第5章 关系数据库语言SQL

5.1 内容概述

SQL语言按各语句完成的功能分为数据定义语句、数据操纵语句和数据控制语句三大类。数据定义语句用于定义数据库的逻辑结构，包括定义基本表、定义视图和定义索引等。数据库操纵语句包括数据查询语句和数据更新语句。数据查询语句能够按照某种要求从数据库中检索出需要的数据，并对其进行统计、分组、排序等；数据更新语句能够实现数据的插入、删除、修改等数据维护操作。数据控制语句实现用户授权、基本表和视图授权、事务控制、完整性和安全性控制等。

在SQL语言中，一个关系即为一个表，一个数据库应用系统中的所有表的集合构成了该数据库应用系统的数据库，一个单独的表不能称为数据库。

一个数据库应用系统主要由存储数据的数据库和对数据库中的数据进行操作和管理的应用程序系统组成。建立数据库过程中的重要步骤就是用CREATE TABLE语句定义表。该语句描述了被定义表的各种相关信息，包括表名，组成该表的属性名，各属性的数据类型、完整性约束，表的完整性约束等。当某个已创建的表命名出错时，用RENAME语句修改表名；当需要增加、删除或修改表的属性时用ALTER TABLE语句，表的撤销用DROP TABLE语句。

当建好一个表后，利用INSERT语句插入数据（在数据库应用系统中称为数据录入），当插入的数据出错时，可利用UPDATE语句进行修改，或利用DELETE语句删除有错误的数据。

数据查询是数据库操作的核心。SQL语言利用SELECT语句进行数据库查询操作，该语句包括SELECT、FROM、WHERE、GROUP BY、HAVING和ORDER BY 6个子句，每个子句随可选项的不同可实现不同的功能。通过各子句、聚合函数和谓词的适当组合与灵活使用，不仅可以实现简单查询，而且可以实现多表连接查询、嵌套查询、谓词演算查询等高级查询功能。

数据更新（数据插入、修改和删除）语句还可以与数据查询语句结合起来应用，从而使数据更新操作变得非常灵活、方便。

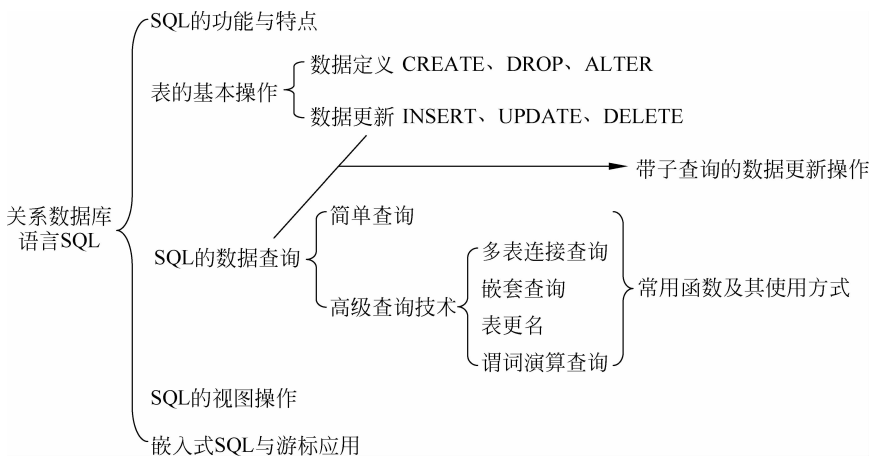
视图是关系数据库系统提供给用户，并能以多种角度观察数据库中数据的重要机制。视图是从一个或几个基本表（或视图）通过视图定义语句所描述的映射关系导出的表，视图

不是数据库中真正存在的表，所以称为虚表。和基本表一样，视图一经定义，就可以对其进行查询，或在其上再定义新的视图，但在视图上进行数据更新有较严格的限制。由于视图定义语句描述了表（或视图）与视图的映射关系，所以当数据库的基本表中的内容（数据）发生变化时，从视图中查询出来的数据当然也就变化了。

SQL 语言语句作为单独的命令在交互方式下使用，是非过程性的，大多数语句的执行都是独立的，与上下文无关，这无法满足绝大多数应用所需的过程性要求，为此引入了嵌入式 SQL。嵌入式 SQL 是指把 SQL 语言语句嵌入到某种高级语言中的应用形式。嵌入有 SQL 语句的高级语言称为宿主语言，简称主语言。含有嵌入 SQL 语句的高级语言应用程序称为宿主应用程序，简称为应用程序。在应用程序中，对数据库的各种操作利用 SQL 语句实现，对数据的各种处理由主语言语句实现。

5.2　知识体系

本章的知识体系结构如下所示：



本章的学习要点包括：

（1）了解 SQL 的功能与特点。

（2）掌握表的定义、修改与撤销的 SQL 语句，掌握数据的插入、修改、删除的 SQL 语句。

（3）熟练掌握 SQL 的数据查询等基本功能，较熟练地掌握 SQL 的高级查询技术。

（4）掌握带有子查询的数据更新操作的 SQL 语句，理解视图的概念和定义方法。

（5）了解嵌入式 SQL 的概念，理解游标的概念及其应用方法。

重点：

（1）SQL 数据定义语句、数据更新语句的应用。

（2）SQL 查询语句及其应用。

难点：

（1）嵌套查询、谓词演算查询等高级查询技术的应用。

（2）嵌入式 SQL 中游标的应用。

5.3 基本知识点

1. 结构化查询语言 SQL

SQL(Structured Query Language,结构化查询语言)是一种介于关系代数和元组演算之间的关系数据库语言,在 1974 年由 Boyce 和 Chamberlin 提出,1975 年至 1979 年由 IBM 公司在 System R 上实现,1986 年由美国国家标准局(American National Standard Institute,ANSI)批准为关系数据库语言的国家标准,1987 年由国际标准化组织(International Standard Organization,ISO)批准为国际标准,1999 年国际标准化组织已公布了最新的 SQL 标准 SQL-99,即 SQL-3。SQL 在世界上绝大多数关系数据库中的采用极大地推进了数据库技术的发展和广泛应用,并已在数据库之外的其他领域的软件产品中得到应用。

2. SQL 的工作方式

SQL 语言提供了交互式命令(在 SQL 的交互式工作方式中,每一个 SQL 语句又可看作是一条 SQL 命令,所以经常称 SQL 语句为 SQL 命令)方式和嵌入式两种工作方式。在交互式命令工作方式下,用户可以以交互式命令方式通过直接输入 SQL 命令(语句)对数据库进行操作。例如在 SQL Server 2005 中,用户可以在查询编辑器窗口直接输入 SQL 命令(语句)对数据库进行操作;在嵌入式工作方式下,SQL 语句可以被嵌入到某种高级语言(例如 VB.NET、VC、Java 等)程序中实现对数据库的操作,并利用主语言(所嵌入的高级语言称为宿主语言,简称主语言)强大的计算功能、逻辑判断功能、屏幕控制及输出功能等实现对数据的处理和输入/输出控制等。

3. 基本表的定义

1) 定义语句

基本表的定义是由 CREATE TABLE 语句实现的,其格式如下:

```
CREATE TABLE <表名>
      (<列名 1><数据类型>[<列 1 的完整性约束>]
        [,<列名 2><数据类型>[<列 2 的完整性约束>],
            …
          <列名 n><数据类型>[<列 n 的完整性约束>],
          [<表的完整性约束>]]);
```

2) 列的完整性约束

列的完整性约束用于给出该列数据的完整性约束条件,由用户根据各属性列的数据特点和要求确定。它是一个可选项,表示该项内容可选,也可以不选。当不选该选项(即该项空缺)时,默认为 NULL,表示该列可为空值;当选该选项(即该项不空缺)时,该选项列的完整性约束条件为下列选项中的一个:

```
[NULL|NOT NULL|PRIMARY KEY|DEFAULT|CHECK|UNIQUE|NOT NULL UNIQUE]
```

(1) NULL。

NULL用于指出该列可以为空值,其含义是指该列的值还没确定,NULL不分类型。NULL不同于数值型列的0,也不同于字符型列的空格。零和空格都是一个具体的值,而NULL是空值,通常情况下不占存储空间。

(2) NOT NULL。

NOT NULL用于指出该列不能为空值,即该表中的所有元组在该列必须有确定的值。每一个表中至少应有一个列的可选项为NOT NULL。

(3) PRIMARY KEY。

SQL语言规定,主键列总具有唯一且非空的值,也就是说,定义为主键的列已经隐含具有NOT NULL选项和UNIQUE选项,所以在主键列中可以省略NOT NULL选项。

(4) DEFAULT。

DEFAULT用于给所在的列设置一个默认值,即在插入一个新记录时,如果带有DEFAULT选项的列没有新值,就将默认值插入该列。DEFAULT约束的格式如下:

```
DEFAULT(<缺省值>)
```

(5) CHECK。

CHECK用于指出所在列的值只能取<值的约束条件>所规定的集合之内的值,其格式如下:

```
CHECK(<值的约束条件>)
```

其中,<值的约束条件>一般是一个条件表达式。CHECK的作用是,当向CHECK所在的列插入一个新值时,系统先检查插入的值是否符合值的约束条件,如果符合就将新值放入该列中,如果不符合就拒绝接受插入的新值。

(6) UNIQUE

UNIQUE用于指出所在列的值对于所在的表来说是唯一的,即该列的值不允许相同。

3) 表的完整性约束

表的完整性约束用于给出所在表的数据的约束条件,由用户根据表中各属性列数据的特点和要求确定。表约束子句放在表中最后一列的后面。表的完整性约束条件主要有以下几个。

(1) 表的主键约束子句,格式如下:

```
PRIMARY KEY(<主键列名 1>[,<主键列名 2>,…,<主键列名 r>])
```

当表的主键由多个列名组合而成时,必须将表的主键定义成表约束格式。当然,当表中只有一个列是主键时,也可以将表的主键定义成列约束格式。

(2) 表的外键约束子句,格式如下:

```
FOREIGN KEY(<列名 1>) REFERENCES <表名>(<列名 2>)
```

本子句定义了一个列名为"<列名1>"的外键,它与表"<表名>"中的"<列名2>"相对应,且"<列名2>"在表"<表名>"中是主键。

（3）表检验约束 CHECK 子句。

表检验约束 CHECK 子句的含义和格式与列检验约束相同，所不同的是，表检验约束 CHECK 子句是一个独立的子句而不是子句中的一部分。表检验约束 CHECK 子句中的＜值的约束条件＞不仅可以是一个条件表达式，而且可以是一个包含 SELECT 的 SQL 语句。

4. 数据的插入

当一个新表建立后，就需要给它插入（输入）数据。向表中插入一行（单元组）数据的 INSERT 语句格式如下：

```
INSERT INTO <表名> [(<列名表>)]
VALUES(<值表>);
```

带有子查询的数据插入操作实际上是数据的导入，其操作的实质是把从某个或某些表中查询出来的数据插入到另一个表中。其语句格式如下：

```
INSERT INTO <表名> [(<列名表>)]
<子查询>;
```

5. 数据的修改

在 SQL 语言中，修改表中的数据是由 UPDATE 语句实现的，其语句格式如下：

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

6. 数据的删除

在 SQL 语言中，对数据的删除是用 DELETE 语句实现的，其语句格式如下：

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

7. 数据的查询

一个比较完整的 SQL 查询语句的格式如下：

```
SELECT <列名表或列表达式序列>
FROM <表名表>
[WHERE <条件>]
[GROUP BY <列名表>]
[HAVING <分组条件>]]
[ORDER BY <列名> [ASC/DESC][, <列名> [,ASC/DESC]] … ];
```

整个语句的含义是根据 WHERE 子句的条件表达式从 FROM 子句指定的基本表或视图中找到满足条件的元组；然后按 GROUP BY 子句指定的列的值进行分组，该属性列值相等的元组为一组；再以这些组为基础，提取满足 HAVING 子句指定条件的组；最后按 SELECT 子句中给出的列名或列表达式求值输出。ORDER BY 子句对输出的结果按给定

列名的值进行排序。其中，只有前两个子句是必需的，其他子句可以根据查询和结果显示要求选择或省略。

8. 视图

视图由数据库中满足一定条件约束的数据组成，它可以由某个或某些表中满足一定条件的行组成，还可以由若干个表经过一定的运算形成。视图可以定义在表上，也可以定义在其他的视图上。视图一旦定义，就可以把它看作表一样在其上进行查询操作，只不过与基本表相比，视图是一种“虚表”。顾名思义，虚表是实际的 SQL 数据库中不存在的表。

视图的定义由 CREATE VIEW 语句实现，其语句格式如下：

```
CREATE VIEW <视图名> [(<视图列名表>)]
AS < SELECT 语句>
[WITH READ ONLY | WITH CHECK OPTION];
```

9. 嵌入式 SQL 技术

SQL 语句可以在各商用数据库管理系统的终端交互环境下独立运行，并分别实现各自的功能。但这种运行方式无法实现各语句操作结果信息的相互关联、传递和对查询结果的数据处理功能，也无法提供一般用户所需要的操作界面和某些应用需求。将 SQL 语言嵌入到高级程序设计语言中，不仅可以发挥 SQL 语言的非过程特点，而且可以利用高级语言强大的数据处理能力对查询结果进行所需的数据处理，利用高级语言的输入/输出和图形图像处理能力为用户设计友好的操作界面，从而推进数据库的广泛应用。

随着查询条件和数据库中数据量的不同，一次查询的结果可能没有满足条件的记录(零个记录)，可能只有一条记录，可能有几十、几百条记录，也可能有几千、几万条记录。这种查询结果记录数的不确定性和可能大的记录数量显然用一般高级语言中的数组或其他变量机制是无法实现的，所以引入了 SQL 游标的概念。SQL 语言中的游标用于实现多记录的查询，并通过与主语言语句的有效配合实现多记录的输出和显示等。

5.4　重点与难点分析

本章的重点是 SQL 查询语句及其应用，难点是嵌套查询、谓词的灵活应用、嵌入式 SQL。

5.4.1　SQL 查询语句及其应用

SQL 语句具有灵活的使用方式和功能强大、复杂的查询内涵。如何灵活地利用 SELECT 语句的不同形式的查询条件实现各种复杂的查询要求是掌握 SQL 的关键。

首先要明确 SQL 查询语句中各部分的功能作用：SELECT 子句对应关系代数中的投影运算，用来列出查询结果中的属性；FROM 子句对应关系代数中的笛卡儿积运算，用来列出需扫描的关系；WHERE 子句对应关系代数中的选择运算，用来给出作用于 FROM 子句中关系属性上的条件表达式；GROUP BY 子句对满足 WHERE 条件的元组集按照指定

的列的值进行分组，该属性列值相等的元组为一组；HAVING 子句指定 GROUP BY 子句形成的分组所应满足的条件；ORDER BY 子句依据给定列的值控制查询结果中元组的排列顺序，合理地使用这些子句可实现不同的查询要求。

其次，要掌握 SQL 查询语句中各部分的选择形式，尤其是“列名表或列表达式序列”和“条件表达式”的多种多样的选择形式，归纳如下：

(1) SELECT 子句的＜列名表或列表达式序列＞有以下可选格式：

① *

② ＜表名＞. *

③ 聚合函数：

$$\begin{Bmatrix}\text{COUNT}\\ \text{SUM}\\ \text{AVG}\\ \text{MAX}\\ \text{MIN}\end{Bmatrix}\ (\text{[DISTINCT|ALL]}\langle\text{列名}\rangle)$$

④ 表达式：由算术运算符号(＋、－、×、/)、连接的属性列名、作用于属性列的聚合函数和常量组成的算术表达式。

(2) WHERE 子句的条件表达式有以下可选格式：

① $\langle\text{列名}\rangle\ \theta\begin{Bmatrix}\langle\text{列名}\rangle\\ \langle\text{常量}\rangle\\ \text{[ANY/ALL]}(\text{SELECT 语句})\end{Bmatrix}$

② $\langle\text{列名}\rangle\ \text{[NOT]BETWEEN}\begin{Bmatrix}\langle\text{列名}\rangle\\ \langle\text{常量}\rangle\\ (\text{SELECT 语句})\end{Bmatrix}\ \text{AND}\begin{Bmatrix}\langle\text{列名}\rangle\\ \langle\text{常量}\rangle\\ (\text{SELECT 语句})\end{Bmatrix}$

③ $\langle\text{列名}\rangle\ \text{[NOT]IN}\begin{Bmatrix}(\langle\text{值 1}\rangle\text{[,}\langle\text{值 2}\rangle\cdots\text{]})\\ (\text{SELECT 语句})\end{Bmatrix}$

④ ＜列名＞ [NOT]LIKE ＜匹配串＞

⑤ ＜列名＞ IS[NOT] NULL

⑥ [NOT] EXISTS (SELECT 语句)

⑦ $\langle\text{条件表达式}\rangle\begin{Bmatrix}\text{AND}\\ \text{OR}\end{Bmatrix}\ \langle\text{条件表达式}\rangle\ \left(\begin{Bmatrix}\text{AND}\\ \text{OR}\end{Bmatrix}\langle\text{条件表达式}\rangle\right)\cdots$

特别强调的是，WHERE 子句的选择形式多数适用于 HAVING 子句，但由于 HAVING 子句是用来指定分组所应满足的条件的，所以其条件必须是描述分组的属性，因此在 HAVING 子句中可使用聚合函数，但在 WHERE 子句中不能使用聚合函数。

在掌握了 SQL 查询语句的基本知识后，关键问题是依据需求书写出正确的查询语句。如图 5.1 所示，用户可从查询条件和查询结果入手进行分析，例如应使用哪些子句？再依照每个子句的选择形式确定实现方法。

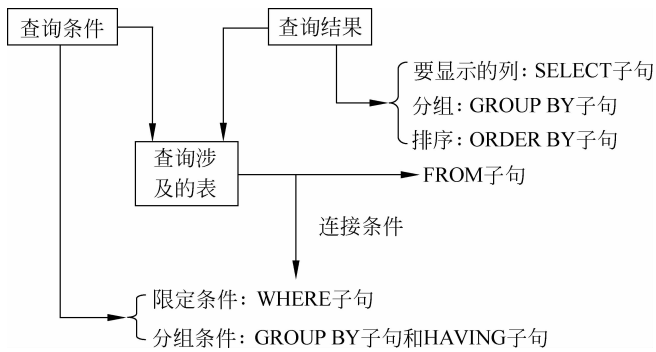


图 5.1　查询分析图

已知教学管理数据库系统中的 7 个关系模式如下：

学生关系模式：S(S#,SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS)

专业关系模式：SS(SCODE#,SSNAME)

课程关系模式：C(C#,CNAME,CLASSH)

设置关系模式：CS(SCODE#,C#)

学习关系模式：SC(S#,C#,GRADE)

教师关系模式：T(T#,TNAME,TSEX,TBIRTHIN,TITLEOF,TRSECTION,TEL)

讲授关系模式：TEACH(T#,C#)

其中，各属性名标识的中文意义如下：

S#(学号)，SNAME(学生姓名)，SSEX(学生性别)，SBIRTHIN(学生出生年月)，PLACEOFB(学生籍贯)，SCODE#(专业代码)，CLASS(班级)，SSNAME(专业名称)，C#(课程号)，CNAME(课程名)，CLASSH(学时)，GRADE(分数)，T#(教职工号)，TNAME(教师姓名)，TSEX(教师性别)，TBIRTHIN(教师出生年月)，TITLEOF(职称)，TRSECTION(所在教研室)，TEL(电话)。

在下面的例题和习题中，如不加特别说明，所提到的关系模式 S、SS、C、CS、SC、T 和 TEACH 均是以教学管理数据库为例详细说明和练习 SQL 语句的用法。

例 5.1　检索选修了“计算机应用”课程的女学生的姓名。

分析：

(1) 查询结果：学生姓名——涉及表 S。

(2)查询条件：

① 选修了“计算机应用”课程——涉及的表是 C，限定条件 CNAME='计算机应用'；

② 女学生——涉及表 S，限定条件 SSEX='女'。

(3) 综上分析，本查询涉及的表有 S 和 C。由于 S 和 C 没有相同的属性，所以引入表 SC 进行衔接，连接条件为 S.S#=SC.S#　AND　SC.C#=C.C#。

解：
```
SELSELECT  SNAME
FROM  S,SC,C
WHERE  S.S# = SC.S# AND SC.C# = C.C#
AND CNAME = '计算机应用' AND SSEX = '女';
```

5.4.2 嵌套查询

在SQL语言中，如果在一个SELECT语句的WHERE子句中嵌入了另一个SELECT语句，则称这种查询为嵌套查询。WHERE子句中的SELECT语句称为子查询，外部的查询称为父查询。用户可以利用嵌套查询技术由多个简单查询构成复杂的查询，从而增加SQL的查询能力。以分层嵌套的方式构造程序正是SQL中“结构化”的含义所在。

子查询分为一般子查询和相关子查询两种，其区别在于相关子查询的检索条件中引用了外部查询的列，把外部查询的列值作为子查询检索条件的条件值。由于这个区别，两种子查询的执行方式不同。

一般子查询的求解方法是由里向外逐层处理，即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件。在该过程中子查询只执行一次，按照子查询返回的是单个值还是一组值选择相应的条件表达形式。

与一般子查询不同，相关子查询的执行顺序如下：

(1) 由外部查询选取一行记录(称为候选行)；

(2) 内部的子查询利用候选行中相关列的值查询结果数据；

(3) 外部查询利用子查询返回的结果集合判断候选行是否满足检索条件，如果满足条件，则把该候选行放入外部查询的结果集合；

(4) 重复执行上述步骤直到处理完外部查询的每一行候选行为止。

在整个查询过程中，根据外部查询候选行的行数，相关的子查询将执行相应的次数。在相关子查询中经常使用更名(详见例5.3)、测试谓词EXISTS和NOT EXISTS。

下面同样以教学管理数据库为例进行说明。

例5.2 检索数据结构课(课程号为C401001)成绩最高的学生的学号。

分析：

(1) 查询结果：学生学号——涉及表SC。

(2) 查询条件：数据结构课(课程号为C401001)的最高成绩，即“数据结构课成绩＝数据结构课最高成绩”，需要用子查询来实现。

① 从SC中检索出数据结构课(课程号为C401001)的最高成绩 x；

② 从SC中检索“数据结构课成绩＝x”的学生的学号。

解：
```
SELECT  S#
    FROM  SC
    WHERE  C# = 'C401001'AND GRADE = (SELECT  MAX(GRADE)
                                      FROM  SC
                                      WHERE  C# = 'C401001');
```

例5.3 检索成绩比该课程平均成绩低的学生成绩表。

分析：

(1) 查询结果：学生成绩表——涉及表SC。

(2) 查询条件：成绩比该课程平均成绩低，即“成绩＜候选行学生所修课程的平均成绩”，需要用子查询来实现。

① 从SC中检索出“课程号＝父查询候选行的课程号 x”的课程的平均成绩 y；

② 从 SC 中检索“成绩＜ y”的学生成绩表。

同一个关系出现在父查询和子查询中，且子查询的条件涉及父查询的属性，因此使用更名进行描述，格式为“旧名　AS　新名”。

解：
```
SELECT  S#,C#,GRADE
    FROM  SC
    WHERE  GRADE < (SELECT  AVG(GRADE)
                    FROM  SC  AS  X
                    WHERE  X.C# = SC.C#);
```

5.4.3　谓词的灵活应用

在 SQL 语言的查询中，可以把谓词当作特殊的比较操作符使用，从而将许多本来比较复杂的查询问题变得容易处理。常用的谓词操作符有 10 种，其基本用法如下。

1. BETWEEN…AND 和 NOT BETWEEN…AND

格式：

```
<数值型列名> BETWEEN <下限值> AND <上限值>
<数值型列名> NOT BETWEEN <下限值> AND <上限值>
```

含义：前者表示将某数值型列的值限定在某个数值区间的情况；后者表示将某数值型列的值限定在某个数值区间外的情况。

2. LIKE

格式：

```
<字符型列名> [NOT]LIKE '<匹配串>'
```

含义：使用 LIKE 比较运算符可以进行两个字符串的匹配比较。

SQL 使用一对单引号来标识字符串，如'LI'。如果单引号是字符串的组成部分，可用两个单引号字符来表示，如字符串 it's right 可标示为'it''s　right'。

LIKE 比较运算符提供了两种通配符。

(1) 下划线通配符_：代表任意一个字符。

(2) 百分号通配符%：代表任意长度的字符串。

3. IN 和 NOT IN

格式：

```
<集合 1 > IN <集合 2 >
<集合 1 > NOT IN <集合 2 >
```

含义：前者表示如果集合 1 中的数据是集合 2 中的成员，那么其逻辑值为 true，否则为 false。后者表示如果集合 1 中的数据不是集合 2 中的成员，那么其逻辑值为 true，否则为 false。其中，集合 2 可以是一个元组集合，或者是一个 SELECT 子查询(的查询结果)。

4. ANY 和 SOME

格式：

```
<列数据> θ ANY <集合>
<列数据> θ SOME <集合>
```

含义：在 SQL 中，ANY 和 SOME 具有相同的含义，指比较运算符 θ 左边的数据与右边集合中至少有一个元素满足 θ 运算。θ 是算术比较运算符＜、＜＝、＞、＞＝、＝、！＝。

5. ALL

格式：

```
<列数据> θ ALL <集合>
```

含义：指比较运算符 θ 左边的数据与右边集合中的每一个元素满足 θ 运算。

6. EXISTS 和 NOT EXISTS

格式：

```
EXISTS(<集合>)
NOT EXISTS(<集合>)
```

含义：前者表示当集合中至少存在一个元素（非空）时，其逻辑值为 true，否则为 false；后者表示当集合中不存在任何元素（为空）时，其逻辑值为 true，否则为 false。

EXISTS 和 NOT EXISTS 是一个测试谓词，在形式上类似于一个函数，EXISTS 为存在量词，通常用于测试子查询是否有返回结果。

在 SQL 语言中没有全称量词和蕴含逻辑运算，因此对于用到全称量词和蕴含逻辑运算的查询，必须利用谓词演算将其转换为等价的带有存在量词的谓词。

例 5.4 检索全部学生都选修了的课程的课程号与课程名。

分析：用 p 表示谓词“学生 x 选修了某课程”，依据题意原题可形式化地表示为(∀x)p。

利用谓词演算将其转换为等价的带有存在量词的谓词形式。

$$(\forall x)p \equiv \neg(\exists x(\neg p))$$

其语义为检索这样的课程，不存在一个学生不选修该课程。

解：

```
SELECT  C#,CNAME
FROM    C
WHERE  NOT  EXISTS
              (SELECT   *
              FROM S
              WHERE  NOT  EXISTS
                           (SELECT   *
                           FROM  SC
                           WHERE SC.S# = S.S#
                                 AND  SC.C# = C.C#));
```

例 5.5　检索所学课程包含学号为 200401003 的学生所学课程的学生的学号。

分析：依据题意查询这样的学生，凡是学号为 200401003 学生选修的课程，他都选修了。换句话说，若有一个学号为 x 的学生，对于所有的课程 y，只要学号为 200401003 的学生选修了课程 y，则 x 也选修了 y，那么就将 x 选出来。若用 p 表示谓词“学号为 200401003 的学生选修了课程 y”，用 q 表示谓词“学号为 x 的学生选修了课程 y”，则上述查询表示为 $(\forall y)(p\rightarrow q)$。利用谓词演算将其转换为等价的带有存在量词的谓词形式：

$$(\forall y)(p\rightarrow q)\equiv\neg(\exists y(\neg(p\rightarrow q)))\equiv\neg(\exists y(\neg(\neg p\vee q)))\equiv\neg(\exists y(p\wedge\neg q))$$

其语义为不存在这样的课程 y，学号为 200401003 的学生选修了 y，但学号为 x 的学生没有选修。

解：

```
SELECT  DISTINCT S#
FROM    SC  AS  X
WHERE  NOT  EXISTS
            (SELECT  *
            FROM  SC  AS  Y
            WHERE  Y.S# = '200401003'
                    AND  NOT  EXISTS
                              (SELECT  *
                              FROM  SC  AS  Z
                              WHERE  Z.S# = X.S#
                              AND Z.C# = Y.C#));
```

5.4.4　游标的使用方法

游标机制的运用需要 4 个步骤，包括定义游标、打开游标、读取游标和关闭游标。

1. 定义游标

(1) 利用符合 SQL92 标准的 DECLARE CURSOR 语句定义游标：

```
DECLARE <游标名> [INSENSITIVE][SCROLL] CURSOR
FOR < SELECT 语句>
[FOR { READ ONLY | UPDATE [OF <列名> [, … ]]}]
```

(2) 利用 SQL 句法结构的 DECLARE CURSOR 语句定义游标：

```
DECLARE <游标名> CURSOR
[LOCAL|GLOBAL]
[FORWARD_ONLY|SCROLL]
[STATIC|KEYSET|DYNAMIC|FAST_FORWARD]
[READ_ONLY|SCROLL_LOCKS|OPTIMISTIC]
[TYPE_WARNING]
FOR < SELECT 语句>
[FOR UPDATE [OF <列名> [, … ]]}]
```

2. 打开游标

打开游标语句用于“打开”由＜游标名＞指定的游标。打开游标语句根据游标名对应的

SELECT 语句中 WHERE 子句的查询条件(若存在)得到由满足查询条件的行组成的结果集,结果集中当前等待被处理的行称为当前行。在刚打开游标时,游标指针指向结果集的第1行之前。

打开游标语句的句法格式如下：

```
OPEN {{[GLOBL] <游标名>} | <游标变量名>}
```

3. 利用游标读取数据

打开游标后,就可以利用游标读取数据了。利用 FETCH 语句可以从结果集中读取一行数据,并把结果赋给 INTO 后的输出主变量。

FETCH 语句的句法格式如下：

```
FETCH
[[NEXT|PRIOR|FIRST|LAST
|ABSOLUTE {<行数>|<@行数变量>}
|RELATIVE {<行数>|<@行数变量>}]
FROM
]
{{[GLOBAL]<游标名>}|<@游标变量名>}
[INTO  <@主变量名>[,… ]]
```

4. 利用游标修改和删除数据

利用游标修改和删除数据的关键首先是使用 FETCH 操作移动游标指针,使其指向要修改的行和要删除的行,即使要修改的行和要删除的行成为指向的当前行；然后利用 UPDATE 语句并辅以“WHERE CURRENT OF ＜游标名＞”子句进行修改数据的操作,或利用 DELETE 语句并辅以“WHERE CURRENT OF ＜游标名＞”子句进行删除修改的操作。

5. 关闭与删除游标

(1) 游标使用完后应及时关闭。关闭游标的句法格式如下：

```
CLOSE {{[GLOBAL] <游标名>}|<游标变量名>}
```

(2) 利用 DEALLOCATE 删除游标后,即可释放该游标占用的内存及其他系统资源。删除游标的句法格式如下：

```
DEALLOCATE {{[GLOBAL] <游标名>}|<@游标变量名>}
```

5.5 典型例题剖析

5.5.1 基本表的定义

基本表的定义是由 CREATE TABLE 语句实现的。在定义一个基本表的时候,首先要

明确表的名称；其次要确定该表包括哪些属性；再次为每个属性命名并指定数据类型，对于需要进行完整性约束的还要描述该属性的完整性约束条件；最后要考虑是否需要描述表的完整性约束条件。

例 5.6　写出建立教学管理数据库系统中的教师关系表 T 的表定义语句，T 的关系模式为 T(T#，TNAME，TSEX，TBIRTHIN，TITLEOF，TRSECTION，TEL)。

解：教师关系表的名称为 T，包括教职工号 T#、姓名 TNAME、性别 TSEX、出生年月 TBIRTHIN、职称 TITLEOF、教研室 TRSECTION 和电话 TEL 7 个属性。

```
CREATE TABLE T
        (T#          CHAR(8)       PRIMARY KEY,
         TNAME       VARCHAR(10)   NOT NULL,
         TSEX        CHAR(2)       CHECK(SSEX IN ('男','女')),
         TBIRTHIN    DATE          NOT NULL,
         TITLEOF     VARCHAR(8),
         TRSECTION   VARCHAR(16)   NOT NULL,
         TEL         CHAR(5));
```

5.5.2　根据题意书写 SQL 数据查询语句

1. 投影查询

使用 SELECT 命令可以选择查询表中的任意列。如果要将 FROM 中的所有关系的所有属性都显示在查询结果中，可以用 * 表示。如果要去掉重复的显示列，可以使用 DISTINCT 选项。如果要改变查询结果中显示的列名，可以使用更名方式为该列重新命名。此外，SELECT 中可含有带＋、－、*、/的算术表达式，运算对象可以是常数或元组的属性。

例 5.7　查询所有学生的姓名、性别、出生年月和班级。

解：
```
SELECT  SNAME,SSEX,SBIRTHIN,CLASS
    FROM  S;
```

例 5.8　查询每个学生的出生年份。

分析：学生的出生年份可利用函数 YEAR 从 SBIRTHIN 得到，并可运用更名机制为其命名。

解：
```
SELECT  SNAME,YEAR(SBIRTHIN) AS BIRTH_YEAR
    FROM  S;
```

2. 条件查询

条件查询就是在 WHERE 子句中用条件表达式指定查询条件，然后从源表中提取或显示满足该查询条件的记录。

例 5.9　检索 200402 班男学生的信息。

分析：

(1) 查询结果：学生信息——涉及表 S；显示学生的全部属性——利用 SELECT * 。

(2) 查询条件：

① 200402 班——涉及表 S，限定条件 CLASS＝'200402'；

② 男学生——涉及表 S，限定条件 SSEX＝'男'。

解：
```
SELECT  *
FROM  S
WHERE  CLASS = '200402' AND SSEX = '男';
```

例 5.10 检索计算机网络课（课程号为 C403001）成绩为 85、88 或 90 的学生的学号。

分析：

（1）查询结果：学号——涉及表 SC。

（2）查询条件：

① 计算机网络课（课程号为 C403001）——涉及表 SC，限定条件 C＃＝'C403001'；

② 成绩为 85、88 或 90——这是一个集合，可以用 IN 来表示限定条件 GRADE IN（85，88，90）。

解：
```
SELECT  S＃
FROM  SC
WHERE  C＃ = 'C403001' AND GRADE IN (85,88,90);
```

3. 聚合函数的使用

聚合函数是能够根据查询结果的记录集或根据查询结果的记录集中某列值的特点返回一个汇总信息的函数。聚合函数用于实现数据统计等功能。

例 5.11 检索 200301 班的学生人数。

分析：

（1）查询结果：学生人数——涉及表 S；

学生人数——即进行记录的统计，利用 SELECT　COUNT(＊)。

（2）查询条件：200301 班——涉及表 S，限定条件 CLASS＝'200301'。

解：
```
SELECT  COUNT(*)
FROM  S
WHERE  CLASS = '200301';
```

4. 分组查询

在 SQL 语言中，把元组按某个或某些列上相同的值分组，然后对各组进行相应操作的查询方式称为分组查询。运用 GROUP BY 子句可以实现分组查询，它按照指定的列把所有该属性有相同值的元组放在一组，从而把单个的元组集分成若干组，再按照 SELECT 子句的要求输出。使用 HAVING 子句可以对这些组进一步加以控制，只选择出满足 HAVING 子句指定条件的分组。

例 5.12 检索最低分大于 70、最高分小于 90 的学生的学号。

分析：

（1）查询结果：学号——涉及表 SC。

（2）查询条件：最低分大于 70、最高分小于 90，这是一个分组条件。要求列出每个学生成绩的最低分和最高分，需通过 GROUP BY 子句将表 SC 按学号 S＃的值分成若干组，使

每个学生所学的各门课程分在同一个组里。然后就可以利用聚合函数 MIN 和 MAX 求每个学生成绩的最低分和最高分，并构造 HAVING 子句的限定条件。

解：
```
SELECT  S#
FROM  SC
GROUP BY  S#
HAVING MIN(GRADE)>70  AND  MAX(GRADE)<90;
```

例 5.13　检索至少有 5 名学生选修的并以 3 开头的课程号的平均分数。

分析：

(1) 查询结果：某课程的平均分数——涉及表 SC；
　　某课程的平均分数——SELECT　C#，AVG(GRADE)；
　　某课程的平均分数——要按照课程号用 GROUP BY 子句进行分组。

(2) 查询条件：

① 以 3 开头的课程号——WHERE 限定条件 C#　LIKE　'3%'；

② 至少有 5 名学生选修——分组条件，需通过 GROUP BY 子句将表 SC 按学号 C# 的值分成若干组，使选修每个课程的学生的成绩分在同一个组里，再构造 HAVING 子句的限定条件 COUNT(*)≥5。

解：
```
SELECT  C#,AVG(GRADE)
FROM  SC
WHERE  C#  LIKE '3%'
GROUP BY  C#
HAVING COUNT(*)≥5;
```

5. 排序查询

SQL 提供了 ORDER BY 子句用来控制查询结果中元组的排列顺序。其语句格式如下：

```
ORDER BY <列名> [ASC/DESC][, <列名> [,ASC/DESC]]…
```

其中，<列名>指对查询结果进行排序显示所依据的列的名称。ASC 是按递增顺序显示，DESC 是按递减顺序显示，默认为按递增顺序显示。当按多个列进行排序时，要分别指出各个列名及它们所对应的递增或递减方式。系统会先按指定的第一个列排序；当第一个列相同时，再按第二个列排序；当第二个列相同时，再按第三个列排序，其余以此类推。

注意：在 SQL 查询语句中，ORDER BY 子句必须在所有其他子句之后作为最后一个子句出现。

例 5.14　检索所有女生记录，并以班级降序排列。

分析：

(1) 查询结果：学生记录——涉及表 S；
　　显示学生记录的全部属性——SELECT　*；
　　以班级降序排列——ORDER BY　CLASS　DESC。

（2）查询条件：女生——限定条件为 SSEX＝'女'。

解：
```
SELECT  *
FROM  S
WHERE SSEX = '女'
ORDER BY  CLASS  DESC;
```

例 5.15 统计每门课程的学生选修人数（超过 10 人的课程才统计），要求输出课程号和选修人数，查询结果按人数降序排列，若人数相同，按课程号升序排列。

分析：

（1）查询结果：

① 输出课程号和选修人数——涉及表 SC；

② 输出课程号和选修人数——SELECT　C#，COUNT(S#)；

③ 选修人数——要按照课程号进行分组；

④ 按人数降序排列，若人数相同，按课程号升序排列——这规定了 ORDER BY 中依据排序的列的次序，首先是人数，其次是课程号，由于人数是由聚合函数求得的，故需用列号表示，即“ORDER BY　2　DESC，C#　ASC”。

（2）查询条件：超过 10 人的课程才统计——分组条件为 HAVING COUNT(*)＞10。

解：
```
SELECT  C#,COUNT(S#)
FROM  SC
GROUP BY C#
HAVING COUNT( * )> 10
ORDER BY  2  DESC,C#  ASC;
```

6. 连接查询

同时涉及两个或两个以上表的查询称为连接查询。连接查询通过连接条件将两个表连接起来，形式如下：

```
[<表名 1>.]<列名 1>  <比较运算符>  [<表名 2>.]<列名 2>
```

其中，当＜列名 1＞和＜列名 2＞的列名相同时，必须用前缀(.)说明属于哪个表；如果列名是唯一的，可不加前缀。

例 5.16 检索成绩在 85 分以上的学生的姓名及所学课程号和成绩。

分析：

（1）查询结果：学生姓名——涉及表 S；

所学课程号和成绩——涉及表 SC。

（2）查询条件：

① 成绩在 85 分以上——限定条件 GRADE ＞＝ 85；

② 由于该查询涉及两个表 S 和 SC，需用连接条件将两个表联系起来，即“S.S#＝SC.S#”。

解：
```
SELECT  SNAME,C#,GRADE
FROM  S,SC
WHERE S.S# = SC.S#  AND  GRADE >= 85;
```

例 5.17　检索选修数据结构课程的学生的姓名及所在专业代码。

分析：

(1) 查询结果：学生姓名及所在专业代码——涉及表 S。

(2) 查询条件：

① 选修数据结构课程——涉及表 C，限定条件 CNAME ='数据结构'；

② 该查询涉及两个表 S 和 C，要把 S 和 C 连接起来，需引入表 SC 作为“桥梁”。先在 C 中找到数据结构课程的课程号，然后在 SC 中通过课程号找到选修了该课程的学生的学号，最后用学号在 S 中查找学生的姓名及所在专业代码，故连接条件为“S.S# = SC.S# AND SC.C# = C.C#”。

解：
```
SELECT  SNAME,SCODE#
    FROM  S,SC,C
    WHERE S.S# = SC.S#  AND  SC.C# = C.C# AND  CNAME = '数据结构';
```

7. 嵌套查询

嵌套查询在条件的构造上是多种多样的，各种谓词在其中的使用非常灵活。

例 5.18　查出所有比李明同学年龄小的学生的姓名和出生年月。

分析：所有比李明同学年龄小的学生即出生年月比李明晚的学生，先查出李明的出生年月，再用李明的出生年月作为条件查询其他同学的姓名和出生年月。

解：
```
SELECT  SNAME,SBIRTHIN
    FROM  S
    WHERE  SBIRTHIN > (SELECT  SBIRTHIN
                       FROM  S
                       WHERE  SNAME = '李明');
```

例 5.19　查出选修了课程号为 C403001 的课程的学生的姓名、性别和专业编码。

分析：这个查询要用到两个表，一个是 S，一个是 SC。先在 SC 中查找选修了课程号为 C403001 的课程的学生的学号，形成一个子查询块结果；然后在 S 中查找存在于子查询块结果中的学生姓名、性别和专业编码。构造条件时可使用谓词 IN、= ANY 或 EXISTS，但要注意区别子查询的类型。

解：答案 1

```
SELECT SNAME,SSEX,SCODE#
FROM  S
WHERE  S# IN (SELECT  S#
              FROM  SC
              WHERE  C# = 'C403001');
```

答案 2

```
SELECT SNAME,SSEX,SCODE#
FROM  S
WHERE  S# = ANY (SELECT  S#
                 FROM  SC
                 WHERE  C# = 'C403001');
```

答案 3

```
SELECT SNAME,SSEX,SCODE#
FROM  S
WHERE  EXISTS (SELECT  *
              FROM  SC
              WHERE  S.S# = SC.S#  AND  C# = 'C403001');
```

例 5.20 检索选修了课程号为 C403003 的课程且成绩高于课程号为 C403001 的课程所有成绩的学生的学号、课程号和成绩。

分析：这个查询只用到了表 SC，但在条件的构造上会使用子查询。先在 SC 中查找课程号为 C403001 的课程的所有成绩，形成一个子查询块结果；然后在 SC 中查找 C403003 课程成绩高于该子查询块结果中每一个成绩的学生的学号、课程号和成绩。根据题意使用谓词＞ALL。

解：

```
SELECT  S#,C#,GRADE
FROM  SC
WHERE  C# = 'C403003'
AND  GRADE > ALL (SELECT  GRADE
                 FROM  SC
                 WHERE  C# = 'C403001');
```

例 5.21 检索所有未任课的教师的姓名和所在教研室。

分析：这个查询会用到两个表 T 和 TEACH。先从 TEACH 中得到任课教师的教职工号，形成一个子查询块结果；然后在 T 中找出不存在于该子查询块结果中的教师的姓名和所在教研室。根据题意可使用谓词 NOT IN 和 NOT EXISTS。

解：答案 1

```
SELECT TNAME,TRSECTION
FROM  T
WHERE  T#  NOT  IN (SELECT  T#
                    FROM  TEACH);
```

答案 2

```
SELECT TNAME,TRSECTION
FROM  T
WHERE  NOT EXISTS  (SELECT  *
                   FROM  TEACH
                   WHERE  T.T# = TEACH.T#);
```

例 5.22 找出至少学习了“计算机网络”和“信息安全技术”两门课程的学生的学号和姓名。

分析：这个查询的关键是构造限定条件——至少学习了“计算机网络”和“信息安全技术”两门课程，可以有 3 种方法实现。

(1) 通过别名的引入，使同一个关系 SC 在一层上出现两次，这样在连接结果中，一个元组中会出现一个学生学习的两门课程，从而可进行条件限定；

(2) 先找出学习了“计算机网络”课程的学生,再从中找出学习了“网络安全技术”课程的学生。

(3) 求学习了“计算机网络”课程的学生集合和学习了“网络安全技术”课程的学生集合的交集。

解：答案 1

```
SELECT  S.S#,SNAME
FROM  S,SC AS X,SC AS Y
WHERE  S.S# = X.S# AND X.S# = Y.S#
                    AND  X.C# IN (SELECT C#
                                  FROM C
                                  WHERE CNAME = '计算机网络')
                    AND  Y.C# IN (SELECT C#
                                  FROM C
                                  WHERE CNAME = '网络安全技术');
```

答案 2

```
SELECT  S.S#,SNAME
FROM  S,SC,C
WHERE S.S# = SC.S#
  AND SC.C# = C.C#
  AND CNAME = '信息安全技术'
  AND S.S# IN (SELECT S#
               FROM SC,C
               WHERE  SC.C# = C.C#
                 AND  CNAME = '计算机网络');
```

答案 3

```
SELECT  S#,SNAME
FROM S
WHERE S# IN (SELECT S#
             FROM SC,C
             WHERE  SC.C# = C.C# AND  CNAME = '计算机网络')
AND S# IN (SELECT S#
           FROM  SC,C
           WHERE  SC.C# = C.C# AND CNAME = '信息安全技术');
```

5.5.3　根据题意书写 SQL 数据更新语句

1. 数据插入

例 5.23　向学生表 S 中插入一个学生记录。

解：
```
INSERT INTO S
    VALUES('200404001','杨小燕','女','09-05-83','西安','S0403','200404');
```

例 5.24　把平均成绩大于 80 分的男学生的学号和平均成绩存到另一个关系 S_GRADE(S#,AVG_GRADE)中。

分析：这个数据插入操作的关键是找出平均成绩大于 80 分的男学生的学号和平均成绩，因此运用查询的基本知识。

解：
```
INSERT INTO S_GRADE(S#,AVG_GRADE)
SELECT S#,AVG(GRADE)
FROM  SC
WHERE  S#  IN  (SELECT  S#
                FROM  S
                WHERE  SSEX = '男')
GROUP BY  S#
HAVING  AVG(GRADE)> 80;
```

2. 数据修改

例 5.25　把课程号为 C401003 的成绩提高 5%。

解：
```
UPDATE SC
SET  GRADE = GRADE * 1.05
WHERE  C#  = 'C401003';
```

例 5.26　把课程名为数学的成绩提高 10%。

解：
```
UPDATE SC
SET  GRADE = GRADE * 1.1
WHERE  C# IN  (SELECT  C#
               FROM  C
               WHERE  CNAME = '数学');
```

3. 数据删除

例 5.27　从 S 中删除专业号为 S0102 的学生记录。

解：
```
DELETE  FROM S
WHERE  SCODE#  = 'S0102';
```

例 5.28　把课程号为 C402005 的课程中小于该课程平均成绩的成绩从关系 SC 中删除。

解：
```
DELETE  FROM SC
WHERE  C#  = 'C402005'
AND  GRADE < (SELECT  AVG(GRADE)
              FROM  SC
              WHERE C#  =  'C402005');
```

5.5.4　视图的定义

例 5.29　创建一个学生成绩视图 S_GRADE，其中的属性包括学号（S#）、选修课程（成绩为非空）的门数（C_NUM）和平均成绩（AVG_GRADE）。

解：
```
CREATE VIEW S_GRADE(S#,C_NUM,AVG_GRADE)
AS SELECT S#,COUNT(C#),AVG(GRADE)
   FROM  SC
```

```
WHERE  GRADE IS NOT NULL
GROUP BY  S#;
```

5.6　自测练习题5及答案

5.6.1　自测练习题5

一、填空题

(1) SQL语言是一种_____的_____。SQL语言只需要用户指出“做什么”，至于“如何做”是由_____实现的。

(2) SQL数据库语言包括_____和_____两部分。

(3) SQL语言具有两种使用方式，一种是_____使用方式，另一种是_____使用方式。

(4) 数据库应用系统中的_____构成了关系数据库的全局逻辑模式。

(5) 用于存储用户数据的_____构成了关系数据库的内模式。

(6) 当表的主键由多个列名组合而成时，必须将表的主键定义成_____格式。当然，当表中只有一个列是主键时，也可以将表的主键定义成_____格式。

(7) 表检验约束CHECK子句中的＜值的约束条件＞既可以是_____，也可以是一个包含_____。

(8) 当查询包含空值的记录时，使用_____比较运算符；当查询不包含空值的记录时，使用_____比较运算符。

(9) 在一个SELECT语句中，若＜列名表＞为“$A_1,A_2,\cdots,A_n$”，＜表名表＞为“$R_1,R_2,\cdots,R_m$”，则该查询语句的意义可用关系代数表达式_____来表示。

(10) 视图是一个虚表，在_____上由创建视图语句定义。数据库管理系统只存储视图的_____，不存储视图结构形式的数据库表的_____。

(11) 在宿主语言系统中，对_____由数据操作语言命令实现，而把对_____留给主语言完成。

(12) 游标是一个与某一个SELECT语句相联系的_____。对游标的操作过程分为_____、_____、_____和_____4个步骤。

二、单项选择题

(1) SQL是_____的缩写。

A. Structured Query Language　　B. Standard Query Language

C. Select Query Language　　D. Specifying Queries Language

(2) 关于UNIQUE约束，下面描述正确的是_____。

A. UNIQUE指出该列是唯一的索引

B. UNIQUE指出的列的值不能相同

C. UNIQUE指出的列的值不能不相同

D. UNIQUE指出的列的值是唯一的主键值

（3）下列不属于 SQL 语言的数据更新语句的是_____。

A. INSERT　　B. ALTER

C. UPDATE　　D. DELETE

（4）下列有关通配符%的含义，其表述正确的是_____。

A. “%”代表一个字符　　B. “%”代表一个汉字

C. “%”代表多个字符　　D. “%”代表零个或多个字符

（5）在 SELECT 查询语句中，当_____必须使用分组子句。

A. 只有一个聚合值时　　B. 有两个无关的聚合值时

C. 当聚合值与其他属性有关时　　D. 无聚合值时

（6）当要进行条件分组查询时，_____。

A. 必须使用 ORDER BY 子句

B. 必须使用 HAVING 子句

C. 只要使用 GROUP BY 子句

D. 应先使用 WHERE，再接着使用 HAVING

（7）在 SQL 语言中，与 NOT IN 等价的操作符是_____。

A. =SOME　　B. =ALL

C. <>SOME　　D. <>ALL

（8）在 SQL 语言中，短语_____用于实现实体的完整性约束。

A. FOREIGN KEY　　B. PRIMARY KEY

C. UNIQUE　　D. NOT NULL

（9）带有子查询条件的 UPDATE 语句用于_____。

A. 数据插入操作　　B. 数据删除操作

C. 数据更新操作　　D. 子图数据查询操作

（10）嵌套查询的含义是_____。

A. WHERE 子句中嵌入了复杂条件

B. SELECT 语句的 WHERE 子句中嵌入了另一个 SELECT 语句

C. WHERE 子句中嵌入了聚合函数

D. WHERE 子句中涉及表的更名查询

三、简答题

（1）在 SQL 的查询语句中，使用分组子句的原则是什么？

（2）请通过引入关系的别名写出 SQL 查询语句：找出至少学习了“计算机网络”和“信息安全技术”两门课程的学生的学号和姓名。

（3）为什么把 SQL 中的视图称为“虚表”？视图的作用是什么？

（4）SQL 查询语句是怎样实现简单的模糊查询的？

（5）用户视图对数据库设计带来了哪些好处？

5.6.2 自测练习题 5 答案

一、填空题

(1)非过程化　关系数据库语言　DBMS

(2) 数据描述语言　数据操作语言

(3) 交互式 SQL　嵌入式 SQL

(4) 全体基本表

(5) 所有存储文件

(6) 表约束　列约束

(7) 一个条件表达式　SELECT 的 SQL 语句

(8) IS NULL　IS NOT NULL

(9) $\pi_{A_1,A_2,\cdots,A_n}(\sigma_F(R_1\times R_2\times\cdots\times R_m))$

(10) 一个或多个基本表或视图　定义　数据

(11) 数据库中数据的操作　数据的各种数据处理操作

(12) 符号名　定义游标　打开游标　取一行数据　关闭游标

二、单项选择题

(1) A　(2) B　(3) B　(4) D　(5) C

(6) B　(7) D　(8) B　(9) C　(10) B

三、简答题

(1) **答**：在 SQL 的查询语句 SELECT 中，使用分组子句的原则如下。

① SELECT 语句中必须有聚合函数。

② 当聚合函数与其他属性值无关时不需要使用分组子句；当聚合函数与其他属性值有关时需要使用分组子句。

(2) **答**：通过引入别名，可以使同一个关系 SC 在一层上出现两次，从而可以在连接结果中使一个元组中出现一个学生学习的两门课程。

```
SELECT  S.S#,SNAME
FROM  S,SC AS X,SC AS Y
WHERE  S.S# = X.S#
AND X.S# = Y.S#
AND  X.C# IN (SELECT C#
          FROM C
          WHERE CNAME = '计算机网络')
AND  Y.C# IN (SELECT C#
          FROM C
          WHERE CNAME = '网络安全技术');
```

(3) **答**：因为视图是从一个或几个基本表(或视图)通过映射关系导出来的表，当视图建立后，系统只将视图的定义存放在数据字典中，并不存储与视图结构对应的数据，只有当应用程序使用视图时，才按视图定义语句中的查询条件和属性输出格式求得对应的数据，所以称为虚表。

视图的作用是简化应用程序设计，给用户提供以多种角度观察数据库中数据的机制，支持数据库的逻辑独立性和方便数据库的重构，对数据库中的数据提供一定程度的安全保护。

(4) **答**：SQL 查询语句是利用 LIKE 比较运算符的字符串匹配功能实现简单的模糊查询的。具体来说，利用下划线“_”实现了与任意一个字符的匹配，利用百分号“%”实现了与

任意一个长度大于等于零的子字符串的匹配，从而利用这两种字符串匹配功能就实现了简单的模糊查询。

（5）**答**：利用视图的定义功能，可以提前把带有复杂查询条件的查询语句定义成用户视图，这样在应用程序设计中就可以利用简单的视图查询语句代替具有复杂查询条件的查询语句，从而可以简化用户应用程序接口，使应用程序中的 SQL 语句变得简单明了、清晰可读；使应用程序员把编写应用程序的主要精力集中在对数据的分析、处理和用户界面的实现上，方便应用程序的设计。

另外，用户视图给数据库提供了逻辑数据独立性和物理数据独立性。

5.7 习题 5 解答

习题 5-1 解释下列术语。

（1）基本表：在 SQL 语言中把关系模式称为基本表。

（2）列级完整性约束：在数据库表创建语句中，由用户根据各属性列数据的特点和要求对列属性的取值域所加的约束条件。

（3）表级完整性约束：在数据库表创建语句中，由用户根据该表属性列数据的特点和要求对该表中各（若干）元组之间的联系、主键值等所加的约束条件。

（4）嵌套查询：在 SQL 语言中，如果在一个 SELECT 语句的 WHERE 子句中嵌入了另一个 SELECT 语句，则称这种查询为嵌套查询。

（5）分组查询：在 SQL 语言中，把元组按某个或某些列上相同的值分组，然后对各组进行相应操作的查询方式称为分组查询。

（6）多元查询：SQL 语言允许用户在同一查询语句中从两个或多个表中查询数据，即在两个表或多个表的连接运算的基础上从其连接结果中选取满足查询条件的元组，一般称为二元查询或多元查询。

（7）聚合函数：能够根据查询结果的记录集或根据查询结果的记录集中某列值的特点返回一个汇总信息的函数称为聚合函数。

（8）虚表：虚表是在实际的 SQL 数据库中不存在的表。

（9）嵌入式 SQL：指把 SQL 语言语句嵌入到某种高级语言中的应用形式。

（10）主语言：嵌入有 SQL 语句的高级语言称为宿主语言，简称主语言。

（11）主变量：即主语言和 SQL 语句都可以对其赋值和引用其值的变量。

（12）结果集：在 SELECT 语句的查询操作中，其查询结果是满足该语句的 WHERE 子句条件的所有记录组成的集合，称其为记录集或结果集，一般放在内存中开辟的一块区域中。

习题 5-2 分别简述数据定义语句、数据查询语句、数据操纵语句和数据控制语句的功能。

答：数据定义语句用于定义数据库的逻辑结构，包括定义基本表、定义视图和定义索引。数据查询语句按不同查询条件实现对数据库中数据的检索查询。数据操纵语句用于更改和操作表中的数据，包括数据插入、数据修改和数据删除。数据控制语句实现用户授权、基本表和视图授权、事务控制、完整性和安全性控制等。

习题 5-3　简述空值(NULL)的含义。

答：所谓该列可以为空值(NULL)是指表中的某些元组(在 SQL 语言中将元组称为行)在该列的值可以暂时不输入。

习题 5-4　主键与列值非空(NOT NULL)有什么区别与联系?

答：SQL 语言规定,主键列总具有唯一且非空的值。也就是说,定义为主键的列已经隐含具有 NOT NULL 选项和 UNIQUE 选项。列值非空(NOT NULL)表示该列不能为空值,指表中的所有元组在该列必须有确定的值。主键一定是列值非空,但列值非空不一定是主键。

习题 5-5　分别利用查询编辑器建立教学管理数据库系统中的学生关系表 S、课程关系表 C、教师关系表 T。

解：

(1)
```
CREATE TABLE S
              (S#       CHAR(9)PRIMARY KEY,
               SNAME VARCHAR (10)   NOT NULL,
               SSEX     CHAR(2)   CHECK(SSEX IN ('男','女')),
               SBIRTHIN DATE   NOT NULL,
               PLACEOFB VARCHAR (16),
               SCODE#   CHAR(5) NOT NULL,
               CLASS     CHAR(5)   NOT NULL);
```

(2)
```
CREATE TABLE C
              (C#       CHAR(7)   PRIMARY KEY,
               CNAME     VARCHAR(30)   NOT NULL,
               CLASSH   INT   NOT NULL);
```

(3)
```
CREATE TABLE T
              (T#       CHAR(8)   PRIMARY KEY,
               TNAME     VARCHAR(10)   NOT NULL,
               TSEX     CHAR(2)   CHECK(SSEX IN ('男','女')),
               TBIRTHIN DATE   NOT NULL,
               TITLEOF   VARCHAR(8),
               TRSECTION VARCHAR(16)   NOT NULL,
               TEL       CHAR(5));
```

习题 5-6　写出给学生关系表 S 和课程设置关系表 CS 中插入数据记录的插入语句。

解：向学生表 S 中插入一个学生记录的语句实例如下。

```
INSERT INTO S
VALUES('200404001','杨小燕','女','09-05-83','西安','S0403','200404');
```

向课程设置关系表 CS 中插入一个数据记录的语句实例如下。

```
INSERT INTO  CS
VALUES('S0401','C402001');
```

习题 5-7　写出满足下列要求的删除语句。

(1) 从学生关系表 S 中删除籍贯为“上海”的所有学生的记录;

(2) 从学习关系表 SC 中删除“李建平”同学的所有课程成绩的记录。

解：

```
(1) DELETE FROM S
    WHERE  PLACEOFB = '上海';

(2) DELETE FROM SC
    WHERE S# = (SELECT S#
                FROM S
                WHERE SNAME = '李建平');
```

习题 5-8 写出满足下列要求的修改语句。

(1) 把学习关系表 SC 中“计算机网络”课程的不及格成绩全部改为 61 分。

(2) 在学习关系表 SC 中修改“数据结构”课程的成绩，若成绩低于该课程的平均成绩，则将其成绩改为该平均成绩。

解：

```
(1) UPDATE SC
    SET GRADE = 61
    WHERE GRADE < 60 AND C# IN (SELECT C#
                                FROM C
                                WHERE CNAME = '计算机网络');

(2) UPDATE SC
    SET GRADE = (SELECT AVG(GRADE)
                 FROM SC
                 WHERE C# IN (SELECT C#
                              FROM C
                              WHERE CNAME = '数据结构'))
    WHERE C# IN (SELECT C#
                 FROM C
                 WHERE CNAME = '数据结构')
    AND GRADE <(SELECT AVG(GRADE)
                FROM SC
                WHERE C# IN (SELECT C#
                             FROM C
                             WHERE CNAME = '数据结构'));
```

习题 5-9 HAVING 子句与 WHERE 子句的区别与联系是什么？

答： WHERE 子句和 HAVING 子句的作用相似，都是进行条件限定的。两者的区别之一是作用对象不同，WHERE 子句的作用对象是元组，针对表中的元组进行条件筛选，而 HAVING 子句的作用对象是由 GROUP BY 获得的分组，主要指定分组所应满足的条件；二是 WHERE 子句不能直接使用聚合函数，HAVING 子句可以使用聚合函数。

习题 5-10 根据教学管理数据库写出创建下列视图的语句。

(1) 学习成绩视图 GRADE_T，其中属性包括学号(S#)、姓名(SNAME)、课程号(C#)、课程名(CNAME)、学时(CLASSH)、成绩(GRADE)、任课教员编号(T#)、任课教员名称(TNAME)。

(2) 教学水平视图 TEACH_L，其中属性包括任课教员编号(T#)、任课教员名称(TNAME)、课程号(C#)、课程名(CNAME)、学时(CLASSH)、平均成绩(AVG_GRADE)。

解：

(1)
```
CREATE VIEW GRADE_T
AS SELECT S.S#,SNAME,C.C#,CNAME,CLASSH,GRADE,T.T#,TNAME
    FROM S,SC,C,TEACH,T
    WHERE S.S# = SC.S#
      AND SC.C# = C.C#
      AND C.C# = TEACH.C#
      AND  TEACH.T# = T.T#;
```

(2)
```
CREATE  VIEW TEACH_L(T#,TNAME,C#,CNAME,CLASSH,AVG_GRADE)
AS SELECT  T.T#,TNAME,C.C#,CNAME,CLASSH,AVG(GRADE)
    FROM T,TEACH,C,SC
    WHERE T.T# = TEACH.T#
      AND  TEACH.C# = C.C#
      AND  C.C# = SC.C#
    GROUP BY  SC.C#;
```

习题 5-11　查询全部男学生的学号和姓名。

解：
```
SELECT  S#
FROM  S
WHERE  SSEX = '男';
```

习题 5-12　找出 1981 年 1 月 1 日以后出生的所有女同学的学号和姓名。

解：
```
SELECT  S#,SNAME
FROM  S
WHERE  SSEX = '女' AND  SBIRTHIN>'1981-01-01';
```

习题 5-13　找出学习了课程名为“操作系统”的所有学生的学号、姓名、性别和专业。

解：
```
SELECT  S.S#,SNAME,SSEX,SCODE#
FROM  S,SC,C
WHERE  S.S# = SC.S#
  AND  SC.C# = C.C#
  AND  C.CNAME = '操作系统';
```

习题 5-14　找出至少学习了“刘少华”老师所讲的一门课程的所有学生的学号、姓名和专业。

解：
```
SELECT  S.S#,SNAME,SCODE#
FROM  S,SC,TEACH,T
WHERE  S.S# = SC.S#
  AND  SC.C# = TEACH.C#
  AND TEACH.T# = T.T#
  AND  TNAME = '刘少华';
```

习题 5-15　找出学习全部课程的所有学生的学号和姓名。

解：答案 1

```
SELECT  S.S#,SNAME
FROM  S
WHERE  NOT EXISTS  (SELECT *
                    FORM  C
```

```
                    WHERE NOT EXIST  (SELECT *
                                      FORM  SC
                                      WHERE  SC.S# = S.S#
                                      AND  SC.C# = C.C# ));
```

答案 2

```
SELECT  S.S#,SNAME
FROM  S,SC
WHERE S.S# = SC.S#
GROUP BY  S.S#
HAVING  COUNT DISTINCT(C#) = (SELECT  COUNT( * )
                              FROM  C);
```

习题 5-16　找出学生“王丽丽”学习的全部课程的课程号、学时数和任课老师姓名。

解：答案 1

```
SELECT  C.C#,CLASSH,TNAME;
FROM S,SC,C,TEACH,T
WHERE  S.S# = SC.S#
  AND SC.C# = C.C#
  AND C.C# = TEACH.C#
  AND TEACH.T# = T.T#
  AND  SNAME = '王丽丽';
```

答案 2

```
SELECT  C.C#,CLASSH,TNAME
FROM  C,TEACH,T
WHERE  C.C# = TEACH.C#
  AND  T.T# = TEACH.T#
  AND C.C#  IN  (SELECT  SC.C#
                 FROM  S,SC
                 WHERE  S.S# = SC.S#  AND  SNAME = '王丽丽');
```

习题 5-17　找出“刘少华”老师所开全部课程的课程号和学时数。

解：
```
SELECT  C#,CLASSH
    FROM  C
    WHERE  C#  IN  (SELECT TEACH.C#
                    FROM  T,TEACH
                    WHERE  T.T# = TEACH.T#  AND TNAME = '刘少华');
```

习题 5-18　找出全部课程的任课老师的姓名。

解：
```
SELECT  C#,TNAME
    FROM  TEACH,T
    WHERE TEACH.T# = T.T#;
```

习题 5-19　按出生年月的升序找出“计算机科学与技术”专业的男学生的学号、姓名和出生年月。

解：
```
SELECT  S#,SNAME,SBIRTHIN
    FROM  S,SS
```

```
WHERE  S.SCODE# = SS.SCODE#
AND  SSEX = '男'
AND SSNAME = '计算机科学与技术'
ORDER BY SBIRTHIN;
```

习题 5-20　找出至少学习了“计算机网络”课程的所有学生的学号和姓名。

解：
```
SELECT  S.S#,SNAME
FROM  S,SC,C
WHERE  S.S# = SC.S# AND SC.C# = C.C# AND CNAME = '计算机网络';
```

习题 5-21　找出至少学习了“计算机网络”和“信息安全技术”两门课程的学生的学号和姓名。

解：答案 1

```
SELECT  S.S#,SNAME
FROM  S,SC AS X,SC AS Y
WHERE  S.S# = X.S# AND X.S# = Y.S#
                   AND  X.C# IN (SELECT C#
                                 FROM C
                                 WHERE CNAME = '计算机网络')
                   ND  Y.C# IN (SELECT C#
                                FROM C
                                WHERE CNAME = '网络安全技术');
```

答案 2

```
SELECT  S.S#,SNAME
FROM  S,SC,C
WHERE S.S# = SC.S#
  AND SC.C# = C.C#
  AND  CNAME = '信息安全技术'
AND S.S# IN (SELECT S#
             FROM SC,C
             WHERE  SC.C# = C.C#
               AND  CNAME = '计算机网络');
```

答案 3

```
SELECT  S#,SNAME
FROM S
WHERE S# IN (SELECT S#
             FROM SC,C
             WHERE  SC.C# = C.C#
             AND  CNAME = '计算机网络')
AND S# IN (SELECT S#
           FROM  SC,C
           WHERE  SC.C# = C.C#
             AND CNAME = '信息安全技术');
```

习题 5-22　找出没有学习“计算机网络”课程的所有学生的学号、姓名和专业名称。

解：
```
SELECT  S#,SNAME,SSNAME
FROM  S,SS
WHERE  S.SCODE# = SS.SCODE#
```

```
AND  S#  NOT IN  (SELECT  S.S#
                  FROM  SC,C
                  WHERE  SC.C# = C.C# AND  CNAME = '计算机网络');
```

习题 5-23 找出至少有 5 个学生学习的课程的课程号和课程名。

解：

```
SELECT  C.C#,CNAME
FROM  SC,C
WHERE SC.C# = C.C#
GROUP BY C.C#
HAVING COUNT( * )>= 5;
```