第3章 关系数据库标准语言SQL

本章内容

3.1 SQL语言的基本概念与特点

返回本章首页

3.1.1 语言的发展及标准化

- 在70年代初，E.F.Codd首先提出了关系模型。70年代中期，IBM公司在研制 SYSTEM R关系数据库管理系统中研制了SQL语言，最早的SQL语言（叫SEQUEL2）是在1976 年 11 月的IBM Journal of R&D上公布的。
- 1979年ORACLE公司首先提供商用的SQL，IBM公司在DB2 和SQL/DS数据库系统中也实现了SQL。
- 1986年10月，美国ANSI采用SQL作为关系数据库管理系统的标准语言(ANSI X3. 135-1986），后为国际标准化组织（ISO）采纳为国际标准。

返回本节首页

3.1.1 语言的发展及标准化

- 1989年，美国ANSI采纳在ANSI X3.135-1989报告中定义的关系数据库管理系统的SQL标准语言，称为ANSI SQL 89。
- 1992年，ISO又推出了SQL92标准，也称为SQL2.
- 1999年，推出了SQL：1999，也称SQL3，增加了面向对象等的功能。
- 2003年，推出了SQL：2003，增加了XML相关的特性等新功能。
- 2006年，又推出了SQL：2006，全面加强了对XML数据的处理与操作能力。
- 2008年，推出了SQL：2008正在起草中。

返回本节首页

3.1.1 语言的发展及标准化

- 结构化查询语言SQL(Structured Query Language)是一种介于关系代数与关系演算之间的语言，其功能包括查询、操纵、定义和控制四个方面，是一个通用的、功能极强的关系数据库语言。目前已成为关系数据库的标准语言，广泛应用于各种数据库。

返回本节首页

3.1.2 SQL语言的基本概念

关系数据库三级模式结构

- SQL语言支持关系数据库三级模式结构，如图3.1所示。其中外模式对应于视图（View）和部分基本表（Base Table），模式对应于基本表，内模式对应于存储文件。

返回本节首页

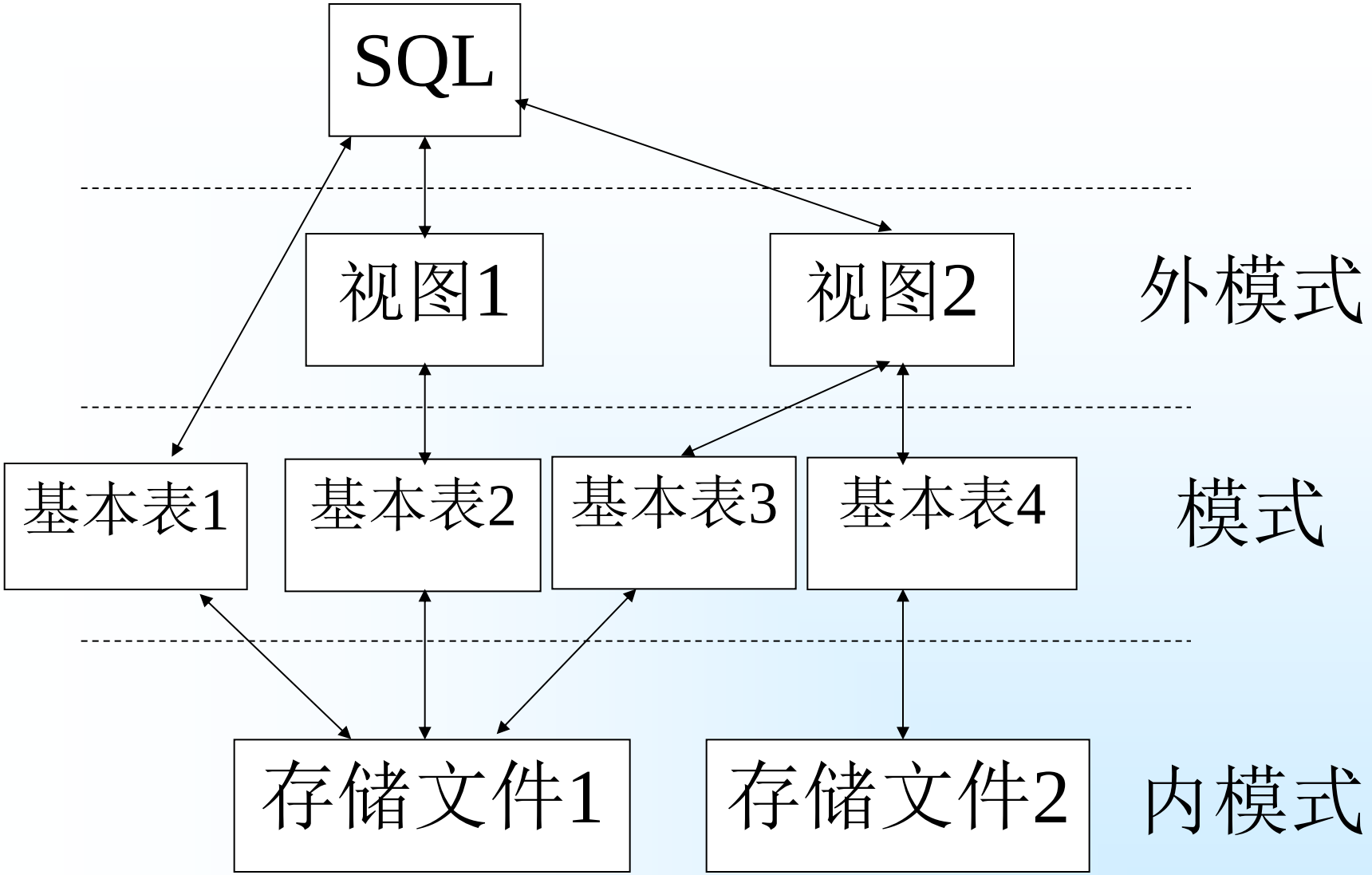

图3.1　数据库三级模式结构

返回本节首页

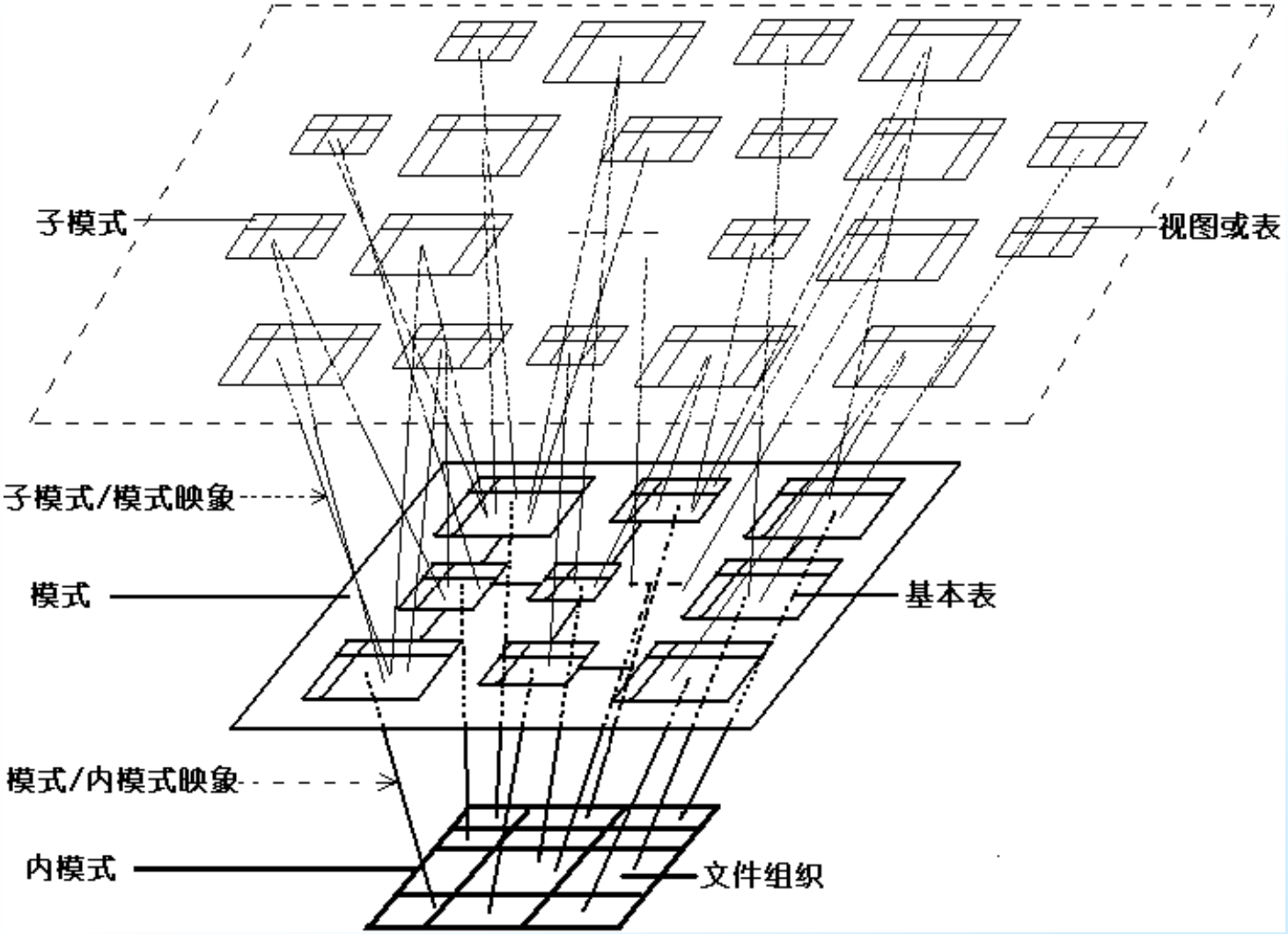

图3.2 关系数据库三级模式结构示意图

返回本节首页

3.1.2 SQL语言的基本概念

- 基本表是本身独立存在的表，在SQL中一个关系就对应一个表。一些基本表对应一个存储文件，一个表可以有若干索引，索引也存放在存储文件中。
- 视图是从基本表或其他视图中导出的表，它本身不独立存储在数据库中，也就是说数据库中只存放视图的定义而不存放视图对应的数据，这些数据仍存放在导出视图的基本表中，因此视图是一个虚表。
- 存储文件的物理结构及存储方式等组成了关系数据库的内模式。存储文件的物理结构及存储方式等不同数据库管理系统往往是不同的，一般也是不公开的。

返回本节首页

3.1.2 SQL语言的基本概念

- 视图和基本表是SQL语言的主要操作对象，用户可以用SQL语言对视图和基本表进行各种操作。在用户眼中，视图和基本表都是关系表，而存储文件对用户是透明的。

返回本节首页

3.1.3 SQL语言的主要特点

1、综合统一
2、高度非过程化
3、面向集合的操作方式
4、以同一种语法结构提供两种使用方式
5、语言简捷，易学易用

返回本节首页

3.2 SQL数据定义

返回本章首页

3.2.1 字段数据类型

- 整数数据类型:bigint,int,smallint,tinyint
- 精确数值类型:numeric,decimal
- 近似浮点数值数据类型:float,real
- 日期时间数据类型 :datetime,smalldatetime
- 字符串数据类型:char,varchar,text
- Unicode字符串数据类型: nchar,nvarchar,ntext

3.2.1 字段数据类型

- 二进制数据类型:binary、varbinary、image
- 货币数据类型: money,smallmoney
- 标记数据类型: timestamp,uniqueidentifier

返回本节首页

3.2.2 创建、修改和删除数据表

1. 定义基本表
2. 修改基本表
3. 删除基本表

返回本节首页

定义基本表

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，必须是合法的标识符，最多可有128个字符，但本地临时表的表名（名称前有一个编号符#）最多只能包含116个字符。表名不允许重名，一个表可以由一个或多个属性（列）组成。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。

返回本节首页

关系模型的完整性规则

(1)实体完整性

① 主码（PRIMARY KEY）

② 空值（NULL/NOT NULL）

③ 惟一值（UNIQUE）

(2)参照完整性

FOREIGN KEY约束指定某一个列或一组列作为外部键

(3)用户自定义的完整性约束规则

返回本节首页

完整性示例

- 下面我们以一个“学生-课程”数据库为例来说明，表内容请参图3.3。

- “学生-课程”数据库中包括三个表：（1）“学生”表S由学号（SNO）、姓名（SN）、性别（SEX）、年龄（AGE）、所在系（DEPT）五个属性组成，可记为： S（SNO，SN，SEX，AGE，DEPT）；（2）“课程”表C由课程号（CNO）、课程名（CN）、学分（CT）三个属性组成，可记为： C （CNO，CN，CT）；（3）“学生选课”表SC由学号（SNO）、课程号（CNO）、成绩（SCORE）三个属性组成，可记为：SC（SNO，CNO，SCORE）。

返回本节首页

S

学号SNO	姓名SN	性别SEX	年龄AGE	系别DEPT
S1	李涛	男	19	信息
S2	王林	女	18	计算机
S3	陈高	女	21	自动化
S4	张杰	男	17	自动化
S5	吴小丽	女	19	信息
S6	徐敏敏	女	20	计算机

图3.3

创建命令

返回本节首页

C

课程号CNO	课程名CN	学分CT
C1	C语言	4
C2	离散数学	2
C3	操作系统	3
C5	数据结构	4
C6	数据库	4
C7	汇编语言	3
C8	信息基础	2

图3.3

返回本节首页

SC

学号 SNO	课程号 SNO	成绩 SCORE
S1	C1	90
S1	C2	85
S2	C1	84
S2	C2	94
S2	C3	83
S3	C1	73
S3	C7	68
S3	C4	88
S3	C5	85
S4	C2	65
S4	C5	90
S4	C6	79
S5	C2	89

图3.3

返回本节首页

例1

- 建立一个“学生”表S，它由学号SNO、姓名SN、性别SEX、年龄AGE、所在系DEPT五个属性组成，其中学号属性为主键，姓名、年龄与性别不为空，假设姓名没有唯一并建立惟一索引，并且性别只能在“男”与“女”中选一个，年龄不能小于0。

- CREATE TABLE S
  (SNO CHAR(5) PRIMARY KEY,
    SN VARCHAR(8) NOT NULL,
    SEX CHAR(2) NOT NULL CHECK (SEX IN ('男','女')),
    AGE INT NOT NULL CHECK (AGE>0),
    DEPT VARCHAR(20),
    CONSTRAINT SN_U UNIQUE(SN)
)

返回本节首页

[例2] 建立“课程”表C，它由课程号（CNO）、课程名（CN）、学分（CT）三个属性组成。CNO为该表主键，学分大于等于1。

```
CREATE TABLE C
 ( CNO CHAR(5) NOT NULL PRIMARY KEY,
   CN VARCHAR(20),
   CT INT CHECK(CT>=1)
 )
```

[例3] 建立“选修”关系表SC，定义SNO,CNO为SC的外部键，(SNO,CNO)为该表的主键。

```
CREATE TABLE SC
 ( SNO CHAR(5) NOT NULL CONSTRAINT S_F FOREIGN
   KEY REFERENCES S(SNO),
   CNO CHAR(5) NOT NULL,
   SCORE NUMERIC(3),
   CONSTRAINT S_C_P PRIMARY KEY(SNO,CNO),
   CONSTRAINT C_F FOREIGN KEY(CNO) REFERENCES
   C(CNO)
 )
```

返回本节首页

修改基本表

ALTER TABLE table {

[**ALTER** COLUMN column_name { new_data_type [(precision [, scale])][COLLATE < collation_name >][NULL | NOT NULL]| {ADD | DROP } ROWGUIDCOL }]

| **ADD** { [< column_definition >]|column_name AS computed_column_expression } [,...n]

|[WITH CHECK | WITH NOCHECK] **ADD**{ < table_constraint > } [,...n]

| **DROP**{ [CONSTRAINT] constraint_name | COLUMN column } [,...n]

|**{CHECK | NOCHECK } CONSTRAINT** { ALL | constraint_name [,...n] }

|**{ ENABLE | DISABLE } TRIGGER** { ALL | trigger_name [,...n] }}

返回本节首页

修改基本表说明

其中:<表名> 指定需要修改的基本表,

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或原有列,

ALTER子句用于修改原有的列定义。

{CHECK|NOCHECK}CONSTRAINT 指定启用或禁用 constraint_name。如果禁用，将来插入或更新该列时将不用该约束条件进行验证。此选项只能与 FOREIGN KEY 和 CHECK 约束一起使用。

{ENABLE | DISABLE} TRIGGER 指定启用或禁用 trigger_name。当一个触发器被禁用时，它对表的定义依然存在；然而，当在表上执行 INSERT、UPDATE 或 DELETE 语句时，触发器中的操作将不执行，除非重新启用该触发器。

返回本节首页

修改表示例

[例4] 向S表增加“入学时间”列，其数据类型为日期型。

ALTER TABLE S **ADD** SCOME DATETIME

不论基本表中原来是否已有数据，新增加的列一律为空值。

[例5] 将年龄的数据类型改为半字长整数。

ALTER TABLE S **ALTER** COLUMN AGE SMALLINT

修改原有的列定义，会使列中数据作新旧类型的自动转化，有可能会破坏已有数据。

[例6] 删除例4增加的“入学时间”列。

ALTER TABLE S **DROP** COLUMN SCOME

[例7] 禁止SC中的参照完整性C_F。

ALTER TABLE S **NOCHECK** CONSTRAINT C_F

返回本节首页

删除基本表

DROP TABLE <表名>

[例8] 删除S表。

DROP TABLE S

返回本节首页

3.2.3 设计、创建和维护索引

1. 索引的概念
2. 创建索引
3. 删除索引

返回本节首页

索引的概念

- 数据库中的索引是为了加速对表中元组（或记录）的检索而创建的一种分散存储结构(如B^+树数据结构)，它实际上是记录的关键字与其相应地址的对应表。索引是对表或视图而建立的，由索引页面组成。
- 改变表中的数据(如增加或删除记录)时，索引将自动更新。索引建立后，在查询使用该列时，系统将自动使用索引进行查询。索引是把双刃剑，由于要建立索引页面，索引也会减慢更新的速度。索引数目无限制，但索引越多，更新数据的速度越慢。对于仅用于查询的表可多建索引，对于数据更新频繁的表则应少建索引。

返回本节首页

索引的概念

- 按照索引记录的存放位置可分为聚集索引(Clustered Index)与非聚集索引(Non-Clustered Index)两类。
- 聚集索引是指索引项的顺序与表中记录的物理顺序一致的索引组织；非聚集索引按照索引的字段排列记录，但是排列的结果并不会存储在表中，而是另外存储。在检索记录时，聚集索引会比非聚集索引速度快，一个表中只能有一个聚集索引，而非聚集索引可以有多个。

返回本节首页

创建索引

CREATE [UNIQUE]
[CLUSTERED|NONCLUSTERED] INDEX <索引名> ON { <表名> | <视图名> } (<列名> [ASC | DESC] [,...n]) [WITH < 索引选项 > [,...n]] [ON 文件组名]

- **[例9]** 为学生-课程数据库中的S、C、SC三个表建立索引。其中S表按学号升序建惟一索引，C表按课程号降序建立聚簇索引，SC表按学号升序和课程号降序建非聚簇索引。

```
CREATE UNIQUE INDEX S_SNO ON S(SNO)
CREATE CLUSTERED INDEX C_CNO ON C(CNO DESC)
CREATE NONCLUSTERED INDEX SC_SNO_CNO ON
    SC(SNO ASC,CNO DESC)
```

返回本节首页

删除索引

删除索引的命令语法：

DROP INDEX 表名.<索引名> |
视图名.<索引名> [,...n]

[例10] 删除S表的S_SNO索引。

```
DROP  INDEX  S.S_SNO
```

返回本节首页

3.3 SQL数据查询

返回本章首页

3.3.1 SELECT命令的格式之一

SELECT [ALL|DISTINCT]<目标列表达式>[,<目标列表达式>]...

[INTO <新表名>]

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]] ...

返回本节首页

3.3.1 SELECT命令的格式2

- SELECT [ALL|DISTINCT]<目标列表达式1> [[AS] 列别名1] [,<目标列表达式2> [[AS] 列别名2]]...
- [INTO <新表名>]
- FROM <表名1或视图名1> [[AS] 表别名1] [,<表名2或视图名2> [[AS] 表别名2]] ...
- [WHERE <元组或记录筛选条件表达式>]
- [GROUP BY <列名11>[,<列名12>] ... [HAVING <分组筛选条件表达式>]]
- [ORDER BY <列名21> [ASC|DESC] [,<列名22> [ASC|DESC]] ...]

返回本节首页

3.3.1 SELECT命令的含义

- 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11>的值进行分组（假设只有一列分组列），该属性列值相等的元组为一个组，每个组将产生结果表中的一条记录，通常会在每组中作用集函数。如果GROUP子句带HAVING短语，则只有满足指定条件的组才给予输出。如果有ORDER子句，则结果表还要按<列名22>的值的升序或降序排序后（假设只有一列排序列）再输出。

返回本节首页

3.3.1 SELECT命令的含义

SELECT语句组成成分的说明：

- 1、目标列表达式的可选格式：
- （1）[<表名>.]属性列名 | 各种普通函数 | 常量 | ...
- （2）[<表名>.]*
- （3）COUNT([ALL|DISTINCT] <属性列名>|*) 等集函数
- （4）算术运算（+、-、*、/）为主的表达式。其中参数可以是属性列名、集函数、常量、普通函数、表达式等形式。

返回本节首页

3.3.1 SELECT命令的含义

SELECT语句组成成分的说明：

- 2、集函数的可选格式：

COUNT([ALL|DISTINCT] <属性列名>|*)；

SUM | AVG | MAX | MIN([ALL|DISTINCT] <属性列名>)

返回本节首页

3.3.1 SELECT命令的含义

SELECT语句组成成分的说明：

- 3、WHERE子句的元组或记录筛选条件表达式有以下可选格式：
- （1）<属性列名> θ { <属性列名> | <常量> | [ANY | ALL] (SELECT 语句) }
- 其中θ为6种关系比较运算符之一。
- （2）<属性列名> NOT BETWEEN {<属性列名> | <常量> | (SELECT 语句)} AND {<属性列名> | <常量> | (SELECT 语句) }
- （3）<属性列名> [NOT] IN { (值1 [,值2]…) | (SELECT语句) }
- （4）<属性列名> [NOT] LIKE <匹配串>
- （5）<属性列名> IS [NOT] NULL
- （6）[NOT] EXISTS (SELECT 语句)
- （7）[NOT] <条件表达式> { AND|OR } [NOT] <条件表达式> [{ AND|OR } ([NOT] <条件表达式>)]…

返回本节首页

3.3.1 SELECT命令的含义

- SELECT语句组成成分的说明：
- 4、HAVING子句的分组筛选条件表达式有以下可选格式：
- HAVING子句的分组筛选条件表达式格式基本同WHERE子句的可选格式。不同的是HAVING子句的条件表达式中出现的属性列名应为GROUP BY子句中的分组列名。HAVING子句的条件表达式中一般要使用到集函数COUNT、SUM、AVG、MAX或MIN等，因为只有这样才能表达出筛选分组的要求。

返回本节首页

3.3.2 SELECT子句的基本使用

1. 查询指定列
2. 查询全部列
3. 查询经过计算的值

返回本节首页

查询指定列

- **[例11]** 查询全体学生的学号与姓名。

  SELECT SNO,SN

  FROM S

  <目标列表达式> 中各个列的先后顺序可以与表中的顺序不一致。也就是说，用户在查询时可以根据应用的需要改变列的显示顺序。
- **[例12]** 查询全体学生的姓名、学号、所在系。

  SELECT SN,SNO,DEPT

  FROM S

  这时结果表中的列的顺序与基表中不同，是按查询要求，先列出姓名属性，然后再列出学号和所在系属性。

返回本节首页

查询全部列

[例13] 查询全体学生的详细记录。

```
SELECT *
FROM S
```

该SELECT语句实际上是无条件地把S表的全部信息都查询出来，所以也称为全表查询，这是最简单的一种查询命令形式。它等价于如下命令：

```
SELECT SNO,SN,SEX,AGE,DEPT
FROM S
```

返回本节首页

查询经过计算的值

SELECT子句的<目标列表达式>不仅可以是表中的属性列，也可以是有关表达式，即可以将查询出来的属性列经过一定的计算后列出结果。

[例14] 查全体学生的姓名及其出生年份。

```
SELECT SN, 2005-AGE
FROM S
```

本例中，<目标列表达式>中第二项不是通常的列名，而是一个计算表达式，是用当前的年份（假设为2005年）减去学生的年龄，这样，所得的即是学生的出生年份。输出的结果为：

SN	
李涛	1986
王林	1987
陈高	1984
张杰	1988
吴小丽	1986
徐敏敏	1985

返回本节首页

3.3.3 WHERE子句的基本使用

1. 消除取值重复的行
2. 指定WHERE查询条件

返回本节首页

消除取值重复的行

[例16] 查所有选修过课的学生的学号。

```
SELECT SNO
FROM SC
```

结果为：

```
SNO
----
S1
S1
S2
S2
S2
S3
S3
S3
S3
S4
S4
S4
S5
```

该查询结果里包含了许多重复的行。如果想去掉结果表中的重复行，必须指定DISTINCT短语：

```
SELECT DISTINCT SNO
FROM SC
```

执行结果为：

```
SNO
----
S1
S2
S3
S4
S5
```

返回本节首页

表3.2 常用的查询条件

查询条件	谓词
比较运算符	=,>,<.>=,<=,!=,<>,!>,!<; Not (上述比较运算符构成的比较关系表达式)
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，NOT

返回本节首页

比较运算符

[例17] 查计算机系全体学生的名单。

```
SELECT SN
FROM S
WHERE DEPT = '计算机'
```

[例18] 查所有年龄在20岁以下的学生姓名及其年龄。

```
SELECT SN, AGE
FROM S
WHERE AGE <20
```

或

```
SELECT SN, AGE
FROM S
WHERE NOT AGE>= 20
```

返回本节首页

确定范围

[例20] 查询年龄在20至23岁之间的学生的姓名、系别和年龄。

```
SELECT SN, DEPT, AGE
FROM S
WHERE AGE BETWEEN 20 AND 23
```

与“BETWEEN…AND…”相对的谓词是“NOT BETWEEN…AND…”。

[例21] 查询年龄不在20至23岁之间的学生姓名、系别和年龄。

```
SELECT SN, DEPT, AGE
FROM S
WHERE AGE NOT BETWEEN 20 AND 23
```

返回本节首页

确定集合

[例22] 查询信息系、自动化系和计算机系的学生的姓名和性别。

```
SELECT SN, SEX
FROM S
WHERE DEPT IN ('信息', '自动化', '计算机')
```

与IN相对的谓词是NOT IN，用于查找属性值不属于指定集合的元组。

[例23] 查既不是信息系、数学系，也不是计算机科学系的学生的姓名和性别。

```
SELECT SN, SEX
FROM S
WHERE DEPT NOT IN ('信息', '自动化', '计算机')
```

返回本节首页

3.3 通配符及其含义

通配符	描述
%（百分号）	代表零个或更多字符的任意字符串。
_（下划线）	代表任何单个字符（长度可以为0）。
[ ]（中扩号）	指定范围 ([a-f]) 或集合 ([abcdef]) 中的任何单个字符。
[^]	不属于指定范围([^a-f])或集合 ([^abcdef]) 的任何单个字符。

返回本节首页

字符匹配示例1

[例24] 查所有姓刘的学生的姓名、学号和性别。

```
SELECT SN, SNO, SEX
FROM S
WHERE SN LIKE '刘%‘
```

[例25] 查姓“欧阳”且全名为三个汉字的学生的姓名。

```
SELECT SN
FROM S
WHERE SN LIKE '欧阳_'
```

返回本节首页

字符匹配示例2

[例26] 查名字中第二字为“阳”字的学生的姓名和学号。

```
SELECT SN, SNO
FROM S
WHERE SN LIKE '_阳%'
```

[例27] 查所有不姓刘的学生姓名。

```
SELECT SN, SNO, SEX
FROM S
WHERE SN NOT LIKE '刘%'
```

返回本节首页

字符匹配示例3

[例28] 查DB_Design课程的课程号和学分。

```
SELECT CNO, CT
FROM C
WHERE CN LIKE 'DB\_Design'  ESCAPE '\'
```

ESCAPE '\'短语表示\为换码字符，这样匹配串中紧跟在\后面的字符'_'不再具有通配符的含义，而是取其本身含义，被转义为普通的'_'字符。

返回本节首页

涉及空值的查询

[例29] 某些学生选修某门课程后没有参加考试，所以有选课记录，但没有考试成绩，下面我们来查一下缺少成绩的学生的学号和相应的课程号。

```
SELECT SNO, CNO
FROM SC
WHERE SCORE IS NULL
```

注意这里的'IS'不能用等号（'='）代替

[例30] 查所有有成绩的记录的学生学号和课程号。

```
SELECT SNO, CNO
FROM SC
WHERE SCORE IS NOT NULL
```

返回本节首页

多重条件查询1

- 逻辑运算符AND、OR和NOT可用来联结多个查询条件。他们的优先级NOT最高，接着是AND，OR优先级最低，但用户可以用括号改变运算的优先顺序。

- **[例31]** 查计算机系年龄在20岁以下的学生姓名。

```
SELECT SN
FROM S
WHERE DEPT='计算机' AND AGE<20
```

返回本节首页

多重条件查询2

[例32] IN谓词实际上是多个OR运算符的缩写，因此“查询信息系、自动化系和计算机系的学生的姓名和性别”一题，也可以用OR运算符写成如下等价形式：

```
SELECT SN, SEX
FROM S
WHERE DEPT='计算机' OR DEPT='信息' OR
  DEPT='自动化'
```

或

```
SELECT SN, SEX
FROM S
WHERE NOT(DEPT<>'计算机' AND DEPT<>'信息'
  AND DEPT<>'自动化')
```

返回本节首页

3.3.4 常用集函数及统计汇总查询

表3.4 常用集函数

COUNT	返回组中项目的数量。
SUM	返回表达式中所有值的和，或只返回 DISTINCT值的和。
AVG	返回组中值的平均值。
MAX	返回组中值的最大值。
MIN	返回组中值的最小值。

返回本节首页

- **[例33]** 查询学生总人数。

```
SELECT COUNT(*)
FROM S
```

- **[例34]** 查询选修了课程的学生人数。

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

学生每选修一门课，在SC中都有一条相应的记录，而一个学生一般都要选修多门课程，为避免重复计算学生人数，必须在COUNT函数中用DISTINCT短语。

- **[例35]** 计算C1课程的学生人数、最高成绩、最低成绩及平均成绩。

```
SELECT
    COUNT(*),MAX(SCORE),MIN(SCORE),AVG(SCORE)
FROM SC
WHERE CNO='C1'
```

返回本节首页

3.3.5 分组查询

GROUP BY子句可以将查询结果表的各行按一列或多列取值相等的原则进行分组。对查询结果分组的目的是为了细化集函数的作用对象。如果未对查询结果分组，集函数将作用于整个查询结果，即整个查询结果为一组对应统计产生一个函数值。否则，集函数将作用于每一个组，即每一组分别统计，分别产生一个函数值。

返回本节首页

[例36] 查询各个课程号与相应的选课人数。

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

该SELECT语句对SC表按CNO的取值进行分组，所有具有相同CNO值的元组为一组，然后对每一组作用集函数COUNT以求得该组的学生人数，执行结果为：

CNO	
C1	3
C2	4
C3	1
C4	1
C5	2
C6	1
C7	1

返回本节首页

如果分组后还要求按一定的条件对这些组进行筛选，
最终只输出满足指定条件组的统计值，
则可以使用HAVING短语指定筛选条件。

[例37] 查询有3人以上学生（包括3人）选修的课程的课程号及选修人数。

```
SELECT CNO, COUNT(SNO)
FROM SC
GROUP BY CNO
HAVING COUNT(*)>=3
```

结果为：

CNO	
C1	3
C2	4

返回本节首页

3.3.6 查询的排序

如果没有指定查询结果的显示顺序，DBMS将按其最方便的顺序（通常是元组在表中的先后顺序）输出查询结果。用户也可以用ORDER BY子句指定按照一个或多个属性列的升序（ASC）或降序（DESC）重新排列查询结果，其中升序ASC为缺省值。

返回本节首页

查询的排序示例1

[例38] 查询选修了3号课程的学生的学号及其成绩，查询结果按分数的降序排列。

```
SELECT SNO, SCORE
FROM SC
WHERE CNO='C3'
ORDER BY SCORE DESC
```

前面已经提到，可能有些学生选修了C3号课程后没有参加考试，即成绩列为空值。用ORDER BY子句对查询结果按成绩排序时，在SQL SERVER 2000中空值（NULL）被认为是最小值。

返回本节首页

查询的排序示例2

[例39] 查询全体学生情况，查询结果按所在系升序排列，对同一系中的学生按年龄降序排列。

```
SELECT *
FROM S
ORDER BY DEPT,AGE DESC
```

返回本节首页

3.3.7 连接查询

1、 等值与非等值连接查询

当连接运算符为=时，称为**等值连接**。使用其它运算符称为**非等值连接**。

2、 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己进行连接，这种连接称为表的**自身连接**。

3、 外连接

返回本节首页

等值与非等值连接查询

从概念上讲DBMS执行连接操作的过程是，首先在表1中找到第一个元组，然后从头开始顺序扫描或按索引扫描表2，查找满足连接条件的元组，每找到一个元组, 就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组。表2全部扫描完毕后，再到表1中找第二个元组，然后再从头开始顺序扫描或按索引扫描表2，查找满足连接条件的元组，每找到一个元组, 就将表1中的第二个元组与该元组拼接起来，形成结果表中一个元组。重复上述操作，直到表1全部元组都处理完毕为止。

返回本节首页

等值与非等值连接查询示例

[例40] 查询每个学生及其选修课程的情况。

学生情况存放在S表中，学生选课情况存放在SC表中，所以本查询实际上同时涉及S与SC两个表中的数据。这两个表之间的联系是通过两个表都具有的属性SNO实现的。要查询学生及其选修课程的情况，就必须将这两个表中学号相同的元组连接起来。这是一个等值连接。完成本查询的SQL语句为：

```
SELECT *
FROM S, SC
WHERE S.SNO=SC.SNO
```

返回本节首页

两种连接运算

- 连接运算中有两种特殊情况，一种称为**广义笛卡尔积连接**，另一种称为**自然连接**。
- 广义笛卡尔积连接是不带连接谓词的连接。两个表的广义笛卡尔积连接即是两表中元组的交叉乘积，也即其中一表中的每一元组都要与另一表中的每一元组作拼接，因此结果表往往很大。
- 如果是按照两个表中的相同属性进行等值连接，且目标列中去掉了重复的属性列，但保留了所有不重复的属性列，则称之为**自然连接**。

返回本节首页

[例41] 自然连接S和SC表。

```
SELECT S.SNO, SN, SEX, AGE, DEPT,
   CNO, SCORE
FROM S, SC
WHERE S.SNO=SC.SNO
```

在本查询中，由于SN、 SEX、 AGE、DEPT、CNO和SCORE属性列在S与SC表中是唯一的，因此引用时可以去掉表名前缀。而SNO在两个表都出现了，因此引用时必须加上表名前缀。该查询的执行结果不再出现SC.SNO列。

返回本节首页

自身连接示例

- **例3.42** 查询比李涛年龄大的学生的姓名、年龄和李涛的年龄。
- 要查询的内容均在同一表S中，可以将表S分别取两个别名，一个是X，一个是Y。将X，Y中满足比李涛年龄大的行连接起来。这实际上是同一表S的大于连接。
- 完成该查询的SQL语句为：

```
SELECT X.SN AS 姓名,X.AGE AS 年龄,Y.AGE AS 李涛的年龄
FROM S AS X, S AS Y
WHERE X.AGE>Y.AGE AND Y.SN='李涛'
```

- 结果为：

姓名	年龄	李涛的年龄
陈高	21	19
徐敏敏	20	19

返回本节首页

外连接 1

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，如在例40和例41的结果表中没有关于学生S6的信息，原因在于她没有选课，在SC表中没有相应的元组。但是有时我们想以S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，则只输出其基本情况信息，其选课信息为空值即可，这时就需要使用外连接（Outer Join）。外连接的运算符通常为*,有的关系数据库中也用+。

这样，我们就可以如下改写例41了：

返回本节首页

外连接 2

SELECT S.SNO, SN, SEX, AGE, DEPT, CNO, SCORE
FROM S **LEFT JOIN** SC **ON** S.SNO=SC.SNO

结果为：

SNO	SN	SEX	AGE	DEPT	CNO	SCORE
S1	李涛	男	19	信息	C1	90
S1	李涛	男	19	信息	C2	85
……						
S5	吴小丽	女	19	信息	C2	89
S6	**徐敏敏**	**女**	**20**	**计算机**	**NULL**	**NULL**

从查询结果可以看到，学号为S6的学生没选课，但S6的信息也出现在查询结果中，上例中外连接符LEFT [OUTER] JOIN称其为左外连接。相应地，如果RIGHT [OUTER] JOIN则称为右外连接，FULL [OUTER] JOIN称为全外连接。

返回本节首页

3.3.8 合并查询

合并查询结果就是使用UNION操作符将来自不同查询的数据组合起来，形成一个具有综合信息的查询结果。UNION操作会自动将重复的数据行剔除。必须注意的是，参加合并查询结果的各子查询的使用的表结构应该相同，即各子查询的数据数目相同，对应的数据类型要相融。

返回本节首页

合并查询示例

[例43] 从SC数据表中查询出学号为“S1”的同学的学号和总分，再从SC数据表中查询出学号为“S5”的同学的学号和总分，然后将两个查询结果合并成一个结果集。

```
SELECT SNO AS 学号，SUM(SCORE) AS 总分
FROM SC
WHERE (SNO='S1')
GROUP BY SNO
UNION
SELECT SNO AS 学号,SUM(SCORE) AS 总分
FROM SC
WHERE (SNO='S5')
GROUP BY SNO
```

返回本节首页

3.3.9 嵌套查询

1、 带有IN谓词的子查询
2、 带有比较运算符的子查询
3、 带有ANY或ALL谓词的子查询
4、 带有EXISTS谓词的子查询

返回本节首页

带有IN谓词的子查询

[例44] 查询与“王林”在同一个系学习的学生的学号、姓名和所在系。

查询与“王林”在同一个系学习的学生，可以首先确定“王林”所在系名，然后再查找所有在该系学习的学生。所以可以分步来完成此查询：

①确定“刘晨”所在系名

```
SELECT DEPT
FROM S
WHERE SN='王林'
```

结果为：

```
DEPT
------------------
计算机
```

②查找所有在计算机系学习的学生。

```
SELECT SNO, SN, DEPT
FROM S
WHERE DEPT='计算机'
```

返回本节首页

结果为：

SNO	SN	DEPT
S2	王林	计算机
S6	徐敏敏	计算机

分步写查询毕竟比较麻烦，上述查询实际上可以用子查询来实现，即将第一步查询嵌入到第二步查询中，用以构造第二步查询的条件。SQL语句如下：

```
SELECT SNO, SN, DEPT
FROM S
WHERE DEPT IN (
            SELECT DEPT
             FROM S
             WHERE SN='王林')
```

返回本节首页

本例中的查询也可以用我们前面学过的表的自身连接查询来完成：

```
SELECT S1.SNO, S1.SN, S1.DEPT
FROM S S1, S S2
WHERE S1.DEPT = S2.DEPT AND S2.SN='王林'
```

返回本节首页

带有比较运算符的子查询示例

例如：在例44中，由于一个学生只可能在一个系学习，也就是说内查询王林所在系的结果是一个唯一值，因此该查询也可以用比较运算符来实现，其SQL语句如下：

```
SELECT SNO, SN, DEPT
FROM S
WHERE DEPT =( SELECT DEPT
                FROM S
                WHERE SN='王林‘ )
```

返回本节首页

带有ANY或ALL谓词的子查询示例1

[例46] 查询其他系中比信息系任一学生年龄小的学生名单。

```
SELECT SN, AGE
FROM S
WHERE AGE<ALL(
    SELECT AGE
    FROM S
    WHERE DEPT='信息') AND DEPT <> '信息'
ORDER BY AGE DESC
```

返回本节首页

带有ANY或ALL谓词的子查询示例2

本查询实际上也可以用集函数（请参阅表3.6）实现。

```
SELECT SN, AGE
FROM S
WHERE AGE<
      ( SELECT MIN(AGE)
        FROM S
        WHERE DEPT='信息')  AND DEPT <> '信息'
ORDER BY AGE DESC
```

事实上，用集函数实现子查询通常比直接用ANY或ALL查询效率要高。

返回本节首页

带有EXISTS谓词的子查询示例1

[例47] 查询所有选修了C1号课程的学生姓名。

查询所有选修了C1号课程的学生姓名涉及S关系和SC关系，我们可以在S关系中依次取每个元组的SNO值，用此S.SNO值去检查SC关系，若SC中存在这样的元组，其SC.SNO值等于用来检查的S.SNO值，并且其SC.CNO='C1'，则取此S.SN送入结果关系。将此想法写成SQL语句就是：

```
SELECT SN
FROM S
WHERE EXISTS
    (SELECT *
      FROM SC
      WHERE SNO=S.SNO AND CNO='C1' )
```


返回本节首页

带有EXISTS谓词的子查询示例2

[例49] 查询选修了全部课程的学生姓名。

由于没有全称量词，我们将题目的意思转换成等价的存在量词的形式：查询这样的学生姓名，没有一门课程是他不选的。该查询涉及三个关系，存放学生姓名的S表，存放所有课程信息的C表，存放学生选课信息的SC表。其SQL语句为：

```
SELECT SN
FROM S
WHERE NOT EXISTS
    ( SELECT *
      FROM C
      WHERE NOT EXISTS
            ( SELECT *
              FROM SC
              WHERE SNO=S.SNO AND CNO=C.CNO))
```


返回本节首页

3.3.10 子查询别名表达式的使用*

在查询语句中，直接使用子查询别名的表达形式不失为一种简捷的查询表达方法。以下举例说明。

例3.50 在选修C2课程成绩大于该课平均成绩的学生中，查询还选C1课程的学生学号、姓名与C1课程成绩。

```
SELECT S.SNO,S.SN,SCORE
FROM SC,S,(SELECT SNO FROM SC
                WHERE CNO='C2' AND SCORE>(SELECT
   AVG(SCORE) FROM SC WHERE CNO='C2')) AS
   T1(sno)
WHERE SC.SNO=T1.SNO AND S.SNO=T1.SNO AND
   CNO='C1'
```

注意：通过AS关键字给子查询命名的表达方式，别名后的括号中可对应给子查询列指定列名。一旦命名，别名表达式可作为一般表一样的使用。

返回本节首页

3.3.10 子查询别名表达式的使用*

[例3.51] 查询选课门数唯一的学生的学号（例：若只有S1学号的学生选2门，则S1应为结果之一）

```
SELECT t3.SNO
FROM (SELECT CT
      FROM (SELECT SNO,COUNT(SNO) AS CT
            FROM SC GROUP BY SNO) AS T1(sno,ct)
      GROUP BY CT HAVING COUNT(*)=1
     ) AS T2(ct),(SELECT SNO,COUNT(SNO) AS CT
                  FROM SC GROUP BY SNO) AS T3(sno,ct)
WHERE T2.CT=T3.CT
```

返回本节首页

3.3.10 子查询别名表达式的使用*

例3.52 查询学习编号为“C2”课成绩为第3名的学生的学号（设选C2课的学生人数>=3）。

```
SELECT SC.SNO
FROM (SELECT MIN(SCORE)
      FROM ( SELECT DISTINCT TOP 3 SCORE FROM SC
             WHERE CNO='C2' ORDER BY SCORE DESC
           ) AS t1(SCORE)
      ) AS t2(SCORE) INNER JOIN SC
                     ON t2.SCORE=SC.SCORE
WHERE CNO='C2'
```

思考：读者可试试若不用子查询别名表达式的表示方法，这些查询该如何表达？

返回本节首页

3.3.11 存储查询结果到表中

使用SELECT…INTO语句可以将查询到的结果存储到一个新建的数据库表或临时表中。

[例53] 从SC数据表中查询出所有同学的学号和总分，并将查询结果存放到一个新的数据表Cal_Table中。

```
SELECT SNO AS 学号,SUM(SCORE) AS 总分
 INTO Cal_Table
 FROM SC
 GROUP BY SNO
```

返回本节首页

3.4 SQL数据更新

返回本章首页

3.4.1 插入数据

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>]...)

2、 插入子查询结果

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>]...)]

子查询

插入单个元组

[例51] 将一个新学生记录（学号：S7；姓名：陈冬；性别：男；年龄：18岁；所在系：信息）插入S表中。

INSERT INTO S VALUES ('S7', '陈冬', '男','18','信息')

[例52] 插入一条选课记录（'S7'，'C1'）。

INSERT INTO SC(SNO, CNO) VALUES('S7', 'C1')

新插入的记录在SCORE列上取空值。

返回本节首页

插入子查询结果

[例53] 对每一个系，求学生的平均年龄，并把结果存入数据库。

对于这道题，首先要在数据库中建立一个有两个属性列的新表，其中一列存放系名，另一列存放相应系的学生平均年龄。

```
CREATE TABLE DEPTAGE( DEPT CHAR(15),
                      AVGAGE TINYINT)
```

然后对数据库的S表按系分组求平均年龄，再把系名和平均年龄存入新表中。

```
INSERT INTO DEPTAGE(DEPT, AVGAGE)
    SELECT DEPT, AVG(AGE)
    FROM S
    GROUP BY DEPT
```

返回本节首页

3.4.2 修改数据

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子句用于指定修改方法，即用<表达式>的值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。

返回本节首页

修改某一个元组的值

[例54] 将学生S3(为学号)的年龄改为20岁。

```
UPDATE S
SET AGE=20
WHERE SNO='S3'
```

返回本节首页

修改多个元组的值

[例55] 将所有学生的年龄增加1岁。

```
UPDATE S
SET AGE=AGE+1
```

返回本节首页

带子查询的修改语句

[例56] 将计算机科学系全体学生的成绩置零。

```
UPDATE SC
SET SCORE=0
WHERE '计算机'=(SELECT DEPT
                FROM S
                WHERE SC.SNO=S.SNO)
```

或

```
UPDATE SC
SET SCORE=0
WHERE SNO IN (SELECT SNO
              FROM S
              WHERE DEPT='计算机')
```

3.4.3 删除数据

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，表示删除表中全部元组，但表的定义仍在字典中。也就是说，DELETE语句删除的只是表中的数据，而不包括表的结构定义。

1、 删除某一个元组的值

[例57] 删除学号为S7的学生记录。

```
DELETE
FROM S
WHERE SNO='S7'
```

2、 删除多个元组的值

[例58] 删除所有的学生选课记录。

```
DELETE  FROM SC
```

3、 带子查询的删除语句

子查询同样也可以嵌套在DELETE语句中

[例59] 删除计算机科学系所有学生的选课记录。

```
DELETE
FROM SC
WHERE '计算机'=
          ( SELECT DEPT
            FROM S
            WHERE S.SNO=SC.SNO)
```

更新操作举例

更新操作练习题

返回本节首页

3.5 视图

返回本章首页

视图的相关概念

关于视图

- 在关系数据库系统中，视图为用户提供了多种看待数据库数据的方法与途径，是关系数据库系统中的一种重要对象。
- 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。通过视图能操作数据，基本表数据的变化也能在刷新的视图中反映出来。从这个意义上讲，视图像一个窗口或望远镜，透过它可以看到数据库中自己感兴趣的数据及其变化。
- 视图在概念上与基本表等同，一经定义，就可以和基本表一样被查询、被删除，我们也可以在一个视图上再定义新的视图，但对视图的更新（插入、删除、修改）操作则有一定的限制。

返回本节首页

3.5.1 创建和删除视图

1、 创建视图

CREATE VIEW <视图名>[(<列名>[,<列名>]...)]

AS <子查询>

2、 删除视图

DROP VIEW <视图名>

[例62] 删除视图IS_S1。

DROP VIEW IS_S1

返回本节首页

创建视图示例

[例61] 建立信息系选修了C1号课程的学生的视图。

```
CREATE VIEW IS_S1(SNO, SN, SCORE)
AS SELECT S.SNO, SN, SCORE
    FROM S,SC
    WHERE  DEPT='信息' AND
S.SNO=SC.SNO AND SC.CNO='C1'
```

返回本节首页

查询视图示例

[例63] 在信息系学生的视图中找出年龄小于20岁的学生。

```
SELECT SNO, AGE
FROM IS_S
WHERE Sage<20
```

视图是定义在基本表上的虚表，它可以和其他基本表一起使用，实现连接查询或嵌套查询。这也就是说，在关系数据库的三级模式结构中，外模式不仅包括视图，而且还可以包括一些基本表。

返回本节首页

更新视图示例

[例64] 将信息系学生视图IS_S中学号为S3的学生姓名改为“刘辰”。

```
UPDATE IS_S
SET SN='刘辰'
WHERE SNO='S3'
```

返回本节首页

不同的数据库对视图的更新有不同的规定，如下是IBM的DB2数据库中视图不允许更新的规定：

1. 若视图是由两个以上基本表导出的，则此视图不允许更新。
2. 若视图的字段来自字段表达式或常数，则不允许对此视图执行INSERT和UPDATE操作，但允许执行DELETE操作。
3. 若视图的字段来自集函数，则此视图不允许更新。
4. 若视图定义中含有GROUP BY子句，则此视图不允许更新。
5. 若视图定义中含有DISTINCT短语，则此视图不允许更新。
6. 若视图定义中有嵌套查询，并且内层查询的FROM子句中涉及的表也是导出该视图的基本表，则此视图不允许更新。
7. 一个不允许更新的视图上定义的视图也不允许更新。

返回本节首页

3.6 SQL数据控制

返回本章首页

3.6.1 权限与角色

1、权限

SQL系统安全机制：视图机制、权限机制

数据库中权限：系统权限、对象权限

2、系统权限

系统权限是指数据库用户能够对数据库系统进行某种特定的操作的权力。它由数据库管理员授予其他用户。如创建一个基本表（CREATE TABLE）的权力。

3、对象权限

对象权限是指数据库用户在指定的数据库对象上进行某种特定的操作的权力。

返回本节首页

3.6.1 权限与角色

1、角色

角色是多种权限的集合，可以把角色授予用户或角色。当要为某一用户同时授予或收回多项权限时，则可以把这些权限定义为一个角色，对此角色进行操作。

这样就避免了许多重复性的工作，简化了管理数据库用户权限的工作。

返回本节首页

权限机制

在SQL系统中，有两个安全机制，一种是视图机制，当用户通过视图访问数据库时，他不能访问此视图外的数据，它提供了一定的安全性。而主要的安全机制是权限机制。权限机制的基本思想是给用户授予不同类型的权限，在必要时，可以收回授权，使用户能够进行的数据库操作以及所操作的数据限定在指定范围内，禁止用户超越权限对数据库进行非法的操作，从而保证数据库的安全性。

返回本节首页

3.6.2 系统权限与角色的授予与收回

1、系统权限与角色的授予

GRANT <系统权限>[,<系统权限>]... TO <用户>|角色[,<用户>|角色]...

[WITH GRANT OPTION]

2、系统权限与角色的收回

REVOKE <系统权限>[,<系统权限>]…

FROM <用户名 >|<角色>|PUBLIC[,<用户名>|<角色>]…

返回本节首页

系统权限与角色的授予与收回示例

[例65] 把创建表的权限授给用户U1。

```
GRANT CREATE TABLE TO U1
```

[例 66] 把U1所拥有的创建表权限收回。

```
REVOKE CREATE TABLE FROM U1
```

返回本节首页

3.6.3 对象权限与角色的授予与收回

1、对象权限与角色的授予

GRANT ALL|<对象权限>[（列名[，列名]…）][,<对象权限>]...ON <对象名> TO <用户>|<角色>|PUBLIC[,< 用 户 >|< 角 色 >]...[WITH GRANT OPTION]

2、对象权限与角色的收回

REVOKE<对象权限>|<角色>[,<对象权限>|<角色>]…

FROM<用户名>|<角色>|PUBLIC[,<用户名>|<角色>]…

权限命令

返回本节首页

对象权限与角色的授予与收回示例

[例67] 把查询S表权限授给用户U1。

GRANT SELECT ON S TO U1

[例 68] 收回用户U1对S表的查询权限。

REVOKE SELECT ON S FROM U1

返回本节首页

3.7 嵌入式SQL语言*

返回本章首页

嵌入式SQL的预编译、编译、连接与运行处理过程

源程序（主语言+SQL）→预编译系统→主语言源程序→主语言编译系统→目标代码→连接与运行

返回本节首页

3.7.2 嵌入式SQL要解决的三个问题

1、区分SQL语句与主语言语句

2、数据库工作单元和程序工作单元之间的通信

3、协调SQL集合式操作与高级语言记录式处理之间的关系

4、举例

返回本节首页

```
void ErrorHandler (void);
#include <stddef.h>          // standard C run-time header
#include <stdio.h>           // standard C run-time header
#include "gcutil.h"          // utility header
int main (int argc,char** argv,char** envp)
{   int nRet;                // for return values
    char yn[2];
     EXEC SQL BEGIN DECLARE SECTION;                    // ①先说明主变量
       char szServerDatabase[(SQLID_MAX * 2)+2] = "";// 放数据库服务器名与数据库名。
       char szLoginPassword[(SQLID_MAX * 2)+2] = ""; // 放登录用户名与口令。
     char tname[21]="xxxxxxxxxxx";  //放表名变量
       char cscustid[8];
       char csname[31];
       double csdiscount;
       double newdisc;
       int csdiscnull=0;
    EXEC SQL END DECLARE SECTION;
```

返回本节首页

```
// ②接着是错误处理设置与连接的相关选项设置。
EXEC SQL WHENEVER SQLERROR CALL ErrorHandler();
EXEC SQL SET OPTION LOGINTIME 10;
EXEC SQL SET OPTION QUERYTIME 100;
printf("Sample Embedded SQL for C application\n"); // display
    logo
// 若不使用“GetConnectToInfo()”，则也可直接指定“服务器
    名.数据库名”与
//“用户名.口令名”来连接，如EXEC SQL CONNECT TO
    qh.qxz USER sa.sa;
// 这里“qh”为服务器名，“qxz”为数据库名，“sa”为用户名，
    “sa”为口令。
// GetConnectToInfo()实现连接信息的获取，一般在
    “gcutil.c”C源程序中的。
nRet = GetConnectToInfo(argc, argv, szServerDatabase,
    szLoginPassword);
if (!nRet) { return (1); }
```

返回本节首页

```
// 下面CONNECT TO命令真正实现与SQL Server的连接
EXEC SQL CONNECT TO :szServerDatabase
    USER :szLoginPassword;
if (SQLCODE == 0) { printf("Connection to SQL Server
    established\n"); }
else { // problem connecting to SQL Server
printf("ERROR: Connection to SQL Server failed\n"); return (1);}
          // 检测数据库是否有customer表？
    EXEC SQL SELECT name into :tname FROM sysobjects // ③
    SELECT INTO 语句
        WHERE (xtype = 'U' and name='customer');
    if (SQLCODE == 0||strcmp(tname,"customer")==0)
    { printf("客户表已经存在。\n");}
    else{        // 若不存在customer表，则创建表并插入若干条记录。
EXEC SQL create table customer    // ④ 创建customer表
        (CustID   Dec( 7,0) not null,
         Name     Char(30)  not null,
         ShipCity Char(30) NULL,
         Discount Dec(5,3) NULL,
         primary key(CustID));
```

返回本节首页

```
if (SQLCODE == 0)
{ printf("create success!%d\n",SQLCODE);}
else
{ printf("ERROR: create %d\n",SQLCODE);return (-1);}
   EXEC SQL insert into customer values('133568','Smith
    Mfg.','Portland',0.050);
   EXEC SQL insert into customer values('246900','Bolt
    Co.','Eugene',0.020);
   EXEC SQL insert into customer values('275978','Ajax
    Inc','Albany',null); //⑤
   EXEC SQL insert into customer
    values('499320','Adapto','Portland',0.000);
   EXEC SQL insert into customer values('499921','Bell
    Bldg.','Eugene',0.100);
if (SQLCODE == 0){ printf("execute
    success!%d\n",SQLCODE);}
else{ printf("ERROR: execute %d\n",SQLCODE);return (-1);}
}
```


返回本节首页

```
EXEC SQL declare customercursor cursor  // ⑥定义游标
    customercursor
        for select custid,name,discount
            from customer
            order by custid
        for update of discount;
    EXEC SQL open customercursor;          // ⑦打开游标
     customercursor
    if (SQLCODE == 0){ printf("open success!%d\n",SQLCODE);}
    else { printf("ERROR: open %d\n",SQLCODE); return (-1);}
    while (SQLCODE == 0){
      EXEC SQL FETCH NEXT customercursor  // ⑧推进游标
     customercursor
            INTO :cscustid,:csname,:csdiscount :csdiscnull;
      if (SQLCODE == 0)
      {   printf("客户号=%s",cscustid);   // 显示客户信息
          printf("客户名=%14s",csname);
          if (csdiscnull==0) printf("折扣率=%lf\n",csdiscount);
          else printf("折扣率=NULL\n");
          printf("需要修改吗? (Y/N)?");   // 询问是否要修改?
```

返回本节首页

```
        scanf("%s",yn);
         if (yn[0]=='y' || yn[0]=='Y'){  // 输入并修改
            printf("请输入新的折扣率:");
            scanf("%lf",&newdisc);
            EXEC SQL update customer set
discount=:newdisc
                    where current of customercursor;  // ⑨
CURRENT形式的UPDATE语句
            if (SQLCODE == 0){printf("该客户的折扣率
修改成功！ ");}
             else{printf("该客户的折扣率修改未成功！ ");}
          };
    }
    else{printf("ERROR: fetch %d\n",SQLCODE);}
}
```

返回本节首页

```
    EXEC SQL close customercursor;  // ⑩关闭游标
    customercursor
    EXEC SQL DISCONNECT ALL;     // 关闭与数据
    库的连接
    return (0);
}
void ErrorHandler (void)          // 显示错误信息子程序
{   printf("Error Handler called:\n");
    printf("    SQL Code = %li\n", SQLCODE);
    printf("    SQL Server Message %li: '%Fs'\n",
    SQLERRD1, SQLERRMC);
}
```

返回本节首页

3.7.3 第四代数据库应用开发工具或高级语言中SQL的使用

1、Visual Basic 6.0中数据操作例子

2、C#中连接并执行SQL命令程序段

3、Java语言中通过“jdbc.odbc”连接并执行数据查询的程序段

返回本节首页

Visual Basic 6.0中数据操作例子

该例子利用VB实现类似SQL Server查询分析器的功能。当运行时，左边文本框中可输入对数据库表的查询类命令（SELECT），按左文本框下的按钮，上网格控件中即能显示出SELECT查询的结果（当然要输入正确的SELECT命令）；右边文本框中可输入对数据库表的更新类命令（如INSERT、UPDATE、DELETE），同样SQL命令正确的话即能更新操作数据库中的数据，更新数据后左边文本框中再输入查询命令能加以检验，如此强大的SQL命令交互操作功能，利用ADO数据对象实现非常轻松。请读者参照实现本例子，领域VB操作数据库数据的方法。

返回本节首页

带适当注释的两命令按钮代码如下：

```
Private Sub RunSelect_Click()
Dim cn As New ADODB.Connection   '定义并实例化ADO连接对象变量cn
Dim cmd As New ADODB.Command    '定义并实例化ADO命令对象变量cmd
'定义并实例化ADO记录集对象变量rs（它相当于C语言中的游标，能返回记录集，并通过它操作数据）
Dim rs As New ADODB.Recordset
On Error GoTo RunSQL_Error      '遇到错误，跳转到RunSQL_Error
'设置连接对象cn的连接数据库属性，这里要求当前目录中已存在含表js的Access数据库js.mdb文件。
```

返回本节首页

```
cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=js.mdb"
cn.Open   '打开连接对象cn
cmd.ActiveConnection = cn        '命令对象cmd的活动连接设置为cn
cmd.CommandType = adCmdText      '命令对象cmd的命令类型为SQL命令
cmd.CommandText = Text1.Text     '查询类SQL命令来自Text1文本框，设置对象cmd
'cmd.CommandText ="select * from js" '如设定本语句，能得到图3.6的运行效果。
rs.CursorLocation = adUseClient  '记录集对象rs定位于客户端
rs.CursorType = adOpenStatic     '记录集对象rs为静态记录集
rs.LockType = adLockReadOnly     '记录集对象rs为只读记录集
rs.Open Cmd     '通过命令对象Cmd，打开记录集对象rs，即借助命令对象含有的SQL命令，从数据库中取得了记录集，并放在记录集对象rs中。
Set DataGrid1.DataSource = rs     '记录集对象rs赋值给网格控件DataGrid1，界面上能看到查询到的记录内容。
Exit Sub  '退出子程序
RunSQL_Error:
MsgBox "错误: " & Err.Description   '遇到错误时，显示错误信息。
End Sub
```


返回本节首页

```
Private Sub RunSqls_Click()
Dim cn As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim rs As New ADODB.Recordset
On Error GoTo RunSQL_Error
cn.ConnectionString =
    "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=js.mdb"
cn.Open
cmd.ActiveConnection = cn
cmd.CommandType = adCmdText
cmd.CommandText = Text2.Text   ' 更新类SQL命令来自Text1
    文本框。
Cmd.Execute   '更新类SQL命令，由Cmd的Execute方法直接
    递交到数据源执行。
MsgBox "已成功执行，请验证执行结果！" & Cmd.State
Exit Sub
RunSQL_Error:
MsgBox "错误: " & Err.Description
End Sub
```

返回本节首页

C#中连接并执行SQL命令程序段

- .NET集成环境中的C#操作数据库是通过ADO.NET应用程序级接口操作数据的，ADO.NET 提供对 Microsoft SQL Server 等数据源以及通过 OLE DB 和 XML 公开的数据源的一致访问。数据共享使用者应用程序可以使用 ADO.NET 来连接到这些数据源,并检索、操作和更新数据。
- 一个 OleDbConnection 对象，表示到数据源的一个唯一的连接。在客户端/服务器数据库系统的情况下，它等效于到服务器的一个网络连接。下面的示例创建一个 OleDbCommand 和一个 OleDbConnection对象。OleDbConnection 打开，并设置为 OleDbCommand 的 Connection。然后，该示例调用 ExecuteNonQuery执行INSERT插入操作，完成向Customers表中插入一条记录，并关闭该连接。

返回本节首页

```
public void InsertRow(string
    myConnectionString)
{
    if(myConnectionString == ""){

    myConnectionString="Provider=SQLOLEDB
    ;Data Source=localhost;Initial
    Catalog=Northwind;Integrated
    Security=SSPI;"; // 连接到SQL Server中的
    Northwind数据库
    }
    // 由连接字符串，创建OleDb连接对象
    myConnection
    OleDbConnection myConnection = new
    OleDbConnection(myConnectionString);
```

```
string myInsertQuery = "INSERT INTO Customers (CustomerID, CompanyName) Values('NWIND', 'Northwind Traders')";     // 把SQL插入命令赋值给变量myInsertQuery
OleDbCommand myCommand = new OleDbCommand(myInsertQuery);  //建立命令对象myCommand
myCommand.Connection = myConnection; // 对象myCommand的当前连接设置为myConnection
myConnection.Open();                    //打开并建立连接
//通过命令对象myCommand，借助myConnection连接对象执行SQL插入操作。
myCommand.ExecuteNonQuery();
myCommand.Connection.Close();       //由命令对象关闭已建立的连接。
}
```

返回本节首页

Java语言中通过“jdbc.odbc”连接并执行数据查询的程序段

```
package javabean;                    //包名
import java.sql.*;                   //导入相关类
public class DBBean{
String sDBDriver ="sun.jdbc.odbc.JdbcodbcDriver";
   //加载Jdbc.odbc驱动
String sConnStr = "jdbc:odbc:DBClothes";
  //数据源连接字符串，其中DBClothes为预先建立的连接
    数据库的ODBC数据源名。
Connection conn = null;      //定义一数据库连接对象conn
Statement stmt = null;       //定义一数据库命令对象stmt
ResultSet rs = null;         //定义一结果集rs
```

返回本节首页

```
public DBBean(){                    <!--注册数据库驱动程序-->
try{ Class.forName(sDBDriver);}    //检查是否有该类数据库的驱动程序
catch(java.lang.ClassNotFoundException e){
System.err.println("DBBean()"+e.getMessage());//异常处理
}
}
<!--建立数据库连接及定义数据查询-->
public ResultSet executeQuery(String sql){
rs = null;
try{ //通过数据源连接字符串(sConnStr)及用户名（sa）与密码(11)，建
    立连接对象conn
conn=DriverManager.getConnection(sConnStr,"sa","11");
stmt = conn.createStatement();  // 通过连接对象创建命令对象stmt
rs = stmt.executeQuery(sql);    // 通过命令对象执行数据查询命令，取
    得的记录集赋给记录集对象rs。
}catch(SQLException ex){
System.err.println("executeQuery:"+ex.getMessage());}
return rs;                      // 函数返回记录集rs
}
……
}
```

返回本节首页

习 题

一、选择题

1、在SQL语言中授权的操作是通过_____语句实现的。

A、CREATE　B、REVOKE　C、GRANT　D、INSERT

2、SQL语言的一体化特点是主要同( )相比较而言的。

A.操作系统命令　　B.非关系模型的数据语言

C.高级语言　　D.关系模型语言

3、在嵌入式SQL语言中使用游标的目的在于_____。

A、区分SQL与宿主语言　B、与数据库通信

C、处理错误信息　　D、处理多行记录

4、设有关系R=（A，B，C）。与SQL语句SELECT DISTINCT A FROM R WHERE B=17等价的关系代数表达式是_____。

A. ∏A(R)　　B. σB=17(R)

C. ∏A(σB=17(R))　　D. σB=17(∏A(R))

5、两个子查询的结果( )时，可以执行并、交、差操作。

A. 结构完全一致　　B.结构完全不一致

C. 结构部分一致　　C.主键一致

6、在SQL查询语句中，用于测试子查询是否为空的谓词是( )。

A.Exists　　B.Unique　　C.Some　　D.All

7、使用SQL语句进行查询操作时，若希望查询结果中不出现重复元组，应在Select子句中使用()保留字。

A.Unique　　B.All　　C.Except　　D.Distinct

8、在视图上不可能完成的操作是( )

A.更新视图　　B.查询

C.在视图上定义新的基本表　　D.在视图上定义新视图

9、SQL中涉及属性Age是否是空值的比较操作，写法( )是错误的。

A. Age Is Null　　B. Not(Age Is Null)

C. Age=Null　　D.Age Is Not Null

10、假定学生关系是S(S#,Sname,Sex,Age)，课程关系是C(C#,CName,TEACHER)，学生选课关系是SC(S#,C#,Grade)。要查找选修“数据库系统概论”课程的“男”学生学号，将涉及到关系( )。

A.S　　B.SC,C　　C.S,SC　　D.S,SC,C

二、填空题

1、SQL操作命令CREATE、DROP、ALTER主要完成的是数据的_____功能。

2、_____为关系数据库语言国际标准语言。

3、SQL中文含义是_____，它集查询、操纵、定义和控制等多种功能。

4、视图是从_____导出的表。它相当于三级结构中的外模式。

5、视图是虚表，它一经定义就可以和基本表一样被查询，但_____操作将有一定限制。

6、SQL的数据更新功能主要包括_____、_____和_____三个语句。

7、在字符匹配查询中，通配符“%”代表_____，“_”代表_____。

8、SQL语句具有_____和_____两种使用方式。

9、SQL语言中，实现数据检索的语句是_____。

10、在SQL中如果希望将查询结果排序，应在Select语句中使用_____子句。

三、简答与SQL操作表达

1、简述SQL的定义功能。

2、简述SQL语言支持的三级逻辑结构。

3、解释本章所涉及的有关基本概念的定义：基本表、导出表、视图、索引、聚集、系统特权、对象特权、角色，并说明视图、索引、聚集、角色的作用。

4、在对数据库进行操作的过程中，设置视图机制有什么优点？它与数据表之间有什么区别？

5、设有四个关系（只示意性给出一条记录）：

S

SNO	SNAME	ADDRESS	TEL
S1	SN1	上海南京路	68564345

SPJ

SNO	PNO	JNO	QTY
S1	P1	J1	200

P

PNO	PNAME	SPEC	CITY	COLOR
P1	PN1	8X8	无锡	红

J

JNO	JNAME	LEADER	BG
J1	JN1	王总	10

S（SNO，SNAME，ADDRESS，TEL）其中，SNO：供应商代码 SNAME：姓名 ADDRESS：地址 TEL：电话；

J（JNO，JNAME，LEADER，BG）其中，JNO：工程代码 JNAME：工程名 LEADER：负责人 BG：预算；

P（PNO，PNAME，SPEC，CITY，COLOR）其中，PNO：零件代码 PNAME：零件名 SPEC：规格 CITY：产地 COLOR：颜色；

SPJ（SNO，JNO，PNO，QTY）其中，SNO：供应商代码 JNO：工程代码 PNO：零件代码 QTY：数量；

为每个关系建立相应的表结构，添加若干记录。

完成如下查询：

1. 找出所有供应商的姓名和地址、电话。
2. 找出所有零件的名称、规格、产地。
3. 找出使用供应商代码为S1供应零件的工程号。
4. 找出工程代码为J2的工程使用的所有零件名称、数量。
5. 找出产地为上海的所有零件代码和规格。
6. 找出使用上海产的零件的工程名称。
7. 找出没有使用天津产的零件的工程号。
8. 求没有使用天津产的红色零件的工程号。
9. 取出为工程J1和J2提供零件的供应商代号。
10. 找出使用供应商S2供应的全部零件的工程号。

完成如下更新操作：

1. 把全部红色零件的颜色改成蓝色;
2. 由S10供给J4的零件P6改为由S8供应，请作必要的修改。
3. 从供应商关系中删除S2的记录，并从供应零件关系中删除相应的记录。
4. 请将 (S2, J8, P4, 200) 插入供应零件关系。
5. 将工程J2的预算改为40万元。
6. 删除工程J8订购的S4的零件。

请将“零件”和“供应零件”关系的连接定义一个视图，完成下列查询：

1. 找出工程代码为J2的工程使用的所有零件名称、数量。
2. 找出使用上海产的零件的工程号。

6、在嵌入式SQL中如何区分SQL语句和主语句的？举例说明。

7、在嵌入式SQL中如何解决数据库工作单元与程序工作单元之间沟通的？

8、SQL的集合处理方式与宿主语言的单记录处理方式之间如何协调？