(SQL)是一种通用的、功能极强的关系数据库标准语言,用于存取数据以及查询、更新和管理关系数据库系统。同时它也是数据库脚本文件的扩展名。

SQL 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国际标准语言,各个数据库厂家纷纷推出各自的 SQL 软件或 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。

目前,很多数据库产品都对 SQL 语句进行了再开发与扩展,如 Oracle 提供的 PL/SQL (Procedure Language/SQL,过程化 SQL)就是对 SQL 的一种扩展。

3.1.2 SQL 发展历程及标准化

1. SQL 的发展

SQL 随着数据库技术的发展而不断更新、丰富,其发展主要经历以下几个阶段:

(1) 1970 年,E. F. Codd 发表了关系数据库理论(Relational Database Theory);

(2) 1972 年,IBM 在 System R 中实现了 SQUARE 语言;

(3) 1974 年,Boyce 和 Chamberlin 在 SQUARE 语言的基础上进行改进,开发了 SEQUEL,并重命名为“结构化查询语言”;

(4) 1981 年,IBM 公司在 SYSTEM R 的基础上推出了商品化的关系数据库管理系统

SQL/DS，并且用 SQL 代替 SEQUEL；

（5）当今，SQL 广泛应用于各种大、中型数据库，如 Oracle、SQL server、DB2、Sybase、MySQL 等；也用于各种小型数据库，如 Access、FoxPro、SQLite 等。

2. SQL 标准化

SQL 功能强大，简单易学，一经推出就受到用户和计算机工业界的欢迎，因而，从 1982 年起，美国国家标准协会开始着手 SQL 的标准化工作，30 多年来已制定了多个 SQL 标准。

（1）1982 年，美国国家标准协会（American National Standard Institute，ANSI）开始制定 SQL 标准；

（2）1986 年，ANSI 公布了 SQL 的第一个标准 SQL—86；

（3）1987 年，国际标准化组织（International Organization for Standardization，ISO）将 SQL—86 标准采纳为国际标准；

（4）1989 年，ISO 对 SQL—86 标准进行了补充，增加了引用完整性，推出了 SQL—89 标准；

（5）1992 年，ISO 推出了 SQL92 标准（也称 SQL2）；

（6）1999 年，ISO 推出了 SQL99 标准（也称 SQL3），它增加了对象数据、递归和触发器等支持功能；

（7）2003 年，ISO 推出了 ISO/IEC 9075：2003 标准，简称 SQL：2003（也称 SQL4）；

（8）2008 年，ISO 推出了 ISO/IEC 9075：2008 标准，简称 SQL：2008；

（9）2011 年，ISO 推出了 ISO/IEC 9075：2011 标准，简称 SQL：2011；

（10）2016 年，ISO 推出了 ISO/IEC 9075：2016 标准，简称 SQL：2016。

3.1.3　SQL 特点

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

1. 综合统一

SQL 集数据定义语言、数据操纵语言、数据查询语言、数据控制语言的功能于一体，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。

SQL 的核心内容包括如下数据语言。

（1）数据定义语言，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，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，用于实现各种不同的数据查询。

（4）数据控制语言，用于数据访问权限的控制。

2. 高度非过程化

SQL 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 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 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、修改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 既是独立的语言,又是嵌入式语言。作为独立的语言,它能够独立地用于联机交互的使用方式,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 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 的语法基本上是一致的。这种统一的语法结构提供两种不同的使用方式,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 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,如表 3.1 所示。而且 SQL 语法简单,接近英语口语,因此易学易用。

表 3.1 SQL 的动词

SQL 功能	命 令 动 词	SQL 功能	命 令 动 词
数据查询	SELECT	数据操纵	INSERT、UPDATE、DELETE
数据定义	CREATE、DROP、ALTER	数据控制	GRANT、REVOKE


视频

3.2 数据定义

通过 SQL 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 3.2 所示。

表 3.2 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	

由于视图的定义与查询操作有关,本节只介绍基本表和索引的数据定义。

3.2.1 基本数据类型

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 SQL 支持的数据类型。

各个厂家的 SQL 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

(1) 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。

(2) SMALLINT 定义数据类型为短整数类型，它的精度由执行机构确定。

(3) 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)。

(4) FLOAT(p)定义数据类型为浮点数值型，p 为指定的精度。

(5) REAL 定义数据类型为浮点数值型，它的精度由执行机构确定。

(6) DOUBLE PRECISION 定义数据类型为双精度浮点型，它的精度由执行机构确定。

2. 字符类型

(1) CHAR(n)定义指定长度的字符串，n 为字符数的固定长度。

(2) 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

(1) BIT(n)定义数据类型为二进制位串，其长度为 n。

(2) 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

(1) DATE 用于定义日期，包含年、月、日，格式为 YYYY-MM-DD。

(2) 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。

3.2.2　约束条件

在 SQL 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码约束、外码约束、属性约束等几类。

1. 主码约束

主码(PRIMARY KEY)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。主码约束也称主键约束。

2. 外码约束

外码(FOREIGN KEY)约束体现参照完整性。外码的取值或者为空，或者参考父表的主码。外码约束也称外键约束。

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。

(1) 非空(NOT NULL)约束：要求某一属性的值不允许为空值。

(2) 唯一(UNIQUE)约束：要求某一属性的值不允许重复。

(3) 检查(CHECK)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。

上述几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，而其他约束可以作为表级完整性约束条件。

3.2.3 基本表的定义

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。

SQL 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相同的基本表名字。

(3) <列名>：规定了该列(属性)的名称，一个表中不能有两个相同的列名字。

(4) 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、三个特殊符号(_、$、#)；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。

(5) <数据类型>：规定了该列的数据类型。

(6) <列级完整性约束>：是指对某一列设置的约束条件。

(7) <表级完整性约束>：规定了关系主码、外码和用户自定义完整性约束。

例题 3.1 创建一个学生表(student)，要求所有约束条件均为列级完整性约束，学生表的结构如表 3.3 所示。

表 3.3 学生表(student)

字 段 名	字 段 类 型	是 否 为 空	说 明	字 段 描 述
sno	CHAR(8)	NOT NULL	主码	学生学号
sname	VARCHAR2(20)		唯一	学生姓名
sex	CHAR(4)	NOT NULL	非空	性别
age	INT		年龄大于 16 岁	年龄
dept	VARCHAR2(20)			学生所在的系别名称

学生表(student)的创建语句如下：

```
CREATE TABLE student
(
    sno CHAR(8) PRIMARY KEY,                    /*主码约束*/
    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(20)
);
```

例题 3.2　创建一个课程表(course)，要求所有约束条件均为列级完整性约束，课程表的结构如表 3.4 所示。

表 3.4　课程表(course)

字　段　名	字段类型	是否为空	说　　明	字段描述
cno	CHAR(8)	NOT NULL	主码	课程编号
cname	VARCHAR2(20)	NOT NULL	非空	课程名称
tname	VARCHAR2(20)			授课教师名
cpno	CHAR(8)		外码(参照课程表中的课程编号)	先修课程号
credit	NUMBER			学分

课程表(course)的创建语句如下：

```
CREATE TABLE course
(
    cno CHAR(8) PRIMARY KEY,                    /*主码约束*/
    cname VARCHAR2(20) NOT NULL,                /*非空约束*/
    tname VARCHAR2(20),
    cpno CHAR(8) REFERENCES course(cno),        /*外码约束*/
    credit NUMBER
);
```

例题 3.3　创建一个选课表(sc)，要求所有约束条件均为表级完整性约束，选课表的结构如表 3.5 所示。

表 3.5　选课表(sc)

字　段　名	字段类型	是否为空	说　　明	字段描述
sno	CHAR(8)	NOT NULL	外码(参照学生表中的学生编号)	学生学号
cno	CHAR(8)	NOT NULL	外码(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)属性组合为主码。

选课表(sc)的创建语句如下：

```
CREATE TABLE sc
(
    sno CHAR(8),
```

```
    cno CHAR(8),
    grade NUMBER,
    PRIMARY KEY(sno,cno),                          /* 主码约束 */
    FOREIGN KEY(sno) REFERENCES student(sno),      /* 外码约束 */
    FOREIGN KEY (cno) REFERENCES course(cno)       /* 外码约束 */
);
```

3.2.4 基本表的修改

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 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。

(5) 以上的命令格式在实际的数据库管理系统中可能有所不同,用户在使用时应参阅实际数据库系统的参考手册。

例题 3.4 向 student 表中增加一个身高(height)属性列,数据类型为 INT。

```
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。

例题 3.5 将 student 表中的 height 属性列的数据类型改为 REAL。

```
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。

例题 3.6 给 student 表中 height 属性列增加一个检查约束,约束的名字为 CHK_HEIGHT,要求学生的身高需超过 140cm。

```
ALTER TABLE student ADD CONSTRAINT CHK_HEIGHT CHECK(height > 140);
```

例题 3.7　删除 height 属性列上的 CHECK 约束。

```
ALTER TABLE student DROP CONSTRAINT CHK_HEIGHT;
```

例题 3.8　删除 student 表中新增加的 height 属性列。

```
ALTER TABLE student DROP COLUMN height;
```

3.2.5　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码约束条件，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。

例题 3.9　删除学生选课表(sc)。

```
DROP TABLE sc;
```

基本表定义一旦被删除，表中的数据、此表上建立的索引和视图都将被自动删除。因此执行删除基本表的操作时一定要格外小心。但在有的系统(如 Oracle)中，删除基本表后建立在此表上的视图定义仍然保留在数据字典中，但是，当用户引用时就会报错。

3.2.6　索引的定义和删除

基本表建立并存放数据后，如果数据相当多，DBMS 则会在顺序扫描上耗费很长的时间，这样将大大影响查询效率。为了解决查询速度问题，需要在针对数据表的查询字段上定义索引。索引提供了一种直接、快速访问记录的方式，可以大大提高数据查询速度。

索引是根据表中一列或若干列按照一定顺序建立的列值与记录行之间的对应关系表。索引属于物理存储的路径概念，而不是用户使用的逻辑概念。建立在多个列上的索引被称为复合索引。系统在存取数据时会自动选择合适的索引作为存取路径，用户不必也不能显式地选择索引。

有两种重要的索引：聚簇索引(Clustered Index)和非聚簇索引(Non-Clustered Index)。

聚簇索引确定表中数据的物理顺序，它类似于按姓氏排列数据的电话簿。由于聚簇索引规定数据在表中的物理存储顺序，因此一个表只能包含一个聚簇索引。建立聚簇索引后，

更新索引列数据时,往往会导致表中记录的物理顺序变更,代价较大,因此对于经常更新的列不宜建立聚簇索引。

非聚簇索引是数据存储在一个地方,索引存储在另一个地方,索引带有指针指向数据的存储位置。索引中的项目按索引码值的顺序存储,而表中的信息按另一种顺序存储。

1. 索引的定义

在 SQL 中,建立索引使用 CREATE INDEX 语句,其一般格式为:

```
CREATE [UNIQUE] [CLUSTER] INDEX <索引名>
ON <基本表名> (<列名> [<次序>],[,<列名> [<次序>]]…);
```

说明:

(1) UNIQUE:规定此索引为唯一性索引。每一个索引值只对应于表中唯一的记录。

(2) CLUSTER:规定此索引为聚簇索引。省略 CLUSTER 则表示创建的索引为非聚簇索引。

(3) <次序>:建立索引时指定列名的索引表是 ASC(升序)或 DESC(降序)。若不指定,默认为升序。

例题 3.10 为 student、course、sc 三张表建立索引。其中 student 表按学号(sno)升序建唯一索引,course 表按课程号(cno)降序建唯一索引,sc 表按学号(sno)升序和课程号(cno)降序建唯一索引。

```
CREATE UNIQUE INDEX  index_stu ON student(sno ASC);
CREATE UNIQUE INDEX  index_cou ON course(cno DESC);
CREATE UNIQUE INDEX  index_sc  ON sc(sno ASC,cno DESC);
```

2. 索引的删除

索引可以加快查询速度,但如果数据的增、删、改操作很频繁,系统就会花许多时间来维护索引,导致系统开销增加。

索引一经建立,就由系统使用和维护,不需用户干预。建立索引是为了减少查询操作的时间,但如果数据增、删、改频繁,系统会花费许多时间来维护索引。过多的索引甚至会导致索引碎片,降低系统效率。因此不必要的索引应该及时删除。

删除索引的格式为:

```
DROP INDEX <索引名>;
```

例题 3.11 删除 course 表的 index_cou 索引。

```
DROP INDEX index_cou;
```

删除索引时,系统会同时从数据字典中删除有关该索引的描述。


视频

3.3 数据查询

3.3.1 SELECT 语句格式

SQL 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 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]];
```

说明：

(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 表示输出结果中无重复记录。

(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 SQL 函数和 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，省略不写的默认情况下是 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集函数。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。下面将通过大量的例子来介绍 SELECT 语句的功能。

下面，以学生选课系统为例说明 SELECT 语句的各种用法。学生选课系统中包含如下三张表。

(1) 学生表：student(sno,sname,sex,age,dept)，表中属性列依次是学生学号、姓名、性别、年龄、学生所在的系别名称，其中 sno 为主码，表中数据如表 3.6 所示。

表 3.6　学生基本信息

sno	sname	sex	age	dept
10172001	陈一	男	17	计算机系
10172002	姚二	女	20	计算机系
10172003	张三	女	19	计算机系
10172004	李四	男	22	日语系
10172005	王五	男	22	日语系

续表

sno	sname	sex	age	dept
10172006	赵六	男	19	日语系
10172007	陈七	女	23	信息系
10172008	刘八	男	21	信息系
10172009	张九	女	18	管理系
10172010	孙十	女	21	管理系

(2) 课程表：course(cno,cname,tname,cpno,credit)，表中属性列依次是课程编号、课程名称、授课教师名、先修课程号和学分，其中 cno 为主码，表中数据如表 3.7 所示。

表 3.7 课程基本信息

cno	cname	tname	cpno	credit
c1	maths	曹老师		3
c2	english	赵老师		5
c3	japanese	刘老师		4
c4	database	杨老师	c1	3
c5	java	陈老师	c1	3
c6	jsp_design	陈老师	c5	2

(3) 选课表：sc(sno,cno,grade)，表中属性列依次是学号、课程号和成绩，其中属性组合(sno,cno)为主码，表中数据如表 3.8 所示。

表 3.8 选课基本信息

sno	cno	grade
10172001	c1	94
10172001	c2	96
10172001	c4	92
10172002	c1	50
10172002	c2	88
10172002	c4	76
10172002	c5	55
10172003	c1	65
10172003	c2	72
10172003	c3	90
10172003	c4	85
10172003	c5	93
10172003	c6	86
10172004	c2	50
10172004	c3	45
10172005	c2	88
10172005	c3	85
10172007	c1	80
10172007	c2	73

续表

sno	cno	grade
10172007	c3	66
10172007	c4	
10172007	c5	
10172008	c1	82
10172008	c2	77
10172008	c3	85
10172008	c4	87
10172008	c5	82
10172008	c6	94
10172009	c1	86
10172009	c2	
10172010	c1	73
10172010	c2	

3.3.2　单表无条件查询

单表查询是指查询的数据只来自一张表，此时，SELECT 语句中的 FROM 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

例题 3.12　查询全体学生的学号、姓名和年龄。

```
SELECT sno,sname,age
FROM student;
```

结果如下：

```
SNO         SNAME          AGE
--------  -----------  ------------
10172001    陈一           17
10172002    姚二           20
10172003    张三           19
10172004    李四           22
10172005    王五           22
10172006    赵六           19
10172007    陈七           23
10172008    刘八           21
10172009    张九           18
10172010    孙十           21
```

例题 3.13　查询全部课程的课程名称和授课教师名。

```
SELECT cname,tname
```

```
FROM course;
```

结果如下：

```
CNAME           TNAME
------------  --------
maths           曹老师
english         赵老师
japanese        刘老师
database        杨老师
java            陈老师
jsp_design      陈老师
```

2）查询全部列

例题 3.14 查询全部课程的详细记录。

```
SELECT   *
FROM course;
```

结果如下：

```
CNO          CNAME          TNAME          CPNO         CREDIT
---------  -----------  -----------  ----------  -----------
c1           maths          曹老师                        3
c2           english        赵老师                        5
c3           japanese       刘老师                        4
c4           database       杨老师         c1             3
c5           java           陈老师         c1             3
c6           jsp_design     陈老师         c5             2
```

当所查询的列是关系的所有属性时，可以使用“*”来表示所显示的列。

3）查询经过计算的值

例题 3.15 查询全体学生的姓名、性别及其出生年份。

```
SELECT sname,sex,2020 - age
FROM student;
```

结果如下：

```
SNAME        SEX        2020 - AGE
---------  ---------  ------------
陈一         男          2003
姚二         女          2000
张三         女          2001
李四         男          1998
王五         男          1998
赵六         男          2001
陈七         女          1997
刘八         男          1999
张九         女          2002
孙十         女          1999
```

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可以看出，显示的每一个属性列的标题就是列名，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 as，然后写上它的别名。在查询结果显示时就会用别名代替列名了。

例题 3.16　查询全体学生的姓名、性别及其出生年份。

```
SELECT sname,sex,2020 - age 出生年份
FROM student;
```

结果如下：

```
SNAME        SEX        出生年份
--------  ---------  -----------
陈一         男         2003
姚二         女         2000
张三         女         2001
李四         男         1998
王五         男         1998
赵六         男         2001
陈七         女         1997
刘八         男         1999
张九         女         2002
孙十         女         1999
```

2. 选择表中若干行

选择表中若干行，这就是选择运算。这里介绍无条件的选择运算，后面再介绍有条件的选择运算，需要注意的是：消除取值重复的行。

例题 3.17　查询所有选修了课程的学生学号。

```
SELECT sno
FROM sc;
```

结果如下：

```
SNO
-------
10172001
10172001
10172002
10172002
10172002
10172002
10172003
10172003
10172003
10172003
10172003
10172003
```

```
10172004
10172004
10172005
10172005
10172007
10172007
10172007
10172007
10172007
10172008
10172008
10172008
10172008
10172008
10172008
10172009
10172009
10172010
10172010
```

由于存在一名同学选修多门课程的情况,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 sno
FROM sc;
```

结果如下:

```
SNO
--------
10172001
10172002
10172003
10172004
10172005
10172007
10172008
10172009
10172010
```

3.3.3 单表有条件查询

单表查询时,若需满足某些条件则可以通过 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,则需要使用单引号把字符串括起来。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,则需要使用单引号把日期括起来。

(3) 如果该列数据类型为数值型,则不必用单引号,如 WHERE age>20。

(4) 在 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 3.9 所示。

表 3.9　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

例题 3.18　查询信息系全体学生的姓名。

```
SELECT sname
FROM student
WHERE dept = '信息系';
```

结果如下:

```
SNAME
-------
陈七
刘八
```

例题 3.19　查询年龄超过 20 岁的学生姓名及年龄。

```
SELECT sname,age
FROM student
WHERE age > 20;
```

结果如下:

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

例题 3.20　查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT sno
FROM sc
WHERE grade < 60;
```

结果如下:

```
SNO
--------
10172004
10172002
```

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)

例题 3.21 查询年龄为 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;
```

结果如下:

```
SNAME           AGE
--------  --------
陈一            17
姚二            20
张三            19
赵六            19
张九            18
```

例题 3.22 查询年龄不是 16～20 岁的学生姓名和年龄。

```
SELECT sname,age
FROM student
WHERE age NOT BETWEEN 16 AND 20;
```

结果如下:

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

3. 确定集合(谓词 IN)

例题 3.23 查询计算机系、信息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','信息系','管理系');
```

结果如下:

```
SNAME           SEX
--------  --------
陈一            男
```

```
姚二            女
张三            女
陈七            女
刘八            男
张九            女
孙十            女
```

例题 3.24　查询既不是计算机系、信息系，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','信息系','管理系');
```

结果如下：

```
SNAME           SEX
--------  --------
李四            男
王五            男
赵六            男
```

4. 字符匹配

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，也可以是含有通配符“%”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。

例题 3.25　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%';
```

结果如下：

```
SNAME           AGE     DEPT
--------  --------  --------------
张三            19      计算机系
张九            18      管理系
```

如果将 LIKE 换成 NOT LIKE，表示查询不姓张的同学。如果用户查询的匹配字符串本身就含有“%”或“_”，这时就要使用 ESCAPE'<换码字符>'短语对通配符进行转义。

例题 3.26　查询以 jsp 开头且倒数第 2 个字符为 g 的课程详细信息。

```
SELECT *
FROM course
WHERE cname LIKE 'jsp\_%g_' ESCAPE'\';
```

结果如下：

```
CNO    CNAME       TNAME     CPNO   CREDIT
------ ----------- --------- ------ --------
c6     jsp_design  陈老师    c5          2
```

5. 涉及空值的查询

例题 3.27 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;
```

结果如下：

```
SNO        CNO
---------- -----
10172007   c4
10172007   c5
10172009   c2
10172010   c2
```

注意：程序中的 IS 不能用等号(=)代替。

例题 3.28 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;
```

结果如下：

```
SNO        CNO
---------- -----
10172001   c1
10172001   c2
10172001   c4
10172002   c1
10172002   c2
10172002   c4
10172002   c5
10172003   c1
10172003   c2
10172003   c3
10172003   c4
10172003   c5
10172003   c6
10172004   c2
10172004   c3
10172005   c2
10172005   c3
10172007   c1
10172007   c2
10172007   c3
10172008   c1
```

```
10172008      c2
10172008      c3
10172008      c4
10172008      c5
10172008      c6
10172009      c1
10172010      c1
```

6. 多重条件查询

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

例题 3.29　查询计算机系女同学的姓名和年龄。

```
SELECT sname,age
FROM student
WHERE dept = '计算机系' AND sex = '女';
```

结果如下：

```
SNAME          AGE
-------  --------
姚二           20
张三           19
```

例题 3.30　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;
```

结果如下：

```
SNAME
-------
陈一
张三
赵六
张九
孙十
```

3.3.4　聚集函数

为了进一步方便用户，增强检索功能，SQL 提供了许多聚集函数，主要有如下几种。

(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。

在聚集函数遇到空值时,除 COUNT(*)外,都跳过空值而只处理非空值。

例题 3.31 查询学生表中的总人数。

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

结果如下:

```
COUNT(*)
--------
      10
```

例题 3.32 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT sno)
FROM  sc;
```

结果如下:

```
COUNT(DISTINCTSNO)
---------------
              9
```

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,所以必须添加 DISTINCT 关键字,表示在统计人数时,取消指定列中的重复值。

例题 3.33 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';
```

结果如下:

```
AVG(GRADE)  MAX(GRADE)  MIN(GRADE)
---------  ---------  ---------
     74.2         90          45
```

例题 3.34 查询学号为 10172001 的学生选修课程的成绩总和。

```
SELECT SUM(grade)
FROM sc
WHERE sno = '10172001';
```

结果如下:

```
SUM(GRADE)
---------
      282
```

3.3.5　分组查询和排序查询

1. 对查询结果分组

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，然后使用聚集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。

例题 3.35　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;
```

结果如下：

```
CNO         COUNT(SNO)
--------  ---------
c2                  9
c5                  4
c6                  2
c4                  5
c3                  5
c1                  7
```

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，然后对每一组用聚集函数 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。

例题 3.36　查询选修 5 门及其以上课程的学生学号。

```
SELECT sno
FROM sc
GROUP BY sno
HAVING COUNT(cno)>=5;
```

结果如下：

```
SNO
---------
10172003
10172007
10172008
```

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，要么包含聚集函数，否则出错。

(2) 可以使用多个属性列进行分组。

(3) 聚集函数只能够出现在 SELECT、HAVING、ORDER BY 子句中。在 WHERE 子

句中是不能使用聚集函数的。

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。WHERE 子句与 HAVING 子句的区别如下。

(1) WHERE 子句的作用是在分组之前过滤数据。WHERE 条件中不能包含聚集函数。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 条件中经常包含聚集函数。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。

2. 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。省略不写的则默认为升序排列。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。

例题 3.37 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;
```

结果如下:

```
SNO              GRADE
---------- ---------
10172003            90
10172005            85
10172008            85
10172007            66
10172004            45
```

例题 3.38 查询所有学生的基本信息,查询结果按学生年龄的降序排列,年龄相同时则按学号升序排列。

```
SELECT *
FROM  student
ORDER BY  age DESC, sno ASC;
```

结果如下:

```
SNO          SNAME    SEX    AGE    DEPT
-------- -------- ----- ----- --------
10172007     陈七     女     23     信息系
10172004     李四     男     22     日语系
10172005     王五     男     22     日语系
10172008     刘八     男     21     信息系
10172010     孙十     女     21     管理系
10172002     姚二     女     20     计算机系
10172003     张三     女     19     计算机系
10172006     赵六     男     19     日语系
```

```
10172009    张九    女    18    管理系
10172001    陈一    男    17    计算机系
```

3.3.6　连接查询

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类型必须是可比的，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。

例题 3.39　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;
```

结果如下：

```
SNO        SNAME    SEX    AGE    DEPT        SNO        CNO    GRADE
--------   ------   ----   -----  ---------   --------   ----   ------
10172001   陈一     男     17     计算机系    10172001   c1       94
10172001   陈一     男     17     计算机系    10172001   c2       96
10172001   陈一     男     17     计算计系    10172001   c4       92
10172002   姚二     女     20     计算机系    10172002   c1       50
10172002   姚二     女     20     计算机系    10172002   c2       88
10172002   姚二     女     20     计算机系    10172002   c4       76
10172002   姚二     女     20     计算机系    10172002   c5       55
10172003   张三     女     19     计算机系    10172003   c1       65
10172003   张三     女     19     计算机系    10172003   c2       72
10172003   张三     女     19     计算机系    10172003   c3       90
10172003   张三     女     19     计算机系    10172003   c4       85
10172003   张三     女     19     计算机系    10172003   c5       93
10172003   张三     女     19     计算机系    10172003   c6       86
10172004   李四     男     22     日语系      10172004   c2       50
```

10172004	李四	男	22	日语系	10172004	c3	45
10172005	王五	男	22	日语系	10172005	c2	88
10172005	王五	男	22	日语系	10172005	c3	85
10172007	陈七	女	23	信息系	10172007	c1	80
10172007	陈七	女	23	信息系	10172007	c2	73
10172007	陈七	女	23	信息系	10172007	c3	66
10172007	陈七	女	23	信息系	10172007	c4	
10172007	陈七	女	23	信息系	10172007	c5	
10172008	刘八	男	21	信息系	10172008	c1	82
10172008	刘八	男	21	信息系	10172008	c2	77
10172008	刘八	男	21	信息系	10172008	c3	85
10172008	刘八	男	21	信息系	10172008	c4	87
10172008	刘八	男	21	信息系	10172008	c5	82
10172008	刘八	男	21	信息系	10172008	c6	94
10172009	张九	女	18	管理系	10172009	c1	86
10172009	张九	女	18	管理系	10172009	c2	
10172010	孙十	女	21	管理系	10172010	c1	73
10172010	孙十	女	21	管理系	10172010	c2	

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，称为自然连接。

例题 3.40 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;
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT	CNO	GRADE
10172001	陈一	男	17	计算机系	c1	94
10172001	陈一	男	17	计算机系	c2	96
10172001	陈一	男	17	计算机系	c4	92
10172002	姚二	女	20	计算机系	c1	50
10172002	姚二	女	20	计算机系	c2	88
10172002	姚二	女	20	计算机系	c4	76
10172002	姚二	女	20	计算机系	c5	55
10172003	张三	女	19	计算机系	c1	65
10172003	张三	女	19	计算机系	c2	72
10172003	张三	女	19	计算机系	c3	90
10172003	张三	女	19	计算机系	c4	85
10172003	张三	女	19	计算机系	c5	93

```
10172003    张三    女    19    计算机系    c6    86
10172004    李四    男    22    日语系      c2    50
10172004    李四    男    22    日语系      c3    45
10172005    王五    男    22    日语系      c2    88
10172005    王五    男    22    日语系      c3    85
10172007    陈七    女    23    信息系      c1    80
10172007    陈七    女    23    信息系      c2    73
10172007    陈七    女    23    信息系      c3    66
10172007    陈七    女    23    信息系      c4
10172007    陈七    女    23    信息系      c5
10172008    刘八    男    21    信息系      c1    82
10172008    刘八    男    21    信息系      c2    77
10172008    刘八    男    21    信息系      c3    85
10172008    刘八    男    21    信息系      c4    87
10172008    刘八    男    21    信息系      c5    82
10172008    刘八    男    21    信息系      c6    94
10172009    张九    女    18    管理系      c1    86
10172009    张九    女    18    管理系      c2
10172010    孙十    女    21    管理系      c1    73
10172010    孙十    女    21    管理系      c2
```

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。

例题 3.41　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*/
```

结果如下：

```
SNO          SNAME      DEPT
--------  ---------  ----------
10172002     姚二       计算机系
```

连接操作除了可以用于两个表的连接外，还可以用于两个以上表的连接，称为多表连接。

例题 3.42　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*/
```

结果如下：

```
SNAME      TNAME      GRADE
------  ---------  ------
陈一       曹老师     94
姚二       曹老师     50
张三       曹老师     65
```

如果是多个表之间的连接，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。自身连接要求必须为表取别名，从而将它们当作两个不同的表来处理。

例题 3.43 在 sc 表中查询至少选修了课程号为 c3 和 c4 的学生学号。

在 sc 表中，每一条记录只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：一个为 x，另一个为 y。完成查询的语句为：

```
SELECT x.sno
FROM sc x,sc y
WHERE x.sno = y.sno                    /*连接条件*/
      AND x.cno = 'c3'                 /*限制条件*/
      AND y.cno = 'c4';                /*限制条件*/
```

结果如下：

```
SNO
--------
10172003
10172007
10172008
```

该例题中，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 c3 和 c4。

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。如例题 3.39 和例题 3.40 的结果中没有 10172006 赵六学生的信息，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(+)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。

例题 3.44 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，则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( + );
```

结果如下：

SNO	SNAME	SEX	AGE	DEPT	CNO	GRADE
10172001	陈一	男	17	计算机系	c1	94
10172001	陈一	男	17	计算机系	c2	96
10172001	陈一	男	17	计算机系	c4	92
10172002	姚二	女	20	计算机系	c1	50
10172002	姚二	女	20	计算机系	c2	88
10172002	姚二	女	20	计算机系	c4	76
10172002	姚二	女	20	计算机系	c5	55
10172003	张三	女	19	计算机系	c1	65
10172003	张三	女	19	计算机系	c2	72
10172003	张三	女	19	计算机系	c3	90
10172003	张三	女	19	计算机系	c4	85
10172003	张三	女	19	计算机系	c5	93
10172003	张三	女	19	计算机系	c6	86
10172004	李四	男	22	日语系	c2	50
10172004	李四	男	22	日语系	c3	45
10172005	王五	男	22	日语系	c2	88
10172005	王五	男	22	日语系	c3	85
10172007	陈七	女	23	信息系	c1	80
10172007	陈七	女	23	信息系	c2	73
10172007	陈七	女	23	信息系	c3	66
10172007	陈七	女	23	信息系	c4	
10172007	陈七	女	23	信息系	c5	
10172008	刘八	男	21	信息系	c1	82
10172008	刘八	男	21	信息系	c2	77
10172008	刘八	男	21	信息系	c3	85
10172008	刘八	男	21	信息系	c4	87
10172008	刘八	男	21	信息系	c5	82
10172008	刘八	男	21	信息系	c6	94
10172009	张九	女	18	管理系	c1	86
10172009	张九	女	18	管理系	c2	
10172010	孙十	女	21	管理系	c1	73
10172010	孙十	女	21	管理系	c2	
10172006	赵六	男	19	日语系		

3.3.7　嵌套查询

在 SQL 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,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便迎刃而解了。

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。

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: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。

例题 3.45 查询选修课程号为 c4 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 = 'c4';
```

结果如下:

```
SNAME
--------
陈一
姚二
张三
陈七
刘八
```

方法二:采用子查询来完成。

```
SELECT sname
FROM   student
WHERE  sno IN
          ( SELECT sno
            FROM   sc
            WHERE cno = 'c4');
```

结果如下:

```
SNAME
--------
陈一
姚二
张三
```

```
陈七
刘八
```

查询选修 c4 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都选修了 c4 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 c4 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。

例题 3.46　查询既没有选修课程号 c3，也没有选修课程号 c4 的学生学号。

```
SELECT sno
FROM student
WHERE sno NOT IN
              (SELECT sno
               FROM sc
               WHERE cno = 'c3' )
    AND sno NOT IN
               (SELECT sno
                FROM sc
                WHERE cno = 'c4');
```

结果如下：

```
SNO
--------
10172006
10172009
10172010
```

例题 3.47　查询选修了课程名为 database 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 = 'database'));
```

结果如下：

```
SNO          SNAME
---------  -----
10172003     张三
10172002     姚二
10172008     刘八
10172001     陈一
10102007     陈七
```

该例题也可以采用多表连接方法来实现，如下所示：

```
SELECT student.sno,sname
FROM   student,sc,course
WHERE  student.sno = sc.sno AND course.cno = sc.cno
       AND cname = 'database';
```

结果如下：

```
SNO        SNAME
--------  ------
10172001   陈一
10172002   姚二
10172003   张三
10172007   陈七
10172008   刘八
```

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。

例题 3.48 查询与学号 10172007 学生在同一系别的学生学号、姓名和系别名称。

```
SELECT sno,sname,dept
FROM student
WHERE dept = ( SELECT dept
               FROM student
               WHERE sno = '10172007');
```

结果如下：

```
SNO        SNAME    DEPT
--------  ------  --------
10172007   陈七     信息系
10172008   刘八     信息系
```

也可以用前面学习的 IN 谓词来实现，如下所示：

```
SELECT sno,sname,dept
FROM student
WHERE dept IN ( SELECT dept
                FROM student
                WHERE sno = '10172007');
```

结果如下：

```
SNO        SNAME    DEPT
--------  ------  --------
10172007   陈七     信息系
10172008   刘八     信息系
```

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 3.10 所示。

表 3.10　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	大于或等于子查询结果集中的某个值
>= ALL	大于或等于子查询结果集中的所有值
<= ANY	小于或等于子查询结果集中的某个值
<= ALL	小于或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。

例题 3.49　查询选修课程号为 c4 的学生姓名(与例题 3.45 相同,IN 与=ANY 等价)。

```
SELECT sname
FROM   student
WHERE sno  = ANY
             (SELECT sno
              FROM   sc
              WHERE cno = 'c4');
```

结果如下：

```
SNAME
--------
陈一
姚二
张三
陈七
刘八
```

若该例题换成查询没有选修课程号为 c4 的学生姓名,则只需将=ANY 换成<> ALL。因为<> ALL 与 NOT IN 等价。

例题 3.50　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');
```

结果如下：

```
SNO         SNAME          AGE
--------  -----------  ------------
10172007    陈七           23
```

使用聚集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 谓词与聚集函数的对应关系如表 3.11 所示。

表 3.11 ANY 或 ALL 谓词与聚集函数的对应关系

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

例题 3.51 查询其他系中比日语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  dept<>'日语系'
       AND age > ANY ( SELECT age
                       FROM student
                       WHERE dept = '日语系');
```

结果如下：

```
SNAME           AGE
--------  --------
陈七            23
孙十            21
刘八            21
姚二            20
```

方法二：

```
SELECT sname,age
FROM student
WHERE dept<>'日语系'
       AND age > (SELECT MIN(age)
                       FROM student
                       WHERE dept = '日语系');
```

结果如下：

```
SNAME           AGE
--------  --------
姚二            20
陈七            23
刘八            21
孙十            21
```

例题 3.52　查询其他系中比日语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  age > ALL ( SELECT age
                   FROM student
                   WHERE dept = '日语系');
```

结果如下：

```
SNAME          AGE
-------  -------
陈七           23
```

方法二：

```
SELECT sname,age
FROM student
WHERE age > (SELECT MAX(age)
             FROM student
             WHERE dept = '日语系');
```

结果如下：

```
SNAME          AGE
-------  -------
陈七           23
```

2. 相关子查询

前面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，这类查询称为相关子查询。

相关子查询的特点有：①先执行父查询，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

例题 3.53　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);
```

结果如下：

```
SNO         SNAME
--------  -----
10172001    陈一
```

```
10172003    张三
10172005    王五
10172007    陈七
10172008    刘八
10172009    张九
10172010    孙十
```

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。

例题 3.54 查询选修课程号为 c4 的学生姓名(与例题 3.45、例题 3.49 相同)。

```
SELECT sname
FROM  student
WHERE EXISTS
       (SELECT  *
        FROM  sc
        WHERE student.sno = sc.sno AND cno = 'c4');
```

结果如下:

```
SNAME
--------
陈一
姚二
张三
陈七
刘八
```

执行过程是: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、……第 n 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。

例题 3.55 查询没有选修课程号为 c4 的学生姓名。

```
SELECT sname
FROM  student
WHERE NOT EXISTS
        (SELECT  *
         FROM  sc
         WHERE student.sno = sc.sno AND cno = 'c4');
```

结果如下：

```
SNAME
--------
李四
王五
赵六
张九
孙十
```

例题 3.56　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);
```

结果如下：

```
SNO         SNAME
---------  -----
10172006    赵六
```

例题 3.57　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  sno IN
       (SELECT sno
        FROM sc
        WHERE  NOT EXISTS
               (SELECT   *
                FROM   sc
                WHERE student.sno = sc.sno AND grade < = 80));
```

结果如下：

```
SNO         SNAME
---------  -----
10172001    陈一
10172005    王五
10172009    张九
```

为了防止没有选课的学生 10172006 赵六出现在结果集里，所以上例中用到了两层嵌套查询。

3.3.8　集合查询

若把多个 SELECT 语句的结果合并为一个结果集，可用集合操作来完成。集合操作主

要包括并操作(UNION)、交操作(INTERSECT)和差操作(MINUS)。参加集合操作的各个结果表的列数必须相同,对应项的数据类型也必须相同。各个结果表中的列名可以不同。

1. 并操作

SQL 使用 UNION 语句把查询的结果合并起来,并且去掉重复的元组。

例题 3.58 查询计算机系和信息系的学生姓名的并集。

```
SELECT sname
FROM student
WHERE dept = '计算机系'
UNION
SELECT sname
FROM student
WHERE dept = '信息系';
```

结果如下:

```
SNAME
--------
陈七
陈一
刘八
姚二
张三
```

上述集合查询语句的结果等价于:

```
SELECT sname
FROM student
WHERE dept = '计算机系'OR dept = '信息系';
```

2. 交操作

SQL 使用 INTERSECT 语句把同时出现在两个查询的结果取出,实现交操作,并且也会去掉重复的元组。

例题 3.59 查询管理系的学生和年龄大于 20 岁的学生的交集。

```
SELECT *
FROM student
WHERE dept = '管理系'
INTERSECT
SELECT *
FROM student
WHERE age > 20;
```

结果如下:

```
SNO          SNAME    SEX    AGE    DEPT
-------- -------- ------ ------ --------
10172010     孙十     女     21     管理系
```

上述集合查询语句的结果等价于:

```
SELECT *
FROM student
WHERE dept = '管理系' AND age > 20;
```

3. 差操作

SQL 使用 MINUS 语句把出现在第一个查询结果中，但不出现在第二个查询结果中的元组取出，实现差操作。

例题 3.60　查询管理系的学生和年龄大于 20 岁的学生的差集。

```
SELECT *
FROM student
WHERE dept = '管理系'
MINUS
SELECT *
FROM student
WHERE age > 20;
```

结果如下：

```
SNO          SNAME    SEX    AGE    DEPT
--------  --------  -----  -----  --------
10172009     张九      女      18     管理系
```

上述集合查询语句的结果等价于：

```
SELECT *
FROM student
WHERE dept = '管理系' AND age < = 20;
```

3.4　数据操纵


视频

3.4.1　插入数据

当基本表建立以后，就可以使用 INSERT 语句向表中插入数据了。INSERT 语句有两种插入形式：插入单个元组和插入多个元组(插入子查询结果)。

1. 插入单个元组

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插入数据。

(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。

例题 3.61 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 ('10172011','小明','男',20,'计算机系');
```

方法二:指出所有列名。

```
INSERT INTO student(sno,sname,sex,age,dept)
VALUES ('10172011','小明','男',20,'计算机系');
```

两种方法的作用是相同的。

例题 3.62 向学生表中指定的属性列插入数据。

```
INSERT INTO student(sno,sname,sex)
VALUES ('10172012','小强','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 stduent 表中插入学号为 10172012 的学生记录时,系统就会给出错误提示信息:违反了主码约束。如果再插入另一个:10172013 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。

2. 插入多个元组

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>)] 子查询;
```

如果列名序列省略则子查询所得到的数据列必须和要插入数据的基本表的数据列完全一致;如果列名序列给出则子查询结果与列名序列要一一对应。

例题 3.63 如果已经创建了课程平均成绩记录表 course_avg(cno,ave),其中 ave 表示每门课程的平均成绩,向 course_avg 表中插入每门课程的课程号及其平均成绩。

```
INSERT INTO course_avg(cno,ave)
SELECT cno,AVG(grade)
FROM sc
GROUP BY cno;
```

3.4.2 修改数据

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。满足指定条件的元组可以是一个元组,也可以是多个元组。

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当中的哪些行。

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

例题 3.64　将 maths 课程的学分改为 4 学分。

```
UPDATE course
SET credit = 4
WHERE cname = 'maths';
```

2. 修改多个元组的值

例题 3.65　将所有男同学的年龄增加 2 岁。

```
UPDATE student
SET age = age + 2
WHERE sex = '男';
```

例题 3.66　将所有课程的学分减 1。

```
UPDATE course
SET credit = credit - 1;
```

3. 带子查询的修改

在 UPDATE 语句中可以嵌套子查询，用于构造修改的条件。

例题 3.67　将所有选修 maths 课程的学生成绩改为 0 分。

```
UPDATE sc
SET grade = 0
WHERE 'maths' = (SELECT cname
                 FROM course
                 WHERE course.cno = sc.cno);
```

注意：在修改表中的数据时，修改后的数据应满足定义表时设定的约束条件，否则系统就会给出错误提示信息。例如，如果将某个学生的年龄修改为 14 岁，就违反了表定义时对于“年龄”字段的检查约束。

3.4.3　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

例题 3.68 删除学号为 10172011 的学生记录。

```
DELETE  FROM  student
WHERE sno = '10172011';
```

2. 删除多个元组的值

例题 3.69 删除学号为 10172005 学生的选课记录。

```
DELETE FROM sc
WHERE sno = '10172005';
```

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。

例题 3.70 删除所有学生的选课记录。

```
DELETE FROM sc;
```

3. 带子查询的删除

在 DELETE 语句中同样可以嵌套子查询，用于构造删除的条件。

例题 3.71 删除王五同学的选课记录。

```
DELETE FROM sc
WHERE '王五' = ( SELECT sname
                 FROM  student
                 WHERE student.sno = sc.sno);
```

注意：在删除表中的数据时，应满足定义表时设定的约束条件，否则系统会给出错误提示信息。例如，如果想删除学生表(student)中的某一个学生记录，但这个学生在选课表(sc)中存在选课记录，此时删除学生记录的操作就会出错，因为外码关联的表中数据删除顺序是先删除从表中的数据，再删除主表中的数据。所以，正确的做法是先从选课表中将这个学生的选课记录删除，再从学生表中删除这个学生的记录。

视频

3.5 视图

视图是从一个或几个基本表(或视图)导出的表，它与基本表不同，是一个虚表。视图一经定义，就可以和基本表一样被查询、删除，也可以在一个视图之上再定义新的视图，但对视图的更新(增加、删除、修改)操作则有一定的限制。

视图的特点如下：

（1）视图是从现有的一个或多个表中提取出来的，可以屏蔽表中的某些信息。

（2）视图是一个虚表，对视图的操作实际上是对基本表的操作。

（3）数据库中只存放视图的定义，不存放视图对应的数据。这些数据仍存放在原来的基本表中，所以基本表中的数据发生变化，从视图中查询的数据也就随之改变了。

（4）视图可以简化用户查询操作，隐蔽表之间的连接。

3.5.1　定义视图

建立视图的一般语法格式如下：

```
CREATE VIEW <视图名>[(<列名>[,<列名>]…)]AS (子查询)
[WITH CHECK OPTION]
[WITH READ ONLY];
```

说明：

（1）视图中的列名序列要么全部指定，要么全部省略。当列名序列省略时，直接使用子查询 SELECT 子句里的各列名作为视图列名。

下列几种情况不能省略列名序列：

① 多表连接时选出了几个同名列作为视图的字段；

② 视图列名中有常数、聚集函数或列表达式；

③ 需要用更合适的新列名做视图列的列名。

（2）WITH CHECK OPTION 是可选项，该选项表示对所建视图进行 INSERT、UPDATE 和 DELETE 操作时，系统需检查该操作的数据是否满足子查询中 WHERE 子句里限定的条件，若不满足，则系统拒绝执行。

（3）WITH READ ONLY 是可选项，该选项保证在视图上不能进行任何 DML 操作。

例题 3.72　建立计算机系学生的视图，包括学号、姓名、性别和年龄。并要求进行插入和修改操作时仍要保证此视图中只有计算机系的学生。

```
CREATE VIEW cs_student
AS
SELECT sno,sname,sex,age
FROM student
WHERE dept = '计算机系'
WITH CHECK OPTION;
```

本例中，视图列名及顺序与 SELECT 子句中一样，所以视图名 cs_student 后的列名被省略。对所建视图进行插入、修改和删除操作时，系统自动检查该操作的数据是否满足计算机系学生的条件，若不满足，则系统拒绝执行。

例题 3.73　建立计算机系学生的只读视图，包括学号、姓名、性别和年龄。

```
CREATE VIEW cs_student_only
AS
SELECT sno,sname,sex,age
FROM student
```

```
WHERE dept = '计算机系'
WITH READ ONLY;
```

本例中,视图 cs_student_only 一旦建立,就不允许在视图上进行任何 DML 操作。

例题 3.74 建立计算机系选修 maths 课程的学生视图,包括学号、姓名和成绩。

```
CREATE VIEW cs_student_maths
AS
SELECT student.sno,sname,grade
FROM   student,course,sc
WHERE  student.sno = sc.sno AND course.cno = sc.cno
       AND dept = '计算机系' AND cname = 'maths';
```

本例中,视图 cs_student_maths 是从多张基本表中提取出来的,所以,不能对视图 cs_student_maths 进行插入、修改和删除操作。

视图不仅可以建立在一个或多个基本表上,也可以建立在一个或多个已经定义好的视图上,或建立在基本表与视图上。

例题 3.75 建立计算机系年龄大于 18 岁的学生视图,包括学号和姓名。

```
CREATE VIEW cs_student_age
AS
SELECT sno,sname
FROM cs_student
WHERE age > 18;
```

本例中,视图 cs_student_age 是从已经创建的视图 cs_student 中提取出来的。有时,还可以用带有聚集函数和 GROUP BY 子句的查询来定义视图。

例题 3.76 建立一个记录每个系别学生人数的视图,包括系别名称和学生人数。

```
CREATE VIEW dept_count(dept,num)
AS
SELECT dept,COUNT(sno)
FROM student
GROUP BY dept;
```

本例中,由于 AS 子句中 SELECT 语句的目标列学生人数是通过使用聚集函数得到的,所以 CREATE VIEW 中必须明确定义组成 dept_count 视图的各个属性列名,必须使用列别名来命名表达式 COUNT(sno)。

3.5.2 查询视图

视图是从一个或多个表中导出的虚表,具有表的基本特性。从用户角度来说,基于视图的数据查询与基于基本表的数据查询一样使用 SELECT 语句,查询视图的方法与查询基本表的方法一致,所以 DBMS 执行对视图的查询实际上是根据视图的定义转换成等价的对基本表的查询。

例题 3.77 在计算机系学生的视图中查找男同学的信息。

```
SELECT *
```

```
FROM cs_student
WHERE sex = '男';
```

DBMS 对某 SELECT 语句进行处理时，若发现被查询对象是视图，则 DBMS 将进行如下操作：

（1）从数据字典中取出视图的定义；

（2）把视图定义的子查询和本 SELECT 语句定义的查询相结合，生成等价的对基本表的查询（此过程称为视图的消解）；

（3）执行对基本表的查询，把查询结果（作为本次对视图的查询结果）向用户显示。

因此，本例转换后的查询语句为：

```
SELECT sno,sname,sex,age
FROM student
WHERE dept = '计算机系' AND sex = '男';
```

通常，对视图的查询是不会出现问题的。但有时视图消解过程不能给出语法正确的查询条件，这可能不是查询语句的语法错误，而是转换后的语法错误。此时，用户需要自己把对视图的查询转化为对基本表的查询。

3.5.3　操纵视图

操纵视图是指通过视图来插入、删除和修改数据。同查询视图一样，由于视图是不实际存储数据的虚表，因此对视图的操纵，最终要转换为对基本表的操纵。

此外，用户通过视图操纵数据不能保证被操纵的数据符合原来视图中定义的 AS <子查询>的条件。因此，在定义视图时，若加上子句 WITH CHECK OPTION，则在对视图操纵时，系统将自动检查先前定义时的条件是否满足。若不满足，则拒绝执行。

1. 向视图中插入数据

例题 3.78　建立信息系学生的视图，包括学号、姓名、性别和系别名称。向信息系学生的视图中插入一个新的学生记录，其中学号为 10172013，姓名为“小文”，性别为“女”，系别为“信息系”。

视图的建立：

```
CREATE VIEW is_student
AS
SELECT sno,sname,sex,dept
FROM student
WHERE dept = '信息系';
```

视图数据的插入：

```
INSERT INTO is_student
VALUES('10172013','小文','女','信息系');
```

上述语句在执行时，将转换成向学生表（student）中插入数据：

```
INSERT INTO student
```

```
VALUES('10172013','小文','女',NULL,'信息系');
```

2. 在视图中修改数据

例题 3.79 将信息系学生的视图中,学号为 10172013 的学生姓名改为“周文”。

```
UPDATE is_student
SET sname = '周文'
WHERE sno = '10172013';
```

上述语句在执行时,将转换成在学生表(student)中修改数据:

```
UPDATE student
SET  sname = '周文'
WHERE sno = '10172013'AND dept = '信息系';
```

3. 从视图中删除数据

例题 3.80 从信息系学生的视图中,删除学号为 10172013 的学生记录。

```
DELETE FROM is_student
WHERE sno = '10172013';
```

上述语句在执行时,将转换成从学生表(student)中删除数据:

```
DELETE FROM student
WHERE sno = '10172013' AND dept = '信息系'
```

并不是所有的视图操纵都能转换成有意义的对基本表的操纵。为了能够正确地执行视图操纵,各 DBMS 对视图操纵都有若干规定,由于各系统在实现方法上存在差异,这些规定也不尽相同。一般的规定如下:

(1) 通常对于由一个基本表导出的视图,如果是从基本表中去掉除码外的某些列和行,是允许操纵的。

(2) 若视图是由两个以上的基本表导出的,则此视图不允许操纵。

(3) 若视图的列是由聚集函数或计算列构成的,则此视图不允许操纵。

(4) 若视图定义中含有 DISTINCT、GROUP BY 等子句,则此视图不允许操纵。

3.5.4 删除视图

删除视图的一般语法格式如下:

```
DROP VIEW <视图名>;
```

注意:

(1) 删除视图后,视图的定义将从数据字典中删除,但基本表中的数据不受影响。

(2) 删除基本表后,由该基本表导出的所有视图并没有被删除,但均已无法使用。

例题 3.81 删除信息系学生的视图。

```
DROP VIEW is_student;
```

3.5.5　视图的优点

视图作为数据库中的一个重要的概念，有很多的优点，主要包括以下几个方面。

(1) 为用户集中数据，节省用户的查询和处理数据的时间。有时用户所需要的数据分散在多个表中，定义视图可将它们集中在一起，从而方便用户。

(2) 屏蔽数据库的复杂信息。用户不必了解复杂的数据库中表的结构，并且数据库中表的更改也不影响用户对数据库的使用。

(3) 简化用户管理的权限。只需授予用户使用视图的权限，不必指定用户只能使用表的某些特定列，增加了安全性。

(4) 便于数据共享。各用户不必都分别定义和存储自己所需的数据，可直接共享数据库的数据，同样的数据只需存储一次。

(5) 可以重新组织数据，以便输出到其他的应用程序中。

3.6　实验

3.6.1　实验 1　SQL * PLUS 常用命令练习

1. 实验目的

(1) 掌握 Oracle 客户端工具 SQL * PLUS 的交互运用。

(2) 熟悉 SQL * PLUS 中的常用命令。

2. 实验内容

(1) 以 system 用户身份登录 SQL * PLUS，登录后显示当前用户。

```
SHOW USER;
```

(2) 查看 system 用户下的表。

```
SELECT table_name FROM user_tables;
```

(3) 查看员工表 emp 的结构。

```
DESC emp;
```

(4) 查看 SQL * PLUS 里的命令。

```
HELP INDEX;
```

(5) 查看 RUN 命令的使用方法及简写形式。

```
? RUN;
```

(6) 设置行宽和列宽(设置前与设置后分别运行“SELECT * FROM emp;”看结果变化)。

```
SET LINESIZE 200;
```

```
SET PAGESIZE 200;
```

(7) 查找缓存区内最近写过的命令。

```
list;
```

(8) 执行缓存区里的命令。

```
/ , run ,r;
```

(9) 替换命令,将当前行中的 old 替换为 new。

```
CHANGE/old/new;
```

错误语句:

```
SELECT * FOM emp
CHANGE/FOM/FROM;
```

(10) 编辑命令,对当前的输入进行编辑(Windows 默认在记事本中编辑)。

```
EDIT;
```

错误语句:

```
SELECT table_name FROM user_table;
```

输入 ED 回车进行编辑:

```
SELECT table_name FROM user_tables;
```

保存修改后,输入/回车即可。

(11) 保存最近写过的命令。

```
SAVE c:\part1; (默认保存成.sql)
SAVE c:\part1.txt;
```

(12) 读入命令。

```
GET c:\part1.sql;
```

(13) 读入并执行。

```
START c:\part1.sql;
```

(14) 保存所有的操作。

```
SPOOL c:\part2.sql;(先创建文本,从想保存的位置开始)
SELECT * FROM emp;(写入想保存的命令,包括结果)
SPOOL OFF;(操作结束的位置)
```

3. 考核标准

本实验为选做实验,根据课时进度安排,既可以在课堂上完成,也可作为学生课外作业独立完成。要求学生在自己的计算机上成功安装 Oracle,并且能够熟练使用 SQL * PLUS 常用命令进行练习即为优秀;如果出现错误,根据错误情况灵活给分。

3.6.2　实验 2　数据定义语言

1. 实验目的

(1) 掌握基本表的创建方法。

(2) 掌握基本表结构的修改方法。

(3) 掌握基本表删除的方法。

2. 实验内容

(1) 按要求采用不同的约束类型创建科室表和医生表。

① 科室表：dept(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,如表 3.12 所示。

表 3.12　科室表结构

列　名	数 据 类 型	长度	完整性约束
deptno	CHAR	10	主码
dname	VARCHAR	15	唯一
loc	VARCHAR	20	无

② 医生表：doctor(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,如表 3.13 所示。

表 3.13　医生表结构

列　名	数 据 类 型	长度	完整性约束
docno	CHAR	10	主码
docname	VARCHAR	15	非空
age	INT	无	年龄为 18～60 岁
sal	NUMBER	无	无
deptno	CHAR	10	外码(参照 dept 表中 deptno)

(2) 按要求对表的结构进行修改。

① 对表增加一列。

在医生表中增加一个属性列：birthday(生日),数据类型是 DATE。

② 改变列的类型。

将科室表中 dname 属性列的类型改为 VARCHAR2(20)。

③ 增加约束条件。

在医生表中添加一个名为 CHK_SAL 的约束,从而保证医生工资的取值总是为 1000～8000,即 sal BETWEEN 1000 AND 8000。

④ 删除原有的列。

删除医生表中的 birthday 属性列。

(3) 按要求删除基本表。

删除医生表(doctor)。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误个数以及难易程度灵活给分。

3.6.3 实验3 数据操纵语言

1. 实验目的

(1) 掌握基本表中数据的插入操作。

(2) 掌握基本表中数据的修改操作。

(3) 掌握基本表中数据的删除操作。

2. 实验内容

(1) 创建教师信息基本表。

教师信息表:teacher(tno,tname,sex,sal,tdept),表中属性列依次是教师编号、教师姓名、性别、工资和系别名称,如表3.14所示。

表3.14 教师信息表

列 名	数据类型	长度	完整性约束
tno	CHAR	8	主码
tname	VARCHAR2	20	非空
tsex	VARCHAR2	6	无
tsal	NUMBER	无	工资大于1800
tdept	CHAR	20	无

(2) 练习向基本表中插入数据、修改数据和删除数据。

① 向教师表中插入如表3.15所示数据。

表3.15 向教师表中插入的数据

tno	tname	tsex	tsal	tdept
T001	张老师	女	3000	计算机系
T002	王老师	男	2800	计算机系
T003	李老师			信息系
T004	张老师	男	3500	信息系
T005	刘老师	女	2200	管理系

② 将“信息系”更名为“网络工程系”。

③ 将王老师的工资更改为3300。

④ 删除教师表中所有计算机系的教师信息。

⑤ 删除教师表中的全部数据。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。程序语句无语法错误、书写规范、运行结果正确为优秀；如果出现错误，根据错误个数以及难易程度灵活给分。

3.6.4　实验 4　单表查询

1. 实验目的

(1) 掌握对单个基本表数据进行查询的方法。

(2) 掌握聚集函数的应用。

2. 实验内容

(1) 创建员工信息表。

员工信息表：employees(eno，ename，sex，age，job，sal，dept)，表中属性列依次是员工编号、员工姓名、性别、年龄、工作岗位、工资和部门名称，如表 3.16 所示。

表 3.16　员工信息表

列　　名	数 据 类 型	长度	完整性约束
eno	CHAR	8	主码
ename	VARCHAR2	10	非空
sex	CHAR	6	取值只允许为男或女
age	INT	无	年龄大于 18 岁
job	VARCHAR2	20	无
sal	NUMBER	无	无
dept	VARCHAR2	20	无

(2) 向已创建的员工表中插入如表 3.17 所示的数据。

表 3.17　向员工表中插入的数据

eno	ename	sex	age	job	sal	dept
1001	张三	男	20	销售	1000	市场部
1002	李四	女	26	会计	1600	财务部
1003	王五	女	22	销售	1000	市场部
1004	赵六	男	19			
1005	张七	女	23	测试	1400	技术部
1006	赵八	男	30	研发	2000	技术部

(3) 按要求完成各种单表信息查询，并验证聚集函数的功能。

① 查询所有员工的姓名、性别和工资。

② 查询员工表中所有的部门名称(要求去掉重复的值)。

③ 查询技术部员工的姓名和出生年份。
④ 查询工资超过 1200 元的员工的姓名和年龄。
⑤ 查询年龄不为 20～25 岁的员工的姓名和工资。
⑥ 查询财务部、技术部的员工的姓名和性别。
⑦ 查询所有姓张的员工的姓名、年龄和工作。
⑧ 查询工作岗位为空的员工姓名和年龄。
⑨ 查询市场部里年龄小于 25 岁的男员工的姓名。
⑩ 查询年龄超过 20 岁的员工的姓名和工资,查询结果按照工资的降序排列。
⑪ 查询市场部的员工人数。
⑫ 查询公司中员工的最高工资。
⑬ 查询公司中员工的最低工资。
⑭ 查询技术部门员工的平均年龄。
⑮ 查询市场部中员工的工资总额。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误个数以及难易程度灵活给分。

3.6.5 实验 5 多表连接查询和集合查询

1. 实验目的

(1) 掌握多表连接的查询方法。
(2) 了解连接查询中的左外连接和右外连接。
(3) 熟悉集合查询的应用。

2. 实验内容

样本数据库中,学生表(student)、课程表(course)和选课表(sc)的数据信息分别如表 3.18～表 3.20 所示。

表 3.18 学生表数据

sno	sname	sex	age	dept
10172001	陈一	男	17	计算机系
10172002	姚二	女	20	计篡机系
10172003	张三	女	19	计算机系
10172004	李四	男	22	日语系
10172005	王五	男	22	日语系
10172006	赵六	男	19	日语系
10172007	陈七	女	23	信息系
10172008	刘八	男	21	信息系
10172009	张九	女	18	管理系
10172010	孙十	女	21	管理系

表 3.19　选课表数据

cno	cname	tname	cpno	credit
c1	maths	曹老师		3
c2	english	赵老师		5
c3	japanese	刘老师		4
c4	database	杨老师	c1	3
c5	java	陈老师	c1	3
c6	jsp_design	陈老师	c5	2

表 3.20　选课表数据

sno	cno	grade
10172001	c1	94
10172001	c2	96
10172001	c4	92
10172002	c1	50
10172002	c2	88
10172002	c4	76
10172002	c5	55
10172003	c1	65
10172003	c2	72
10172003	c3	90
10172003	c4	85
10172003	c5	93
10172003	c6	86
10172004	c2	50
10172004	c3	45
10172005	c2	88
10172005	c3	85
10172007	c1	80
10172007	c2	73
10172007	c3	66
10172007	c4	
10172007	c5	
10172008	c1	82
10172008	c2	77
10172008	c3	85
10172008	c4	87
10172008	c5	82
10172008	c6	94
10172009	c1	86
10172009	c2	
10172010	c1	73
10172010	c2	

(1) 根据样本数据库中的表和数据,进行多表连接查询操作的练习。

① 查询选修了 c4 课程的学生的姓名及其成绩,查询结果按成绩降序排列。

② 求男同学的总人数和平均年龄。

③ 查询每名学生的学号、选课门数和平均成绩。

④ 查询平均成绩大于 80 分的学生学号以及平均成绩。

⑤ 查询选修了 java 课程的学生学号及成绩。

⑥ 查询计算机系学生的选课情况,要求列出学生的名字、所选课程的名称和成绩。

(2) 根据样本数据库中的表和数据,进行集合查询操作的练习。

① 查询计算机系和日语系学生的基本信息(集合并运算)。

② 查询信息系中选修 c4 课程的学生学号(集合交运算)。

③ 查询管理系的学生与年龄不大于 20 岁的学生的差集(集合差运算)。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误个数以及难易程度灵活给分。

3.6.6 实验 6 嵌套查询

1. 实验目的

(1) 掌握不相关子查询的查询方法。

(2) 掌握相关子查询的查询方法。

(3) 理解不相关子查询与相关子查询的区别。

2. 实验内容

样本数据库中,学生表(student)、课程表(course)和选课表(sc)的数据信息分别如表 3.21～表 3.23 所示。

表 3.21 学生表数据

sno	sname	sex	age	dept
10172001	陈一	男	17	计算机系
10172002	姚二	女	20	计篡机系
10172003	张三	女	19	计算机系
10172004	李四	男	22	日语系
10172005	王五	男	22	日语系
10172006	赵六	男	19	日语系
10172007	陈七	女	23	信息系
10172008	刘八	男	21	信息系
10172009	张九	女	18	管理系
10172010	孙十	女	21	管理系

表 3.22　课程表数据

cno	cname	tname	cpno	credit
c1	maths	曹老师		3
c2	english	赵老师		5
c3	japanese	刘老师		4
c4	database	杨老师	c1	3
c5	java	陈老师	c1	3
c6	jsp_design	陈老师	c5	2

表 3.23　选课表数据

sno	cno	grade
10172001	c1	94
10172001	c2	96
10172001	c4	92
10172002	c1	50
10172002	c2	88
10172002	c4	76
10172002	c5	55
10172003	c1	65
10172003	c2	72
10172003	c3	90
10172003	c4	85
10172003	c5	93
10172003	c6	86
10172004	c2	50
10172004	c3	45
10172005	c2	88
10172005	c3	85
10172007	c1	80
10172007	c2	73
10172007	c3	66
10172007	c4	
10172007	c5	
10172008	c1	82
10172008	c2	77
10172008	c3	85
10172008	c4	87
10172008	c5	82
10172008	c6	94
10172009	c1	86
10172009	c2	
10172010	c1	73
10172010	c2	

(1) 根据样本数据库中的表和数据,进行不相关子查询的练习。

① 查询与王五同一个系别的学生姓名和年龄。

② 查询选修了 jsp_design 课程的学生学号和姓名。

③ 查询比所有计算机系学生年龄都大的学生的基本情况。

④ 查询平均成绩最高的学生学号。

⑤ 查询李四同学不学课程的课程号。

(2) 根据样本数据库中的表和数据,进行相关子查询的练习。

① 在选课信息表中查询选修 japanese 课程的学生学号和成绩。

② 查询没有选修 c2 课程的学生学号和姓名。

③ 查询所有课程成绩均大于 70 分的学生姓名。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。程序语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误点数以及难易程度灵活给分。

3.6.7 实验 7 视图

1. 实验目的

(1) 掌握视图的建立方法。

(2) 掌握视图的查询方法。

(3) 掌握视图的删除方法。

2. 实验内容

样本数据库中,学生表(student)的数据信息如表 3.24 所示。

表 3.24 学生表数据

sno	sname	sex	age	dept
10172001	陈一	男	17	计算机系
10172002	姚二	女	20	计篡机系
10172003	张三	女	19	计算机系
10172004	李四	男	22	日语系
10172005	王五	男	22	日语系
10172006	赵六	男	19	日语系
10172007	陈七	女	23	信息系
10172008	刘八	男	21	信息系
10172009	张九	女	18	管理系
10172010	孙十	女	21	管理系

根据样本数据库中学生表的结构和数据,进行视图操作的练习。

(1) 建立“日语系”学生的视图 jp_student,包括学号、姓名和年龄。

（2）查询视图 jp_student 中年龄大于 20 岁的学生姓名。

（3）将视图 jp_student 中学号为 10172004 同学的姓名改为李子。

（4）删除视图 jp_student 中姓名为“赵六”的学生信息。

（5）删除视图 jp_student。

3. 考核标准

本实验为必做实验，要求学生在课堂上独立完成。根据题目要求，按照实验步骤完成相应实验内容。程序语句无语法错误、书写规范、运行结果正确为优秀；如果出现错误，根据错误个数以及难易程度灵活给分。

3.7　本章小结

本章首先介绍了 SQL 的产生、发展和特点。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。

其次，介绍了 SQL 的数据定义功能、数据查询功能和数据操纵功能。数据定义功能包括基本表、索引、视图的创建、修改和删除。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集函数完成相关计算，也可以使用分组子句将查询结果按某一属性列的值分组，还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。数据操纵功能包括数据插入、数据修改和数据删除。

最后，介绍了视图，视图是为了确保数据表的安全性和隐蔽性从一个或多个表中或其他视图中使用 SELECT 语句导出的虚表。数据库中仅存放视图的定义，而不存放视图所对应的数据，数据仍然存放在基本表中，对视图中数据的操纵实际上仍是对组成视图的基本表数据的操纵。

3.8　课后习题

1. 在关系数据库中，SQL 是指（　　）。

   A. Selected Query Language　　B. Procedured Query Language

   C. Standard Query Language　　D. Structured Query Language

2. SQL 的运算对象和结果都是（　　）。

   A. 数据　　B. 属性　　C. 关系　　D. 数据项

3. 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. 在给表命名时，第一个字符不能是数字

   C. 表名和属性列的名字区分大小写

   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

4. 下列（　　）操作符号可以和 NULL 值进行比较。

   A. IS　　B. =　　C. LIKE　　D. <>

5. 涉及四张表的查询时,WHERE 子句中至少有(　　)个条件表达式。

A. 1　　B. 2　　C. 3　　D. 4

6. 自然连接是关系数据库中重要的关系运算,下列说法正确的是(　　)。

A. 自然连接就是连接,只是说法不同罢了

B. 自然连接其实是等值连接,它与连接不同

C. 自然连接是去掉重复属性的等值连接

D. 自然连接是去掉重复元组的等值连接

7. 在关系数据库中,实现表与表之间的联系是通过(　　)。

A. 实体完整性规则　　B. 参照完整性规则

C. 用户自定义的完整性　　D. 属性的值域

8. 在 SQL 语句中,HAVING 子句用于筛选满足条件的(　　)。

A. 行　　B. 列　　C. 元组　　D. 分组

9. 下列不属于不相关子查询特点的是(　　)。

A. 可以运行多次

B. 能独立运行,子查询条件不依赖父查询

C. 只能运行一次

D. 先执行子查询,后执行父查询

10. 关于视图,以下说法不正确的是(　　)。

A. 视图是虚表

B. 数据库中只存放视图的定义,不存放视图对应的数据

C. 使用视图可以简化用户的数据查询和处理

D. 使用视图可以加快查询语句的执行速度

11. 简述建立索引的目的,并分析是否索引建立得越多越好。

12. 简述在 SELECT 语句中,HAVING 子句与 WHERE 子句的区别。

13. 简述基本表和视图的含义,并分析两者之间的区别和联系。

14. 某数据库中包含图书信息、读者信息和借阅信息三张基本表。

图书信息表:book(bno,bname,author,price),表中属性列依次是图书编号、图书名称、图书作者和图书价格。

读者信息表:reader(rno,rname,address),表中属性列依次是借书证号、读者姓名和读者地址。

借阅信息表:br(bno,rno,datetime),表中属性列依次是图书编号、借书证号和借书日期。

图书信息表(book)结构如表 3.25 所示。

表 3.25　图书信息表结构

列　名	数据类型	长度	完整性约束
bno	CHAR	10	主码
bname	VARCHAR	20	非空
author	VARCHAR	20	无
price	NUMBER	无	大于 0

读者信息表(reader)结构如表 3.26 所示。

表 3.26　读者信息表结构

列　　名	数 据 类 型	长度	完整性约束
rno	CHAR	10	主码
rname	VARCHAR	20	非空
address	VARCHAR	50	无

借阅信息表(br)结构如表 3.27 所示。

表 3.27　借阅信息表

列　　名	数 据 类 型	长度	完整性约束
bno	CHAR	10	外码(参照 book 表中 bno)
rno	CHAR	10	外码(参照 reader 表中 rno)
datetime	DATE	无	无

主码为(bno,rno)。

(1) 用 SQL 语句实现以下基本表的创建。

① 图书信息表的创建。

② 读者信息表的创建。

③ 借阅信息表的创建。

(2) 根据各表结构,用 SQL 语句完成下列操作。

① 将图书表中图书名称属性列的数据类型改为 varchar(30)。

② 向读者表中增加年龄属性列 age,数据类型为 number。

③ 将图书编号为 b1 的图书价格改为 24。

④ 删除“张三”的借阅信息。

⑤ 查询图书价格超过 100 元的图书数量。

⑥ 查询书名中含有“数据库”的图书编号和图书价格。

⑦ 查询借阅了《时间简史》的读者姓名和借阅时间。

⑧ 查询借阅了 5 本以上图书的读者姓名。

⑨ 查询一本图书都没有借阅的借书证号和读者姓名。

⑩ 查询图书表中最贵的图书名称和价格。

15. 某数据库中包含供应商信息、零件信息、项目信息和供应情况信息四张基本表。

供应商信息表：s(sno,sn,city)，表中属性列依次是供应商编号、供应商名和供应商所在城市。

零件信息表：p(pno,pn,color,weight)，表中属性列依次是零件编号、零件名、颜色和重量。

项目信息表：j(jno,jn,city)，表中属性列依次是项目编号、项目名称和项目所在城市。

供应情况信息表：spj(sno,pno,jno,qty)，表中属性列依次是供应商编号、零件编号、项目编号和供应数量。

供应商信息表(s)结构如表 3.28 所示。

表 3.28 供应商信息表结构

列 名	数 据 类 型	长度	完整性约束
sno	CHAR	10	主码
sn	VARCHAR	20	非空
city	VARCHAR	20	无

零件信息表(p)结构如表 3.29 所示。

表 3.29 零件信息表结构

列 名	数 据 类 型	长度	完整性约束
pno	CHAR	10	主码
pn	VARCHAR	20	非空
color	VARCHAR	20	无
weight	NUMBER	无	大于 0

项目信息表(j)结构如表 3.30 所示。

表 3.30 项目信息表结构

列 名	数 据 类 型	长度	完整性约束
jno	CHAR	10	主码
jn	VARCHAR	20	非空
city	VARCHAR	20	无

供应情况信息表(spj)结构如表 3.31 所示。

表 3.31 供应情况信息表结构

列 名	数 据 类 型	长度	完整性约束
sno	CHAR	10	外码(参照 s 表中 sno)
pno	CHAR	10	外码(参照 p 表中 pno)
jno	CHAR	10	外码(参照 j 表中 jno)
qty	INT	无	无

主码为(sno,pno,jno)。

(1) 用 SQL 语句实现以下基本表的创建。

① 供应商信息表的创建。

② 零件信息表的创建。

③ 项目信息表的创建。

④ 供应情况信息表的创建。

(2) 根据各表结构,用 SQL 语句完成下列操作。

① 查询所有零件的名称、颜色和重量。

② 查询所在城市为“大连”的所有项目的详细信息。

③ 查询重量最轻的零件名称。

④ 查询为项目编号 j1 提供零件的供应商名称。

⑤ 查询由供应商编号 s1 提供的零件颜色。

⑥ 查询项目编号 j2 使用的各种零件名称及数量。

⑦ 查询供应绿色的零件编号为 p2 且供应数量超过 500 的供应商名称。

⑧ 查询每个城市的城市名称及供应商数量，查询结果按照数量的降序排列。

⑨ 查询没有使用“大连”生产的零件的项目编号。

⑩ 查询为编号 j1 和 j2 的项目提供零件的供应商编号。