第3章 关系数据库语言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,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,1993 年我国也批准为中国国家标准。随着 SQL 语言的发展和完善,至 1999 年国际标准化组织已公布了最新的 SQL 标准 SQL-99,也即 SQL-3。SQL 在世界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。也进一步凸显出学习数据库技术和 SQL 语言的重要性。

3.1 SQL 的功能与特点

在 SQL 语言中,把关系模式称为基本表(base table),简称为表,也称为基本关系;有时在容易与上下文有关概念相混淆的地方也称为关系表。

3.1.1 SQL 的功能

SQL 语言按各语句完成的功能主要分为数据定义语句、数据操纵语句和数据控制语句 3 大类,相应的功能也分为 3 类。

1. 数据定义功能

SQL 的数据定义功能包括定义基本表、定义视图、定义索引等,由 SQL 语言的数据定义语句实现。

本章将介绍基本表和视图的定义,及基本表的变更(修改)。由于索引属于数据库的物理存储中涉及的问题,将在 6.6 节中介绍。

2. 数据库操作功能

SQL 的数据库操作功能包括数据查询和数据更新。数据查询是指按照某种要求从数

据库中检索出需要的数据，并对其进行统计、分组、排序等，由 SQL 语言的数据查询语句实现；数据更新包括数据的插入、删除、修改等数据维护操作，由 SQL 语言的更新类语句实现。

3. 数据控制功能

SQL 的数据库控制功能包括用户授权、基本表和视图授权、事务控制、数据完整性和安全性控制等，由 SQL 的数据控制类语句实现。

3.1.2　SQL 的特点

SQL 语言集数据定义、数据查询、数据控制功能于一体；简捷易学，灵活易用；非过程性强，开发应用过程简单。同时在应用中并具有以下两个特点。

1. SQL 具有交互式命令和嵌入式两种工作方式

SQL 语言提供了交互式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，用户可以在联机终端上以交互式方式通过直接输入 SQL 命令（语句）对数据库进行操作；在嵌入式工作方式下，SQL 语句可以嵌入到某种高级语言（如 C、Java、PL/1 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）的强大计算功能、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。

2. SQL 支持数据库的三级模式结构

SQL 语言支持的关系数据库三级模式结构如图 3.1 所示。视图和部分基本表构成了关系数据库的外模式。视图由某个或某些数据库表中满足一定条件约束的数据组成，从用户和程序员的观点看，视图和基本表是一样的。数据库应用系统中的全体基本表构成了关系数据库的全局逻辑模式。用于存储用户数据的所有存储文件构成了关系数据库的内模式；一般情况下，一个表可以带有一个或多个索引，一个或多个表存放在一个存储文件中。存储文件对用户是透明的。

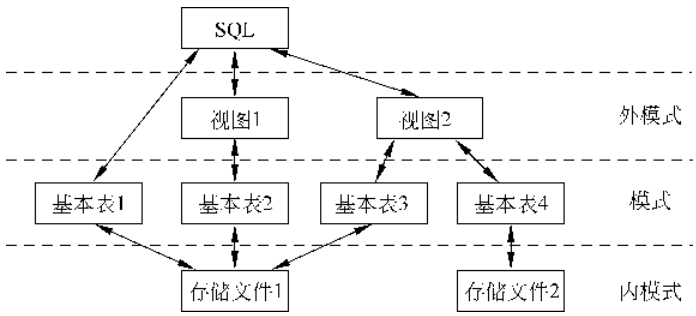

图 3.1　SQL 对关系数据库三级模式的支持

本章将以交互式命令方式的形式介绍 SQL 语言中主要的语句及其功能。在应用举例中除特殊声明外，总是假设使用图 1.7 给出的大学教学管理数据库中的关系及其当前值和

图 1.9 所示的大学教学管理数据库概念模式中的关系模式。

3.2 表的基本操作

3.2.1 表的定义、修改与撤销

1. 表的定义

SQL 数据库是表的集合,所以用户要建立数据库时就要定义表。在 SQL 语言中,表的定义是由 CREATE TABLE 语句实现的。

表的创建定义语句格式为:

```
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>]]);
```

其中:

(1) "< >"表示该项是必选项,"[ ]"表示该项是可选项。SQL 语言中的语句都必须以分号";"结束。

(2) <表名>是要定义的表的名称。表名不能与 SQL 语言中的保留字同名,不能与其他表名或视图名同名。表名和列名是以字母开头,由字母、数字和下划线"_"组成的字符串,长度不超过 30 个字符。

(3) 一个表至少要有一列(在 SQL 语言中将属性称为列),每一列必须有一个列名和相应的数据类型,同一表中的列名不能重名。

(4) SQL 语言的典型数据类型如表 3.1 所示。

表 3.1 SQL 中的典型数据类型

数据类型	表示符号	表示方式说明
字符型数据	CHAR(n)	长度为 n 的定长字符数据,长度不够时用空白字符补充
	VARCHAR(n)	长度为 n 的变长字符数据,长度不够时不需补充字符
数值型数据	SMALLINT	半字长整型数据,范围为-2^{15}～$+2^{15}$,占用 2 个字节
	INT/INTEGER	全字长整型数据,范围为-2^{31}～$+2^{31}$,占用 4 个字节
	DECIMAL(p[,q])	长度为 p 位的十进制数据,小数位数为 q,无小数时 q 可省略不写
	FLOAT	双字长浮点数,字长为 64 位
二进制型数据	BINARY[(n)]	长度为 n 的定长二进制数据,占用 n+4 个字节
	VAR BINARY[(n)]	长度为 n 的变长二进制数据,占用实际字符数+4 个字节
	IMAGE	长度为 n 的变长二进制数据,最大长度$2^{31}-1$个字节
日期时间型数据	DATETIME	日期型数据,形式为 YYYY-MM-DD
	TIME	时间型数据,形式为 HH:MM:SS

(5) [<列的完整性约束>]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(也即该项空缺)时,默认为 NULL,表示该列可为空值;当选该选项(也即该项不空缺)时,该选项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。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:用于指出该列可以为空值,其含义是指该列的值还没确定(例如在学习关系 SC 中,当某课程还没有开设或还没有考试时,其中的"分数 GRADE"列可以为空值,也即可以先不输入数据),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:用于指出该列不能为空值,也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。如果在表的完整性约束部分定义有主键,则除主键以外的所有其他列的可选项都可以是 NULL,因为主键列已隐含可选项为 NOT NULL。

③ PRIMARY KEY:当表中只有一个列是主键时,只要在说明该列的数据类型之后添加 PRIMARY KEY 即可。

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。

例 3.1　图 1.9 所示的大学教学管理数据库中的专业关系 SS,可用如下的表定义语句定义:

```
CREATE TABLE SS
   (SCODE# CHAR(5) PRIMARY KEY,
    SSNAME VARCHAR(30) NOT NULL);
```

④ DEFAULT:用于给所在的列设置一个默认值,也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。之所以使用默认选项,是因为对于一些数值型列,如果不输入数据,其中就可能是空值(NULL)而不是 0,由于在对数值型列进行求和等计算时,空值会使计算操作出错,所以对暂时不输入数据的数值型列一般应施加 DEFAULT 约束。DEFAULT 约束的格式为:

```
DEFAULT(<默认值>)
```

⑤ 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。格式为:

```
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,系统先要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。

例 3.2　图 1.9 所示的大学教学管理数据库中的学生关系表 S,可用如下的表定义语句定义:

```
CREATE TABLE S
   (S#       CHAR(9) PRIMARY KEY,
```

```
SNAME     CHAR(10) NOT NULL,
SSEX      CHAR(2) CHECK(SSEX IN ('男','女')),
SBIRTHIN  DATETIME NOT NULL,
PLACEOFB  CHAR(16),
SCODE#    CHAR(5) NOT NULL,
CLASS     CHAR(5) NOT NULL);
```

其中,对性别 SSEX 的约束表示该属性的值只能取“男”和“女”两个值中的一个。

⑥ UNIQUE:用于指出所在列的值对于所在的表来说是唯一的,也即该列的值不允许相同。例如,如果某些表中需要设置一个表示元组顺序的“序号”列,则将该列的约束设置成 UNIQUE 是比较恰当的。

(6)[<表的完整性约束>]是一个可选项,当选该可选项时,<表的完整性约束>用于给出所在表的数据的约束条件。<表的完整性约束>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:

① 表的主键约束子句,格式为:

```
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。

例 3.3 大学教学管理数据库中的学习关系 SC,可用如下的表定义语句定义:

```
CREATE TABLE SC
   (S# CHAR(9),
    C# CHAR(7),
    GRADE SMALLINT DEFAULT(0),
    PRIMARY KEY(S#,C#));
```

其中,分数 GRADE 属性可先不输入数据,但不能将它设置成空值(NULL),因为当对 GRADE 进行运算操作时会出现错误,所以可给其设置默认值 0。

② 表的外键约束子句,格式为:

```
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。

例 3.4 大学教学管理数据库中的学习关系 SC,可重新用如下的表定义语句定义如下:

```
CREATE TABLE SC
   (S# CHAR(9),
    C# CHAR(7),
    GRADE SMALLINT DEFAULT(0),
    PRIMARY KEY(S#,C#),
    FOREIGN KEY(C#) REFERENCES C(C#));
```

其中,学习关系表 SC 的主键是由学号 S# 和课程号 C# 组成的合成主键,且课程号 C# 又被定义成学习关系表 SC 的外键,它在课程关系表 C 中是主键。

③ 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，而且还可以是一个包含 SELECT 的 SQL 语句。

例 3.5　大学教学管理数据库中的学习关系 SC，还可用如下的表定义语句定义如下：

```
CREATE TABLE SC
   (S# CHAR(9),
    C# CHAR(7),
    GRADE SMALLINT DEFAULT(0),
    PRIMARY KEY(S#,C#),
    FOREIGN KEY(C#) REFERENCES C(C#),
    CHECK(GRADE BETWEEN 0 AND 100));
```

最后需要指出的是，在 SQL 的交互式命令工作方式中，每一个 SQL 语句又可看作是一条 SQL 命令，所以有时称其为“语句”，有时称其为“命令”。

当一个新表生成时，它是一个没有数据的空关系，接下来的工作就是给它装入数据。数据的装入方式详见 3.2.2 节。

2. 表的修改

常常因为事先考虑不周需要对已经建立好的表进行修改。对表的修改包括改变表名、增加列、删除列、修改列的定义等。下面介绍利用 SQL 语句修改表的方法。

1）改变表名

修改表名的语句格式为：

```
RENAME ＜原表名＞ TO ＜新表名＞;
```

例 3.6　将 SS 表改名为 SS1。

```
RENAME SS TO SS1;
```

2）增加列

在表的最后一列后面增加新的一列，但不允许将一个列插入到原表的中间。增加列语句的格式为：

```
ALTER TABLE ＜表名＞ ADD ＜增加的列名＞ ＜数据类型＞;
```

其中，新增加的列后不能有可选项“[NOT NULL]”，也就是说，新增加的列不能定义为“NOT NULL”。表在增加了一个列后，原来的元组在新增加的属性上的值都被定义为空值。

例 3.7　给专业表 SS 增加一个新属性 NOUSE_COLUMN，设其数据类型为 DECIMAL(8,1)。

```
ALTER TABLE SS ADD NOUSE_COLUMN DECIMAL(8,1);
```

3）删除列

删除表中不再需要的列，语句格式为：

```
ALTER TABLE ＜表名＞ DROP ＜删除的列名＞ [CASCADE | RESTRICT];
```

其中，可选项“[CASCADE | RESTRICT]”是删除方式。当选择该选项时，由用户根据属性列的特点确定其中之一。CASCADE 表示在表“＜表名＞”中删除列“＜删除的列名＞”，所

有引用到该列的视图(视图操作详见 3.4 节)或有关约束也一起被删除；RESTRICT 表示在没有视图或有关约束引用列“<删除的属性列名>”时，关系中的该列才能被删除，否则拒绝该删除操作。

例 3.8 删除专业表 SS 中增加的属性 NOUSE_COLUMN 的两种删除语句分别为：

```
ALTER TABLE SS DROP NOUSE_COLUMN CASCADE;
ALTER TABLE SS DROP NOUSE_COLUMN RESTRICT;
```

4) 修改列的定义

修改属性列的定义语句只用于修改列的类型和长度，列的名称不能改变。当表中已有数据时，不能缩短列的长度，但可以增加列的长度。修改列定义语句格式为：

```
ALTER TABLE <表名> MODIFY <列名> <新的数据类型及其长度>;
```

例 3.9 修改专业表 SS 中的专业名称 SSNAME(30)为 SSNAME(40)，即长度增加 10。

```
ALTER TABLE SS MODIFY SSNAME VARCHAR(40);
```

3. 表的撤销

表的撤销就是将不再需要的表或定义有错误的表删除掉。当一个表被撤销时，该表中的数据也一同被撤销(删除)。撤销表的语句格式为：

```
DROP TABLE <表名> [CASCADE | RESTRICT];
```

其中，CASCADE 表示在撤销表“<表名>”时，所有引用这个表的视图或有关约束也一起被撤销；RESTRICT 表示在没有视图或有关约束引用该表的属性列时，表“<表名>”才能被撤销，否则拒绝该撤销操作。

3.2.2 数据的插入、修改与删除

1. 数据插入

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 语句格式为：

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

其中：

① <列名表>的格式为：<列名 1>[,<列名 2>,…,<列名 m>]。

② <值表>的格式为：<常量 1>[,<常量 2>,…,<常量 m>]，用于指出要插入列的具体值。

③ 如果选择可选项“[(<列名表>)]”，表示在插入一个新元组时，只向由<列名 i>指出的列中插入数据，其他没有列出的列不插入数据，且“<列名表>”中至少必须包括在表定

义中为 NOT NULL 的列和主键列；并且，＜值表＞中的属性列值必须与＜列名表＞中的属性列名一一对应。如果没有选择可选项[＜列名表＞]，则默认表中所有的列都要插入数据，并且＜值表＞中的列值必须与＜列名表＞中的属性列名一一对应。

例 3.10　给学习关系 SC 中插入王丽丽同学(学号为 200401003)学习计算机网络课(课程号为 C403001)的成绩(89 分)。

```
INSERT INTO SC
   (S#,C#,GRADE)
    VALUES('200401003','C403001',89);
```

其中，由于插入的元组中的属性列个数、顺序与学习关系 SC 的结构完全相同，所以可以略写可选项，也即上面的语句可简写为：

```
INSERT INTO SC
        VALUES('200401003','C403001',89);
```

例 3.11　如果在创建学习关系 SC 时已经把分数属性 GRADE 的值默认定义成 0，那么在学生的考试成绩出来之前就可输入学生的学号 S# 和课程号 C# 信息，等考试成绩出来后再通过修改表内容来输入成绩。

```
INSERT INTO SC
   (S#,C#)
    VALUES('200401003','C403001');
```

其中，由于 S# 和 C# 都是主键属性，所以必须插入这两个属性列的值。而且表名 SC 后的列名表(S#,C#)是不能省略的。

2. 数据修改

在 SQL 语言中，修改表中的数据是由 UPDATE 语句来实现的，其语句格式为：

```
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＞];
```

其中，“＜列名 i＞＝＜表达式 i＞”指出将列“＜列名 i ＞”的列值修改成＜表达式 i ＞。可选项“[WHERE ＜条件＞]”中的＜条件＞指定修改有关列的数据时所应满足的条件。当不选择该选项时，表示修改表中全部元组中相应列的数据。

例 3.12　将学生关系 S 中的学生名字“王丽丽”改为“王黎丽”。

```
UPDATE S
SET SNAME = '王黎丽'
WHERE S# = '200401003';
```

UPDATE 语句不仅可以修改一行数据，还可以同时修改多行数据。

例 3.13　将所有女同学的专业改为 S0404。

```
UPDATE S
SET SCODE# = 'S0404'
WHERE SSEX = '女';
```

3. 数据删除

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

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

例 3.14 在学生关系 S 中删除学号为 200403001 的学生的信息。

```
DELETE FROM S
WHERE S# = '200403001';
```

例 3.15 删除专业关系中的全部信息。

```
DELETE FROM SS;
```

3.3 SQL 的数据查询

SQL 语言中最重要和最核心的部分就是它的查询功能。查询语句根据其查询要求的不同具有多种不同的表示形式,所以查询语句也是 SQL 语言中最复杂的语句。

3.3.1 简单查询

考虑到学习上的循序渐进和便于理解,本节只介绍在查询过程中仅涉及一个表的查询。多表联接查询等高级查询技术将在 3.3.3 节介绍。

1. SELECT 查询语句

SQL 查询语句的基本格式为:

```
SELECT <列名表>
FROM <表名表>
[WHERE <条件>]
```

其中:

① 若设<列名表>为 $A_1,A_2,\cdots,A_n$;<表名表>为 $R_1,R_2,\cdots,R_m$,则上述查询语句的意义可表示成如下的关系代数表达式:

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

② 当不选择可选项"[WHERE <条件>]"时,SQL 查询语句为无条件查询。当选择可选项"[WHERE <条件>]"时,其中的<条件>用于指定查询所需数据必须满足的条件。

2. 无条件查询

当要查询表中的全部数据,或要查询表中某个或某些特定列上的全部数据时,就要将表中的全部行(元组)都选择出来,因而无须任何选择条件,这种查询就称为无条件查询。

例 3.16　查询教学管理数据库中全部学生的基本信息。

```
SELECT *
FROM  S;
```

其中,“*”表示表的全部列名。对于图 1.7 中给出的关系的当前值来说,查询结果为:

S#	SNAME	SSEX	SBIRTHIN	PLACEOFB	SCODE#	CLASS
200401001	张华	男	14-dec-82	北京	S0401	200401
200401002	李建平	男	20-aug-82	上海	S0401	200401
200401003	王丽丽	女	02-feb-83	上海	S0401	200401
200402001	杨秋红	女	09-may-83	西安	S0402	200402
200402002	吴志伟	男	30-jun-82	南京	S0402	200402
200402003	李涛	男	25-jun-83	西安	S0402	200402
200403001	赵晓艳	女	11-mar-82	长沙	S0403	200403

例 3.17　查询教学管理数据库中全部教师的教职工编号、姓名、职称和所属教研室。

```
SELECT T#,TNAME,TITLEOF,TRSECTION
FROM  T;
```

对于图 1.7 中给出的关系的当前值来说,本例的查询结果为 6 行 4 列(教师的教职工编号、姓名、职称和所属教研室)。

例 3.18　查询课程关系 C 中的记录数,也即开课的总门数。

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

其中,函数 COUNT(*)用于统计课程关系中的总记录数(元组个数)。这种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。在 SQL 语言中,常用的聚合函数如表 3.2 所示。

表 3.2　SQL 语言中的常用的聚合函数

聚合函数名称及格式	功 能 说 明
COUNT(*)	计算元组的个数
COUNT(列名)	计算某一列中数据的个数
COUNT DISTINCT(列名)	计算某一列中不同值的个数
SUM(列名)	计算某一数据列中值的总和
AVG(列名)	计算某一数据列中值的平均值
MIN(列名)	求(字符、日期、属性列)的最小值
MAX(列名)	求(字符、日期、属性列)的最大值

例 3.19　计算所有学生所学课程的最高分数、最低分数和平均分数。

```
SELECT MAX(GRADE),MIN(GRADE),AVG(GRADE)
FROM  SC;
```

3. 单条件查询

在 SQL 的查询语句中,更多的是有条件查询。单条件查询就是 WHERE 子句中只有一个条件表达式的查询,在条件表达式中可以使用的关系比较符如表 3.3 所示。

表 3.3 条件表达式中的关系比较符

运算符	含　义	运算符	含　义
=	等于	<	小于
!= 或 <>	不等于	<=	小于等于
>	大于	IS NULL	是空值
>=	大于等于	IS NOT NULL	不是空值

例 3.20 查询所有学习了计算机网络课(课程号为 C403001)的学生的学号和成绩。

```
SELECT S#,GRADE
FROM   SC
WHERE  C#  = 'C403001';
```

在上面的例子中,WHERE 子句中的条件是由唯一的条件表达式"C# ='C403001'"构成的。对于图 1.7 中给出的关系的当前值来说,查询结果为:

S#	GRADE
200401001	85
200402002	92
200402003	83

4. 多条件查询

多条件查询就是 WHERE 子句中有多个条件表达式的查询。在 SQL 语言中,WHERE 子句中的多个条件表达式是由如表 3.4 中的逻辑运算符将它们组合在一起构成查询条件的。

表 3.4 逻辑运算符

运算符	含　义
NOT	逻辑非
AND	逻辑与
OR	逻辑或

WHERE 子句中的优先级顺序为:关系比较符的优先级高于逻辑运算符;逻辑运算符的优先级从高到低依次为 NOT、AND、OR。

例 3.21 查询选修了计算机网络课(课程号为 C403001)或信息安全技术课(课程号为 C403002)的学生的学号。

```
SELECT S#
FROM   SC
WHERE  C#  = 'C403001' OR C#  = 'C403002';
```

例 3.22 若有学生关系 S1(S#,SNAME,SSEX,SAGE,PLACEOFB,SCODE,CLASS),即该学生关系中给出的不是出生年月,而是年龄。要求查询年龄在 21～25 岁之间学生的基本信息。

```
SELECT *
FROM   S1
WHERE  SAGE >= 21 AND SAGE <= 25;
```