学习情境 4 关系数据库标准语言——SQL

SQL 是高级的非过程化编程语言，是沟通数据库服务器和客户端的重要工具，允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以，具有完全不同底层结构的不同数据库系统可以使用相同的 SQL 语言作为数据输入与管理的 SQL 接口。它以记录集合作为操作对象，所有 SQL 语句接受集合作为输入，返回集合作为输出，这种集合特性允许一条 SQL 语句的输出作为另一条 SQL 语句的输入，所以 SQL 语句可以嵌套，这使它具有极大的灵活性和强大的功能，在多数情况下，在其他语言中需要一大段程序实现的功能只需要一个 SQL 语句就可以达到目的，这也意味着用 SQL 语言可以写出非常复杂的语句。

SQL 语言是关系数据库的国际标准语言，目前各大数据库厂家均有各自的 SQL 软件或与 SQL 的接口软件，几乎所有著名的数据库管理系统，例如 Oracle、DB2、SQL Server、Access 等都支持 SQL 语言。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互相操作有了共同的基础。

在本学习情境中，我们将详细分解 SQL 语言的核心知识点，了解 SQL 语言的产生和发展背景，掌握 SQL 数据定义（包括二维表的创建、修改、删除，视图的创建、删除，索引的创建、删除）、SQL 数据查询（包括单表查询和多表查询）、SQL 数据更新（包括插入数据、修改数据、删除数据）、SQL 数据控制（包括授权与删除权限）的语法格式，并用这些 SQL 语句完成相应的操作。

SQL 语言是关系数据库最为核心的技术之一。作为电子商务专业的学生，我们要掌握 SQL 语言的所有语法格式，这些内容将会在后续课程网页设计、网站设计中应用。

- SQL 语言的发展情况、功能、特点。
- SQL 操作基本表、视图、索引的语句格式。
- SQL 单表查询、多表查询的语句格式。
- SQL 数据更新的语句格式。
- SQL 数据控制的语句格式。

子学习情境一　认识 SQL 语言

任务一　SQL 语言的产生与发展

SQL（Structured Query Language）语言是 1974 年由 Boyce 和 Chamberlin 提出的。1975 年至 1979 年 IBM 公司 San Jose Research Laboratory 研制了著名的关系数据库管理系统原型 System R 并实现了这种语言。

1986 年 10 月美国国家标准局（American National Standard Institute，ANSI）的数据库委员会 X3H2 批准了 SQL 作为关系数据库语言的美国标准，同年公布了 SQL 标准文本（简称 SQL-86）。1987 年国际标准化组织（International Organization for Standardization，ISO）也通过了这一标准。此后 ANSI 不断修改和完善 SQL 标准，并于 1989 年公布了 SQI-89 标准，1992 年又公布了 SQL-92 标准，人们习惯上称之为 SQL2。1999 年 ISO 发布了标准化文件 ISO/IEC9075：数据库语言 SQL（1999），延续了 SQL 的叫法，人们习惯称之为 SQL3。

由于 SQL 语言功能丰富、语言简洁而备受业界欢迎，被众多公司所采用。经过各公司的不断修改、扩充和完善，SQL 语言最终发展成为关系数据库的标准语言。

自 SQL 成为国际标准语言以后，各个数据库厂家纷纷推出各自的 SQL 软件或与 SQL 的接口软件，目前几乎所有著名的数据库管理系统，例如 Oracle、DB2、SQL Server、Access 等都支持 SQL 语言。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互相操作有了共同的基础。

SQL 成为国际标准，对数据库以外的领域也产生了很大影响，有不少软件产品将 SQL 语言的数据查询功能与图形功能、软件工程工具、软件开发工具、人工智能程序结合起来。SQL 已成为数据库领域中的一个主流语言。

任务二　SQL 语言的特点

SQL 是一个通用的、功能极强的关系数据库语言。SQL 语言之所以能够为用户和业界所接受，并成为国际标准，是因为它是一个综合的、功能极强同时又简洁易学的语言。SQL 语言集数据查询（Data Query）、数据操纵（Data Manipulation）、数据定义（Data Definition）和数据控制（Data Control）功能于一体，主要特点包括 5 个方面：

（1）综合统一。

数据库系统的主要功能是通过数据库支持的数据语言来实现的。SQL 语言集数据定义语言 DDL、数据操纵语言 DML、数据控制语言 DCL 的功能于一体，语言风格统一，可以独立完成数据库生命周期中的全部活动，包括定义关系模式、插入数据建立数据库、查询、更新、维护、数据库重构、数据库安全性控制等一系列操作要求，这就为数据库应用系统的开发提供了良好的环境。此外，在关系模型中，实体和联系均用关系表示，这种单一的数据结构使得数据的查询和更新操作都只有一种操作符，克服了非关系系统由于信息表示方式的多样性带来的操作复杂性。

（2）高度非过程化。

非关系数据模型的数据操纵语言是面向过程的语言，用其完成某项请示必须指定存取路

径。而用 SQL 语言进行数据操作，只要提出“做什么”，而无须指明“怎么做”，因此无需了解存取路径，存取路径的选择以及 SQL 语句的操作过程由系统自动完成。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。例如查询所有平均成绩在 80 分以上的学生姓名，用户必须一条一条地把满足条件的学生记录找出来（通常要说明具体处理过程，即按照哪条路径、如何循环等）。而 SQL 语言采用集合操作方式，不仅操作对象、查找结果可以是元组的集合，而且一次插入、删除、更新操作的对象也可以是元组的集合。

（4）以同一种语法结构提供两种使用方式。

SQL 语言既是自含式语言，又是嵌入式语言。作为自含式语言，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键入 SQL 命令对数据库进行操作；作为嵌入式语言，SQL 语句能够嵌入到高级语言程序中，供程序员设计程序时使用。而在两种不同的使用方式下，SQL 语言的语法结构基本上是一致的。这种以统一的语法结构提供两种不同的使用方式的做法，提供了极大的灵活性与方便性。

（5）语言简洁，易学易用。

SQL 语言功能极强，但由于设计巧妙，语言十分简洁，完成核心功能只用了 9 个动词，如表 4-1 所示。

表 4-1　SQL 的命令动词

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

任务三　SQL 语言的功能

SQL 语言功能丰富，语法简洁，不仅能满足用户查询的需要，同时还拥有数据定义、更新和控制等功能。由于其功能强大且简单易学，使得 SQL 很快成为了关系数据库的标准语言，几乎所有关系型数据库软件产品都配备了 SQL 语言或提供有 SQL 接口。

SQL 的主要功能可以分为以下 4 类：

（1）数据定义功能。SQL 的数据定义功能通过 DDL（Data Definition Language，数据定义语言）实现，它用来定义关系数据库的模式、外模式和内模式，以实现对基本表、视图、索引文件的定义、修改和删除等操作。

（2）数据操纵功能。SQL 的数据操纵功能通过 DML（Data Manipulation Language，数据操纵语言）实现，用于增加、删除和修改数据。

（3）数据查询功能。SQL 的数据查询功能通过 QL（Query Language，数据查询语言）实现，用于查询数据（包括单表查询、多表查询及查询附加功能）。

（4）数据控制功能。数据控制指数据的安全性和完整性控制。SQL 的数据控制通过 DCL（Data Control Language，数据控制语言）实现。SQL 通过对数据库用户的授权和收权命令来实现有关数据的存取控制，以保证数据库的安全性。SQL 还提供了数据完整性约束条件的定义和检查机制，以保障数据库的完整性。

子学习情境二　SQL 的数据定义功能

关系数据库系统支持三级模式结构，其模式、外模式和内模式中的基本对象有表、视图和索引。因此，SQL 的数据定义功能包括定义表、定义视图和定义索引，如表 4-2 所示。

表 4-2　SQL 数据定义语句

操作对象	操作方法		
	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	

任务一　定义、删除和修改基本表

1. 定义基本表

建立数据库最重要的一步就是定义基本表。SQL 语言使用 CREATE TABLE 语句定义基本表，其一般格式如下：

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

其中<表名>是所要定义的基本表的名字，它可以由一个或多个属性（列）组成。定义表的各个属性时，需要指明其数据类型及长度。不同的数据库系统支持的数据类型不完全相同，比如 Access 的数据类型主要有文本、数字、日期、备注、逻辑型数据、货币、自动编号、OLE、超链接、查阅向导等。建表的同时通常还可以定义与该表有关的完整性约束条件，这些完整性约束条件被存入系统的数据字典中，当用户操作表中的数据时由 DBMS 自动检查该操作是否违背这些完整性约束条件。如果完整性约束条件涉及到该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级。

例 1　建立一个“学生”表 student，它由学号 sno、姓名 sname、性别 ssex、年龄 sage、所在系 sdept 五个属性组成。其中学号不能为空，值是唯一的，并且姓名取值也唯一。

```
CREAT TABLE student (sno    CHAR(5) Not Null UNIQUE,
                     sname    CHAR(20) UNIQUE,
                     ssex CHAR(1),
                     sage INT,
                     sdept CHAR(15));
```

该语句执行完后，系统中就会建立名称为 Student 的新表，并且表的定义及相关完整性约束条件会存放在数据字典中。

例 2 建立一个成绩表 score，它由学号 sno、课程编号 cno、课程名 cname 和成绩 grade 组成。其中，学号和课程编号不能为空。

```
CREAT TABLE score (sno CHAR(7) Not Null,
                   cno CHAR(6) Not Null,
                   cname CHAR(10) Not Null,
                   grade CHAR(3));
```

2. 修改基本表

随着应用环境和应用需求的变化，有时需要修改已建立好的基本表，SQL 语言用 ALTER TABLE 语句修改基本表，其一般格式为：

```
ALTER TABLE<表名> [ADD<新列名><数据类型>[完整性约束]]
                  [DROP<完整性约束名>]
                  [MODIFY<列名><数据类型>];
```

其中<表名>是要修改的基本表，ADD 子句用于增加新列和新的完整性约束条件，DROP 子句用于删除指定的完整性约束条件，MODIFY 子句用于修改原有的列定义，包括个性列名和数据类型。Access 不支持 MODIFY 子句，要修改列名及数据类型需要进行先删除后增加的操作。

例 3 向 student 表中增加身高 sheight 列，其数据类型为字符型。

```
ALTER TABLE student ADD sheight CHAR(5);
```

3. 删除基本表

当某个基本表不再需要时，可以使用 DROP TABLE 语句删除它。其一般格式为：

```
DROP TABLE<表名>
```

基本表定义一旦删除，表中的数据、此表上建立的索引和视图都将自动被删除掉。因此，执行删除基本表的操作一定要格外小心。

例 4 删除 student 表。

```
DROP TABLE student;
```

4. 补充定义主键

由于 SQL 并不要求每个表都定义主键，所以提供了一个补充定义主键的命令，在需要时定义主键。定义主键仍然用 ALTER TABLE 命令，其语法格式为：

```
ALTER TABLE 表名 ADD PRIMARY KEY(列名);
```

被定义为主键的列名必须满足取值非空且值唯一的条件。

例 5 将 student 表中的学号 sno 定义为主键。

```
ALTER TABLE student ADD PRIMARY KEY(sno);
```

5. 撤消主键定义

撤消主键定义也用 ALTER TABLE 命令，Access 软件中撤消主键定义的语法格式为：

```
ALTER TABLE 表名 DROP CONSTRAINT PRIMARYKEY;
```

被定义为主键的列名必须满足取值非空且值唯一的条件。

例 6 将 student 表中定义的主键撤消。

```
ALTER TABLE student DROP CONSTRAINT PRIMARYKEY;
```

6. 补充定义外键

在需要定义外键时，可以用 ALTER TABLE 命令，其语法格式为：

```
ALTER TABLE 表1
        ADD FOREIGN KEY[外键名]
        REFERENCES 表2
        [ON DELETE {RESTRICT|CASCADE|SET NULL}];
```

上述命令中，花括号中的三项需要任选一项，缺省情况下默认为 RESTRICT。

例 7　将 score 表中的学号 sno 定义为外键。

```
ALTER TABLE score
        ADD FOREIGN KEY(sno)
        REFERENCES student
        ON DELETE RESTRICT;
```

7. 撤消外键定义

在需要撤消外键时，可以用 ALTER TABLE 命令，其语法格式为：

```
ALTER TABLE 表名 DROP CONSTRAINT 外键名;
```

例 8　将 score 表中外键的定义撤消。

```
ALTER TABLE score DROP CONSTRAINT sno;
```

任务二　视图

视图是关系数据库系统提供给用户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）导出的表，它与基本表不同，是一个虚表。数据库中只存放视图的定义，而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，从视图中查询出的数据也就随之改变了。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及其变化。

视图一经定义，就可以和基本表一样被查询、被删除，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。

1. 建立视图

SQL 语言用 CREATE VIEW 命令建立视图，其一般格式为：

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

其中子查询可以是任意复杂的 SELECT 语句，但通常不允许含有 ORDER BY 子句和 DISTINCT 短语。

WITH CHECK OPTION 表示对视图进行 UPDATE、INSERT 和 DELETE 操作时要保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）。

组成视图的属性列名或者全部省略或者全部指定，没有第 3 种选择。如果省略了视图的各个属性列名，则隐含该视图由子查询中 SELECT 子句目标列中的诸字段组成。但在下列 3 种情况下必须明确指定组成视图的所有列名：

（1）某个目标列不是单纯的属性名，而是集函数或列表达式。

（2）多表连接时选出了几个同名列作为视图的字段。

（3）需要在视图中为某个列启用新的更合适的名字。

例 1 建立物流学院学生的视图。

```
CREATE VIEW ISstudent
        AS
        SELECT sno,sname,sage
        FROM   student
        WHERE sdept='WL';
```

本例中省略了视图 ISstudent 的列名，隐含了由子查询中 SELECT 子句中的三个列名组成。

DBMS 执行 CREATE VIEW 语句的结果只是把视图的定义存入数据字典，并不执行其中的 SELECT 语句。只是在对视图查询时，才按视图的定义从基本表中将数据查出。

例 2 建立物流学院学生的视图，并要求进行修改和插入操作时仍需保证该视图只有物流学院的学生。

```
CREATE VIEW ISstudent
        AS
        SELECT sno,sname,sage
        FROM   student
        WHERE sdept='WL'
        WITH CHECK OPTION;
```

由于在定义 ISstudent 视图时加上了 WITH CHECK OPTION 子句，以后对该视图进行插入、修改和删除操作时，DBMS 会自动加上 sdept＝'WL'的条件。

若一个视图是从单个基本表导出的，并且只是去掉了基本表的某些行和某些列，但保留了键，我们称这类视图为行列子集视图。ISstudent 视图就是一个行列子集视图。

视图不仅可以建立在单个基本表上，也可以建立在多个基本表上。

例 3 建立物流学院选修了 1 号课程的学生的视图。

```
CREATE VIEW ISS1(sno,sname,grade)
        AS
        SELECT student.sno,sname,grade
        FROM   student,score
        WHERE sdept='WL' AND   student.sno=score.sno   AND    score.cno=1;
```

由于视图 ISS1 的属性列中包含了 student 表与 score 表的同名列 sno，所以必须在视图名后面明确说明视图的各个属性列名。

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定义好的视图上，或建立在基本表与视图上。

例 4 建立物流学院选修了 1 号课程且成绩在 90 分以上的学生的视图。

```
CREATE VIEW ISS2
        AS
        SELECT sno,sname,grade
        FROM   ISS1
        WHERE grade>=90;
```

这里的视图 ISS2 就是建立在视图 ISS1 之上的。

定义基本表时，为了减少数据库中的冗余数据，表中只存放基本数据，由基本数据经过各种计算派生出的数据一般是不存储的。但由于视图中的数据并不实际存储，所以定义视图时可以根据应用的需要设置一些派生属性列。这些派生属性由于在基本表中并不实际存在也称它

们为虚拟列。带虚拟列的视图也称为带表达式的视图。

2. 删除视图

删除视图语句的格式为：

```
DROP VIEW <视图名>;
```

视图删除后，视图的定义将从数据字典中删除。但由该视图导出的其他视图定义仍在数据字典中，不过该视图已失效。用户使用时会出错，要用 DROP VIEW 语句将它们一一删除。

例 5　删除视图 ISS1。

```
DROP VIEW ISS1;
```

执行此语句后，ISS1 视图的定义将从数据字典中删除。由 ISS1 视图导出 ISS2 视图的定义虽然仍在数据字典中，但是该视图已无法使用了，因此应该同时删除。

3. 查询视图

视图定义后，用户就可以像对基本表一样对视图进行查询了。

例 6　在物流学院的学生视图中找出年龄小于 20 岁的学生。

```
SELECT    sno,sage
FROM    ISstudent
WHERE    sage<20;
```

DBMS 执行对视图的查询时，首先进行有效性检查，检查查询的表、视图等是否存在。如果存在，则从数据字典中取出视图的定义，把定义中的子查询和用户的查询结合起来转换成等价的对基本表的查询，然后再执行修正了的查询。这一转换过程称为视图消解（View Resolution）。

在一般情况下，视图查询的转换是直截了当的。但有些情况下，这种转换不能直接进行，查询时就会出现问题。目前多数关系数据库系统对行列子集视图的查询均能进行正确转换。但对非行列子集的查询就不一定能做转换了，因此这类查询应该直接对基本表进行。

4. 更新视图

更新视图是指通过视图来插入（INSERT）、删除（DELETE）和修改（UPDATE）数据。由于视图是不实际存储数据的虚表，因此对视图的更新最终要转换为对基本表的更新。

在关系数据库中，并不是所有的视图都是可更新的，因为有些视图的更新不能唯一地、有意义地转换成对相应基本表的更新。

5. 视图的作用

视图最终是定义在基本表之上的，对视图的一切操作最终也要转换为对基本表的操作。

（1）视图能够简化用户的操作视图机制，使用户可以将注意力集中在所关心的数据上。

（2）视图使用户能以多种角度看待同一数据视图机制，能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户共享同一个数据库时，这种灵活性是非常重要的。

（3）视图为重构数据库提供了一定程度的逻辑独立性。

（4）视图能够对机密数据提供安全保护。

任务三　索引

建立索引是加快查询速度的有效手段。用户可以根据应用环境的需要，在基本表上建立一个或多个索引，以提供多种存取路径，加快查找速度。一般说来，建立与删除索引由数据库

管理员 DBA 或表的属主（即建立表的人）负责完成。系统在存取数据时会自动选择合适的索引作为存取路径，用户不必也不能选择索引。

1. 建立索引

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

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

其中，<表名>是要建立索引的基本表的名字。索引可以建立在该表的一列或多列上，各列名之间用逗号分隔。<列名>后面还可以用<次序>指定索引值的排列次序，可选 ASC（升序）或 DESC（降序），默认值为 ASC。

UNIQUE 表明此索引的每一个索引值只对应唯一的数据记录。

CLUSTER 表示要建立的索引是聚簇索引，即索引项的顺序与表中记录的物理顺序一致的索引组织。

例 1 为学生表 student 建立一个索引。

```
CREATE UNIQUE INDEX stusno ON student(sno);
```

2. 删除索引

索引一经建立，就由系统使用和维护它，不需要用户干预。建立索引是为了节省查询操作的时间，但如果数据增加删改频繁，系统会花费许多时间来维护索引。这时，可以删除一些不必要的索引。

在 SQL 语言中，Access 删除索引使用 DROP INDEX 语句，其一般格式为：

```
DROP INDEX<索引名> ON 表名;
```

例 2 删除 student 表中的 stusno 索引。

```
DROP INDEX stusno ON student;
```

删除索引时，系统会同时从数据字典中删去有关该索引的描述。

子学习情境三 SQL 数据查询

数据库查询是数据库的核心操作。SQL 语言提供了 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。其一般格式为：

```
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<条件表达式>]]
ORDER BY<列名 2>[ASC|DESC];
```

（1）SELECT 子句。该子句用于指明查询结果集的目标列。目标列可以是直接从数据源中投影得到的字段、与字段相关的表达式或数据统计的函数表达式，目标列还可以是常量。如果目标列中使用了两个基本表（或视图）中相同的列名，那么就要在列名前加表名限定，即使用“(表名).(列名)”表示。

（2）FROM 子句。该子句用于指明查询的数据源。查询操作需要的数据源指基本表（或视图表）组，表间用“,”分隔。如果查询使用的基本表或视图不在当前数据库中，则还需要在表或视图前加上数据库名加以说明，即使用“(数据库名).(表名)”的形式表示。如果在查询

中需要一表多用，则每种使用都需要一个表的别名标识，并在各自使用中用不同的表别名表示。定义表别名的格式为“(表名)(别名)”。

（3）WHERE 子句。该子句通过条件表达式描述关系中元组的选择条件。DBMS 处理语句时，以元组为单位，逐个考察每个元组是否满足条件，将不满足条件的元组筛选掉。

（4）GROUP BY 子句。该子句的作用是按分组列的值对结果集分组。分组可以使同组的元组集中在一起，也使数据能够分组统计。当 SELECT 子句后的目标列中有统计函数时，如果查询语句中有分组子句，则统计为分组统计，否则为对整个结果集的统计。GROUP BY 子句后可以带上 HAVING 子句表达组选择条件，组选择条件为带有函数的条件表达式，它决定着整个组记录的取舍条件。

（5）ORDER BY 子句。该子句的作用是对结果集进行排序。查询结果集可以按多个排序列进行排序，每个排序列后都可以跟一个排序要求：当排序要求为 ASC 时，元组按排序列值的升序排序；排序要求为 DESC 时，结果集的元组按排序列值的降序排列。

整个 SELECT 语句的含义是，根据 WHERE 子句的条件表达式，从 FROM 子句指定的基本表或视图中找出满足条件的元组，再按 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 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。如果 GROUP 子句带 HAVING 短语，则只有满足指定条件的组才予以输出。如果有 ORDER 子句，则结果表还要按<列名 2>的值的升序或降序排序。

SELECT 语句既可以完成简单的单表查询，也可以完成复杂的连接查询和嵌套查询。在 SQL 的查询操作中，我们用如表 4-3 至表 4-5 所示的学生表 student、课程表 course、成绩表 score 为例来说明。

表 4-3　student 表的相关资料

sno	sname	ssex	sbirthday	sheight	shome
3030101	王平	男	1988-5-17	170	西安
3030102	李力	男	1987-8-5	178	咸阳
3030103	周福	女	1988-9-12	160	铜川
3030104	李宁	女	1989-12-26	165	安康
3030105	赵胜荣	男	1988-11-25	172	榆林
3030106	周勇毅	男	1989-1-16	185	广州

表 4-4　course 表的相关资料

cno	cname	chour	credit
A00001	电子商务	64	4
A00002	网络技术	32	2
A00003	数据库技术	64	4
A00004	网页设计	64	4
A00005	英语	32	2

表 4-5　score 表的相关资料

sno	cno	cname	grade
3030101	A00001	电子商务	91
3030101	A00003	数据库技术	78
3030102	A00005	英语	60
3030103	A00004	网页设计	88
3030103	A00002	网络技术	95
3030105	A00002	网络技术	92
3030104	A00002	网络技术	66
3030104	A00001	电子商务	85
3030106	A00003	数据库技术	93
3030106	A00004	网页设计	70
3030106	A00005	英语	52

任务一　SQL 语句的单表查询操作

1．无查询条件的单表查询

例 1　查询 student、course 和 score 三个表的全部内容。

```
SELECT * FROM student;
SELECT * FROM course;
SELECT * FROM score;
```

例 2　查询所有学生的学号、姓名和籍贯。

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

例 3　查询选修了课程的学生的学号，并去掉重复的元组。

```
SELECT DISTINCT sno
        FROM score;
```

当用户需要查询指定的元组时，可以通过 WHERE 子句实现。WHERE 子句常用的查询条件如表 4-6 所示。

表 4-6　常用的查询条件

查询条件	谓词	功能
比较	= > < >= <= !=	用于比较运算
确定范围	BETWEEN AND NOT BETWEEN AND	用于查找属性值在或不在指定范围内的元组，其中 BETWEEN 后是范围下限，AND 后是范围上限

续表

查询条件	谓词	功能
确定集合	IN NOT IN	用于查找属性值属于或不属于指定集合的元组
字符匹配	LIKE NOT LIKE	用于进行字符匹配，其含义是查找指定的属性值列与“匹配串”相匹配或不相匹配的元组。“匹配串”可以是一个完整的字符串，也可以含有通配符*（星号，在 Access 中代表任意长度的字符串）和?（问号，在 Access 中代表任意单个字符）
空值	IS NULL IS NOT NULL	用于涉及空值的查询
多重条件	AND OR	用于连接多个查询条件，AND 表示多个条件必须同时满足，OR 表示只需满足多个条件中的一个

2. 比较运算的单表查询

例 4　查询所有女生的姓名和身高。

```
SELECT sname,sheight
        FROM student
        WHERE ssex='女';
```

例 5　查询所有有考试成绩不及格的学生的学号。

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

3. 确定范围的单表查询

例 6　查询考试成绩在 60～70 分之间的学生的学号和课程名称。

```
SELECT sno,cname
        FROM score
        WHERE grade BETWEEN 60 AND 70;
```

4. 确定集合的单表查询

例 7　查询籍贯是西安、咸阳、铜川的学生的学号和姓名。

```
SELECT sno,sname
        FROM student
        WHERE shome IN('西安','咸阳','铜川');
```

5. 字符匹配的单表查询

例 8　查询所有姓周的学生的学号和姓名。

```
SELECT sno,sname
        FROM student
        WHERE sname LIKE'周*';
```

例 9　查询所有姓周且名字为三个字的学生的学号和姓名。

```
SELECT sno,sname
        FROM student
        WHERE sname LIKE'周??';
```

6. 涉及空值的单表查询

例 10　查询所有有考试成绩的学生的学号、课程名称和成绩。

```
SELECT sno,cname,grade
        FROM score
        WHERE grade IS NOT NULL;
```

7. 多重条件的单表查询

例 11 查询所有选修了数据库技术这门课且考试成绩在 70 分以上的学生的学号。

```
SELECT sno
        FROM score
        WHERE cname='数据库技术' AND grade>70;
```

8. 对查询结果进行排序

例 12 查询选修了电子商务这门课的学生的学号、课程名称及成绩，查询结果按分数的降序排列。

```
SELECT sno,cname,grade
        FROM score
        WHERE cname='电子商务'
        ORDER BY grade DESC;
```

任务二 SQL 语句的复杂查询操作

SQL 语句的复杂查询操作主要有连接查询、嵌套查询、集合查询 3 种。

1. 连接查询

若一个查询同时涉及两个以上的基本表，则称之为连接查询。连接查询是关系数据库中最主要的查询。为了避免属性名出现混淆，在连接查询的查询表达式中，属性名前可加上表名前缀。

例 1 查询所有学生的基本信息及其选修课程的课程名及成绩。

```
SELECT student.*,score.cname,score.grade
        FROM student,score
        WHERE student.sno=score.sno;
```

其查询结果如表 4-7 所示。

表 4-7 两个基本表的连接查询结果

sno	sname	ssex	sbirthday	sheight	shome	cname	grade
3030101	王平	男	1988-5-17	170	西安	电子商务	91
3030101	王平	男	1988-5-17	170	西安	数据库技术	78
3030102	李力	男	1987-8-5	178	咸阳	英语	60
3030103	周福	女	1988-9-12	160	铜川	网页设计	88
3030103	周福	女	1988-9-12	160	铜川	网络技术	95
3030105	赵胜荣	男	1988-11-25	172	榆林	网络技术	92
3030104	李宁	女	1989-12-26	165	安康	网络技术	66
3030104	李宁	女	1989-12-26	165	安康	电子商务	85
3030106	周勇毅	男	1989-1-16	185	广州	数据库技术	93
3030106	周勇毅	男	1989-1-16	185	广州	网页设计	70
3030106	周勇毅	男	1989-1-16	185	广州	英语	52

例 2　查询每个学生的学号、姓名、选修的课程名、学时及成绩。

```
SELECT student.sno,student.sname,course.cname,course.chour,score.grade
        FROM student,course,score
        WHERE student.sno=score.sno AND score.cno=course.sno;
```

其查询结果如表 4-8 所示。

表 4-8　3 个基本表的连接查询结果

sno	sname	cname	chour	grade
3030101	王平	电子商务	64	91
3030101	王平	数据库技术	64	78
3030102	李力	英语	32	60
3030103	周福	网页设计	64	88
3030103	周福	网络技术	32	95
3030105	赵胜荣	网络技术	32	92
3030104	李宁	网络技术	32	66
3030104	李宁	电子商务	64	85
3030106	周勇毅	数据库技术	64	93
3030106	周勇毅	网页设计	64	70
3030106	周勇毅	英语	32	52

2. 嵌套查询

在 SQL 语言中，一个 SELECT-FROM-WHERE 语句称为一个查询块。将一个查询块嵌套在另一个查询块的 WHERE 子句 HAVING 短语的条件中的查询称为嵌套查询。上层的查询块称为外层查询或父查询，下层的查询块称为内层查询或子查询。SQL 语言允许多层嵌套，但规定子查询的 SELECT 语句中不能使用 ORDER BY 子句。

例 3　查询选修了数据库技术课的学生的学号和姓名。

```
SELECT sno, sname
        FROM student
            WHERE sno IN
                (SELECT sno
                    FROM score
                    WHERE cname='数据库技术');
```

其查询结果如表 4-9 所示。

表 4-9　嵌套查询结果

sno	sname
3030101	王平
3030106	周勇毅

3. 集合查询

SELECT 语句的查询结果是元组的集合，所以多个 SELECT 语句的结果可进行集合操作。

集合操作主要包括并操作 UNION、交操作 INTERSECT 和差操作 MINUS。

例 4 查询选修了电子商务和数据库技术课的学生的学号和课程名。

```
SELECT sno,cname
        FROM score
            WHERE cname='电子商务'
            UNION
SELECT sno,cname
        FROM score
        WHERE cname='数据库技术';
```

其查询结果如表 4-10 所示。

表 4-10 集合查询结果

sno	cname
3030101	电子商务
3030104	电子商务
3030101	数据库技术
3030106	数据库技术

子学习情境四 SQL 数据更新

SQL 中数据更新包括插入数据、修改数据和删除数据 3 条语句。关系数据库软件一般都能自动进行完整性约束检查。当删除主表中的某个元组时，系统可以自动地删除参照表中相应的元组，或者，系统检查参照表中是否存在相应的元组，若有则操作失败。

任务一 插入数据

SQL 的数据插入语句 INSERT 通常有两种形式：一种是插入一个元组；另一种是插入子查询结果。后者可以一次插入多个元组。

1. 插入单个元组

插入单个元组的 INSERT 语句的格式为：

```
INSERT   INTO<表名>[<属性列 1>[,<属性列 2>…]]
          VALUES(<常量 1>[,<常量 2>…]);
```

其功能是将新元组插入指定表中。其中新记录属性列 1 的值为常量 1，属性列 2 的值为常量 2，……。INTO 子句中没有出现的属性列，新记录在这些列上将取空值。

但必须注意的是，在表定义时说明了 NOT NULL 的属性列不能取空值，否则会出错。

如果 INTO 子句中没有指明任何列名，则新插入的记录必须在每个属性列上均有值。

例 1 将一个新学生记录（学号：3030107；姓名：陈东；性别：男；生日：1989-2-13；身高：176；年龄：18 岁）插入到 student 表中。

```
INSERT
INTO student
VALUES('3030107', '陈东', '男', '1989-2-13', '176','18');
```

例 2 插入一条考试成绩记录（'3030105','数据库技术','60'）。

```
INSERT
    INTO score('sno','cname','grade')
VALUES('3030105','数据库技术', '60');
```

2. 插入子查询结果

子查询不仅可以嵌套在 SELECT 语句中，用以构造父查询的条件，也可以嵌套在 INSERT 语句中，用以生成要插入的批量数据。

插入子查询结果的 INSERT 语句的格式为：

```
INSERT  INTO<表名>[<属性列 1>[ ,<属性列 2>…]]
```

例 3　把 student 表中的 sno>3030103 的所有行复制并且添加到表 stu1 中，stu1 的列包括学号、姓名、性别、籍贯。

首先在数据库中创建 stu1 表：

```
CREAT TABLE stu1 (sno   CHAR(5) Not Null UNIQUE,
                  sname   CHAR(20) UNIQUE,
                  ssex CHAR(1),
                  shome CHAR(10));
```

然后将 student 表中满足条件的元组存入 stu1 中：

```
INSERT INTO stu1
SELECT sno,sname,ssex,shome
        FROM student
        WHERE sno>3030103;
```

任务二　修改数据

修改数据使用 UPDATE 语句，可以通过指定修改条件来修改表中的一行或多行，也可以使用游标进行定位修改。

修改数据语句的一般格式为：

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

其功能是修改指定表中满足 WHERE 子句条件的元组。其中 SET 子句给出<表达式>的值，用于取代相应的属性列值。如果省略 WHERE 子句，则表示要修改表中的所有元组。

例 1　对 student 表中 sno 为 3030106 的元组修改其中两个属性的取值，将生日改为 1988-2-7，将身高改为 180。

```
UPDATE student
    SET sbrithday=' 1988-2-7',sheight=180
    WHERE sno=3030106;
```

例 2　对 course 表中的所有课时加 2。

```
UPDATE course
    SET chour = chour + 2;
```

任务三　删除数据

删除数据使用 DELETE 语句，可以通过指定选取元组的条件删除表中的一行或多行，也可以使用游标进行定位删除。删除一行后，该行将无法再恢复，同时也无法再查看。

删除语句的一般格式为：

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

DELETE 语句的功能是从指定表中删除满足 WHERE 子句条件的所有元组。如果省略 WHERE 子句，表示删除表中的全部元组，但表的定义仍在数据字典中。也就是说，DELETE 语句删除的是表中的数据，而不是关于表的定义。

例 1 删除 student 表中 sno 为 3030104 的元组。

```
DELETE
    FROM student
    WHERE sno=3030104;
```

例 2 删除 student 表中的所有记录。

```
DELETE
    FROM student;
```

该语句执行的结果是将 student 表中的所有元组删除，但 student 表的定义仍然存在。

子学习情境五 SQL 数据控制

由 DBMS 提供统一的数据控制功能是数据库系统的特点之一。SQL 中数据控制功能包括事务管理功能和数据保护功能，即数据库的恢复、并发控制、数据库的安全性和完整性控制。

SQL 语言定义完整性约束条件的功能主要体现在 CREATE TABLE 语句和 ALTER TABLE 语句中，可以在这些语句中声明码、取值唯一的列、不允许空值的列、外码（参照完整性）及其他一些约束条件。

DBMS 必须具有以下功能：

（1）把授权的决定告知系统，这是由 SQL 的 GRANT 和 REVOKE 语句来完成的。

（2）把授权的结果存入数据字典中。

（3）当用户提出操作请求时，根据授权情况进行检查，以决定是否执行操作请求。

任务一 授权

SQL 语言用 GRANT 语句向用户授予操作权限，GRANT 语句的一般格式为：

```
GRANT<权限>[,<权限>]…
    [ON<对象类型><对象名>]
    TO<用户>[,<用户>]…
    [WITH GRANT OPTION];
```

其语义为：将对指定操作对象的指定操作权限授予指定的用户。

对属性列和视图的操作权限有：查询（SELECT）、插入（INSERT）、修改（UPDATE）、删除（DELETE）以及这 4 种权限的总和（ALL PRIVILEGES）。

例 1 把查询 student 表的权限授予用户 U1。

```
GRANT SELECT
    ON TABLE student
    TO U1;
```

例 2　把修改 student 表学生学号的权限授予用户 U2。

```
GRANT UPDATE(sno)
    ON TABLE student
    TO U2;
```

对基本表的操作权限有：查询（SELECT）、插入（INSERT）、修改（UPDATE）、删除（DELETE）、修改表（ALTER）、建立索引（INDEX）以及这 6 种权限的总和（ALL PRIVILEGES）。

对数据库可以有建立表（CREATE TABLE）的权限，该权限属于 DBA，可由 DBA 授予普通用户。普通用户拥有此权限后可以建立基本表，基本表的属主（Owner）拥有对该表的一切操作权限。

任务二　收回权限

授予的权限可以由 DBA 或其他授权者用 REVOKE 语句收回，REVOKE 语句的一般格式为：

```
REVOKE<权限>[,<权限>]…
      [ON<对象类型><对象名>]
      FROM<用户>[,<用户>]…;
```

例 1　收回用户 U2 修改学生学号的权限。

```
REVOKE UPDATE(sno)
    ON TABLE student
    FROM U2;
```

DBA 拥有对数据库中所有对象的所有权限，并可以根据应用的需要将不同的权限授予不同的用户。

用户对自己建立的基本表和视图拥有全部的操作权限，并且可以用 GRANT 语句把其中某些权限授予其他用户。被授权的用户如果有“继续授权”的许可，还可以把获得的权限再授予其他用户。

所有授予出去的权限在必要时又都可以用 REVOKE 语句收回。

拓展知识　嵌入式 SQL

SQL 有两种方式：一种是独立式 SQL；另一种是嵌入式 SQL。独立式 SQL 作为独立语言，在终端上以交互式方式使用；嵌入式 SQL 使用时嵌入到某种高级语言中，与高级语言混合使用。

嵌入 SQL 的高级语言称为主语言或宿主语言。使用嵌入式 SQL 和主语言相配合设计的应用程序，利用了主语言的过程性结构和专业应用功能强的优点，保留了 SQL 强大的数据库管理功能，两者相结合，功能更加完善和实用。

一、嵌入式 SQL 的特点

SQL 的功能只包括数据定义功能 DDL、数据操纵功能 DML 和数据控制语言 DCL，而缺少程序设计必要的程序流程控制和交互式功能，也缺少一些专业应用的功能，例如 SQL 没有分支、循环、赋值等语句等。因此，在实际应用中 SQL 经常需要和主语言配合使用才能完成

各种复杂的处理操作。

SQL 嵌入主语言时必须解决以下 3 个问题：

（1）区别 SQL 和主语言。

在嵌入式 SQL 中，为了能够区分 SQL 语句与主语言语句，必须在所有的 SQL 语句前面加上前缀 EXEC SQL。SQL 语句的结束标志则随主语言的不同而不同。

（2）使数据库的工作单元与程序工作单元之间能够通信。

在含有嵌入式 SQL 的应用程序中，SQL 语句负责管理数据库，主语言语句负责控制程序流程和其他功能。数据库的工作单元和程序工作单元之间通信的主要方式有以下两种：

- 主语言通过主变量向 SQL 语句提供参数。
- SQL 语句的当前工作状态和运行环境数据要反馈给应用程序。

（3）使用游标解决 SQL 一次一集合的操作与主语言一次一记录的操作的矛盾。

SQL 语言与主语言具有不同的数据处理方式。SQL 语言是面向集合的，一条 SQL 语句原则上可以产生或处理多条记录。而主语言是面向记录的，一组主变量一次只能存放一条记录。所以仅使用主变量并不能完全满足 SQL 语句向应用程序输出数据的要求，为此，嵌入式 SQL 引入了游标概念，用游标来协调这两种不同的处理方式。

游标是系统为用户开设的一个数据缓冲区，存放 SQL 语句的执行结果。每个游标区都有一个名字。用户可以通过游标逐一获取记录，并将记录赋给主变量，交给主语言做进一步处理。

二、不用游标的 SQL 语句

嵌入式 SQL 即使是不需要游标的语句，其语句格式和功能特点也与独立式 SQL 不同。下面介绍几种不使用游标的 SQL 语句及其特点和使用方法。

1. 几种不需要使用游标的 SQL 语句

下面 4 种 SQL 语句不需要使用游标：

（1）用于说明主变量的说明性语句。SQL 的说明性语句主要有两条：

```
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL END DECLARE SECTION;
```

这两条语句必须配对出现，两条语句中间是主变量的说明。由于说明性语句与数据记录无关，所以不需要使用游标。

（2）数据定义和数据控制语句。数据定义和数据控制语句在执行时不需要返回结果，也不需要使用主变量，因而也就不需要使用游标。

（3）查询结果为单记录的查询语句。如果在操作前明确知道查询结果为单记录，主语句可一次将查询结果读完，不需要使用游标。

（4）数据的插入语句和某些数据删除、修改语句。对于数据插入语句，即使插入批量数据，也只是在数据库工作区内部进行，不需要主语言介入，故不使用游标。

数据删除和修改语句分两种情况：独立的数据删除和修改语句不需要使用游标：与查询语句配合，删除或修改查询到的当前记录（在更新语句中，WHERE 的条件中使用 CURRENTOF{游标名}）的操作，与游标有关。

2. 不用游标的查询语句

不用游标的查询语句的一般格式为：

```
EXEC SQL SELECT[ALL|DISTINCT]<目标表达式>[,<目标表达式>]…
```

INTO<主变量>[<指示变量>][,<主变量>[<指示变量>]]
FROM<表名或视图名>[,…n]
[WHERE<条件表达式>];

其中：

（1）在语句开始前要加 EXEC SQL 前缀，这也是所有嵌入式 SQL 语言都必须加的前缀。

（2）该查询语句中又扩充了 INTO 子句，该子句的作用是把从数据库中找到的符合条件的记录放到 INTO 子句指定的主变量中去。

（3）在 WHERE 子句的条件表达式中可以使用主变量。

（4）由于查询的结果集中只有一条记录，故该 SELECT 语句中不必有排序和分组子句。

3．不用游标的数据维护语句

（1）不用游标的数据删除语句。在删除语句中，WHERE 子句的条件中可以使用主变量。

（2）不用游标的数据修改语句。在 UPDATE 语句中，SET 子句和 WHERE 子句中均可以使用主变量。SET 子句中的主变量可以使用指示变量，当指示变量的值是负值时，无论它前面的主变量是什么，都会使它所在的表达式值成为空值。

（3）不用游标的数据插入语句。INSERT 语句的 VALUES 子句可以使用主变量和指示变量，当需要插入空值时，可以把指示变量置为负值。

三、使用游标的 SQL

游标机制用于解决 SQL 查询结果为集合而主语言处理方式为记录方式的矛盾。在处理中，必须使用游标的 SQL 语句有两种：一种是查询结果为多条记录的 SELECT 语句；另一种是使用游标的 DELETE 语句和 UPDATE 语句。

1．定义游标命令

游标通过 DECLARE 语句定义，其语句格式为：

EXEC SQL DECLARE <游标名> CURSOR
FOR<子查询>
[FOR UPDATE OF{字段名 1}{,…n}];

定义游标语句仅仅是一条说明性语句。游标在定义时 DBMS 并不执行其子查询，只是将其定义内容记录下来，待打开游标时才按它的定义执行子查询。

在利用游标的删除和修改数据的语句中，WHERE 子句应表达为：

WHERE CURRENT OF <游标名>

2．打开游标命令

游标通过 OPEN 命令打开，其语句格式为：

EXEC SQL OPEN<游标名>;

OPEN 语句的作用是执行游标对应的查询语句，并将游标指向结果集的第一条记录前。打开的游标处于活动状态，可以被推进。但由于游标指向的是第一条记录前，所以还不能读出结果集中的数据。

3．推进游标命令

游标通过 FETCH 命令向前（或称向下）推进一条记录。推进游标的语句格式为：

EXEC SQL FETCH <游标名> INTO{主变量组};

推进游标的作用是将游标下移一行，读出当前的记录，将当前记录的各数据项值放到 INTO 后的主变量组中。

SQL 的游标在使用时只能向前推进，不能后退。如果需要后退游标，就需要执行关闭该游标、再重新打开、逐步推进游标到指定的位置等一系列操作。

4. 关闭游标命令

由于许多系统允许打开的游标数有一定的限制，所以当数据处理完后应及时把不使用的游标关闭，以释放结果集占用的缓冲区及其他资源。

关闭游标使用 CLOSE 命令，其格式为：

EXEC SQL CLOSE<游标名>;

基础训练

1．名词解释：

SQL 语言　索引　视图

2．简述 SQL 数据库体系结构的特点。

3．总结基本表与视图的共同点和不同点。

4．简要描述关系数据库标准语言 SQL 的功能。

技能训练

1．创建如下表，并输入适当的元组：

教师信息表，属性包括：职工编号、姓名、性别、年龄、学历、职称和专业。

教师工资表，属性包括：职工编号、基本工资、奖励津贴。

2．用 SQL 语句实现对教师表的单表查询操作。

（1）找出全体女教师。

（2）查询具有硕士和博士学位的教师的职工编号和姓名。

（3）查询具有教授职称且具有博士学位的教师的姓名。

3．用 SQL 语句找出具有博士学位的教师的基本工资。

4．用 SQL 语句对教师表实现以下操作：

（1）插入一条教师记录，其属性取值为，职工编号：9966，姓名：周文；性别：男；年龄：36；学历：博士；职称：副教授；专业：电子商务。

（2）将以上新插入元组的专业改为“物流管理”，职称改为“教授”。

（3）将以上记录从教师表中删除。