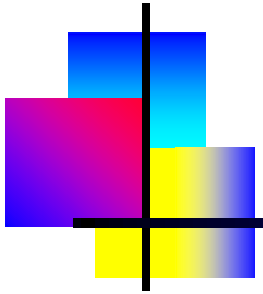
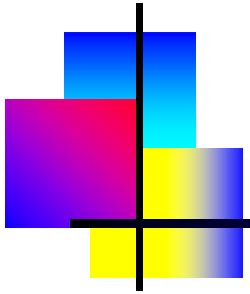


教材：数据库实用教程（第四版）

《数据库原理》课程

清华大学出版社

2024年12月30日



《数据库原理》

第四章 结构化查询语言

清华大学出版社
2024年12月30日

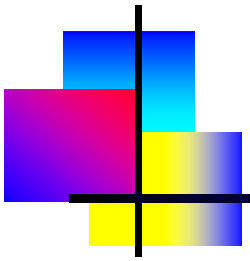
第四章　结构化查询语言SQL

教学内容：

- SQL数据库的体系结构，SQL的组成；
- SQL的数据定义：SQL模式、基本表和索引的创建和撤销；
- SQL的数据查询：SELECT语句的句法和使用；
- SQL的数据更新：插入、删除和修改语句；
- 视图的创建和撤消，对视图更新操作的限制；
- 嵌入式SQL：预处理方式，使用规定，使用技术，动态SQL语句。

教学重点：

- SQL的数据查询；
- 嵌入式SQL的使用。



§1 SQL概貌及特点

一、 SQL数据库的体系结构

SQL数据库的体系结构基本上也是三级模式结构。

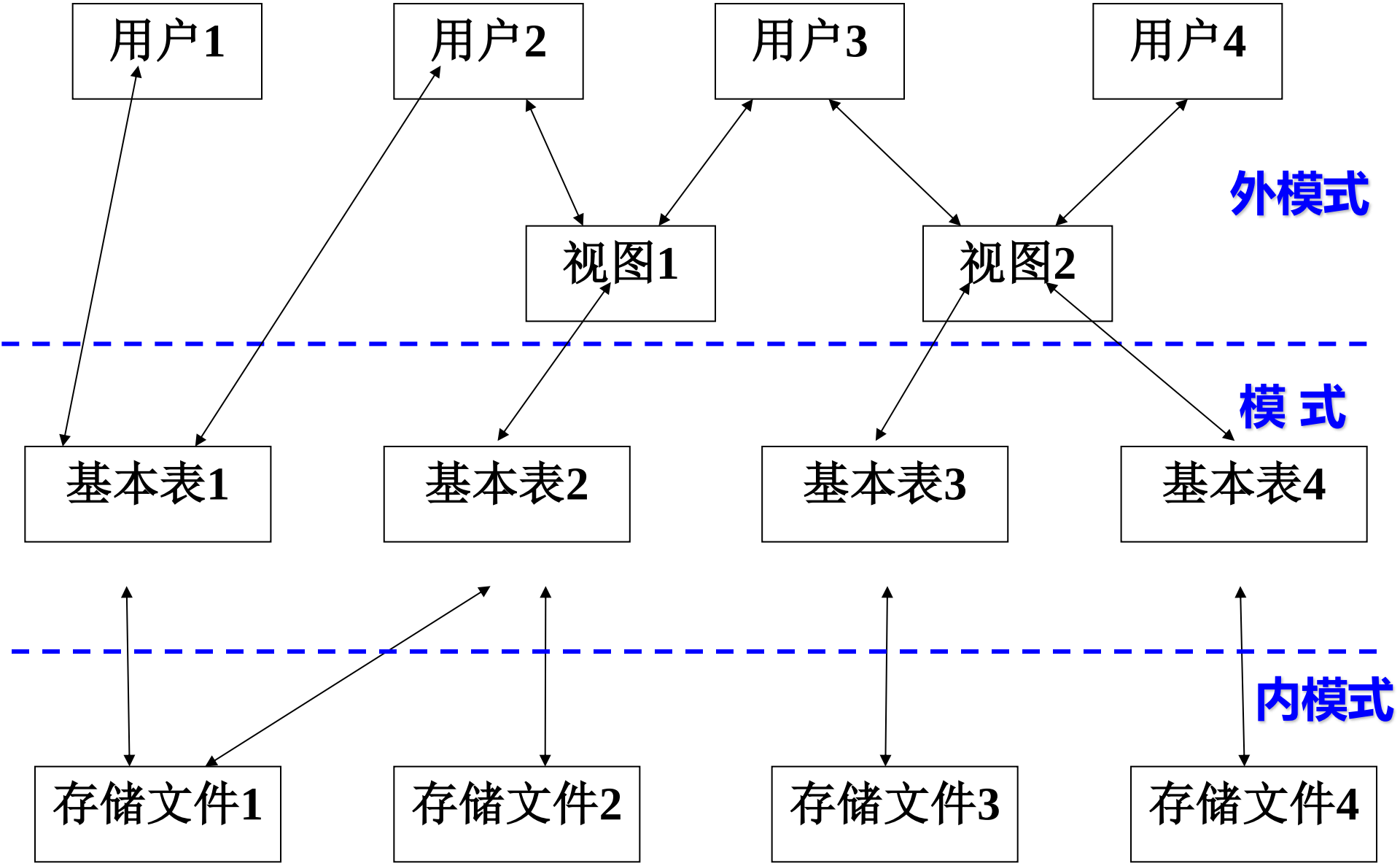
在SQL 中： 外模式对应于视图，

模式对应于基本表，

元组称为“行”，

属性称为“列”，

内模式对应于存储文件。


用户1
用户2
用户3
用户4
外模式
视图1
视图2
模 式
基本表1
基本表2
基本表3
基本表4
内模式
存储文件1
存储文件2
存储文件3
存储文件4

二、SQL的组成

SQL主要分成四个部分：

数据定义；

数据操纵；

数据控制；

嵌入式SQL的使用。

三、SQL的主要特点

1. 一体化；

2. 两种使用方式，统一的语法结构；

3. 高度非过程化；

4. 语言简洁，易学易用。

§2 SQL的数据定义

一、SQL模式的创建和撤消

1. SQL模式的创建

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>

2. SQL模式的撤消

DROP SCHEMA <模式名> [CASCADE|RESTRICT]

CREATE DATABASE <数据库名>

DROP DATABASE <数据库名>

例：创建工程数据库PROJECT

```
CREATE  DATABASE  PROJECT
```

例：撤消工程数据库PROJECT

```
DROP  DATABASE  PROJECT
```

例：创建学生数据库 STUDENT

```
CREATE  DATABASE  STUDENT
```

例：撤消学生数据库 STUDENT

```
DROP  DATABASE  STUDENT
```

二、基本表的创建、修改和撤消

1.基本表的创建

CREATE TABLE SQL 模式名.基本表名

（列名 类型，

……

完整性约束，

……）

完整性规则主要有三种：

主键子句（PRIMARY KEY）；

检查子句（CHECK）；

外键子句（FOREIGN KEY）。

SQL提供的基本数据类型:

数值型: INTEGER 长整数（也可写成INT）

SMALLINT 短整数

REAL 取决于机器精度的浮点数

DOUBLE PRECISION 取决于机器精度的双精度

浮点数: FLOAT(n) 浮点数，精度至少为n位数字;

NUMERIC（p,d） 定点数，由p位数字（不包括符号、小数点）组成，小数点后面有d位数字;

也可写成:DECIMAL（P，d）或DEC（P，d））

SQL提供的基本数据类型

字符串型：	CHAR（n）	长度为n的定长字符串
	VARCHAR(n)	具有最大长度为n的变长字符串
位串型：	BIT（n）	长度为n的二进制位串
	BIT VARYING（n）	最大长度为 n的变长二进制位
时间型：	DATE	日期，包含年、月、日，YYYY—MM－DD
	TIME	时间，包含时、分、秒，HH:MM:SS

例：工程项目数据库PROJECT中有四个关系，其结构如下：

供应商关系：　S(SNO，SNAME，SADDR)

零件关系：　P(PNO，PNAME，COLOR，WEIGHT)

工程项目关系：J(JNO，JNAME，JCITY，BALANCE)

供应情况关系：SPJ(SNO，PNO，JNO，PRICE，QTY)

可用下列语句创建表S:

```
CREATE TABLE S
(SNO CHAR(4) NOT NULL,
 SNAME CHAR(20) NOT NULL,
 SADDR CHAR(20),
 PRIMARY KEY(SNO));
```

可用下列语句创建表P:

```
CREATE TABLE P
 (PNO CHAR(4) NOT NULL,
 PNAME CHAR(20) NOT NULL,
 COLOR  CHAR(8),
 WEIGHT SMALLINT,
 PRIMARY KEY (PNO));
```

可用下列语句创建表S:

```
CREATE TABLE J
(JNO CHAR(4) NOT NULL,
 JNAME CHAR(20),
 JCITY CHAR(20),
 BALANCE NUMERIC(7,2),
  PRIMARY KEY(JNO));
```

可用下列语句创建表SPJ:

```
CREATE TABLE SPJ
(SNO CHAR(4)  NOT NULL,
PNO CHAR(4) NOT NULL,
JNO CHAR(4) NOT NULL,
PRICE NUMERIC(7,2),
QTY  SMALLINT,
PRIMARY KEY(SNO,PNO,JNO),
FOREIGN KEY(SNO)REFERENCES S(SNO),
FOREIGN KEY(PNO)REFERENCES P(PNO),
FOREIGN KEY(JNO)REFERENCES J(JNO),
CONSTRAINT C_QTY CHECK(QTY BETWEEN 0 AND 10000)
```

例：学生数据库STUDENT中有三个关系，其结构如下：

学生关系：　S(SNO，SNAME，AGE，SEX，SDEPT)

课程关系：　C(CNO，CNAME，CDEPT，TNAME)

学习关系：　SC(SNO，CNO，GRADE)

可用下列语句创建表S:

```
CREATE TABLE S
(SNO CHAR(4) NOT NULL,
 SNAME CHAR(20) NOT NULL,
 AGE CHAR(2),
 SEX CHAR(2),
 SDEPT  CHAR(10),
 PRIMARY KEY(SNO));
```

主键子句

可用下列语句创建表C:

```
CREATE TABLE C
 (CNO CHAR(4) NOT NULL,
 CNAME CHAR(20) NOT NULL,
 CDEPT  CHAR(10),
 TNAME CHAR(8),
 PRIMARY KEY (CNO));
```

主键子句

可用下列语句创建表SC:

```
CREATE TABLE SC
  (SNO CHAR(4)   NOT NULL,
   CNO CHAR(4) NOT NULL,
   GRADE NUMERIC(7,2),
   PRIMARY KEY(SNO,CNO),
   FOREIGN KEY(SNO)REFERENCES S(SNO),
   FOREIGN KEY(CNO)REFERENCES P(CNO),
   CONSTRAINT SC_GRADE CHECK(GRADE BETWEEN 0 AND 100));
```

主键子句

外键子句

检查子句

2. 基本表结构的修改

- 增加新的属性：

ALTER TABLE 基本表名 ADD 新属性名 新属性类型

例:在基本表S中增加一个电话号码（TELE）属性语句如下：

```
ALTER TABLE S ADD TELE CHAR(12);
```

- 删除原有的属性：

ALTER TABLE 基本表名 DROP 属性名 [CASCADE|RESTRICT]

例:在表S中删除电话号码（TELE）属性，并且将引用该属性的所有视图和约束也一起删除，可用下列语句：

```
ALTER TABLE S DROP TELE CASCADE;
```

- 删除指定的完整性约束条件，其句法如下：

ALTER TABLE 基本表名 DROP 约束名；

例：在基本表SC中删除对属性GRADE的约束，可用下列语句：

ALTER TABLE SC DROP SC_GRADE ；

另外，可以用下列语句增加（即恢复）对属性GRADE的约束：

```
ALTER TABLE SPJ ADD CONSTRAINT SC_GRADE
                    CHECK(GRADE BETWEEN 0 AND 100)
```

3. 删除基本表

删除基本表可以用“DROP TABLE”语句删除基本表。其句法如下：

DROP TABLE 基本表[CASCADE|RESTRICT]

此处的CASCADE和RESTRICT的语义同前面句法中的语义一样。

在一个基本表删除后，其所有数据也就丢失了，使用时要谨慎。

§3　SQL的数据查询

SQL的数据查询(SELECT语句)是SQL的核心内容。

一、SELECT语句的来历

在关系代数中最常用的式子是下列表达式：

$$\pi_{A1,\ldots,An}(\sigma_F(R_1\times\ldots\times R_m))$$

这里R1、…、Rm为关系，F是公式，A1、…、An为属性。

为此SQL设计成 SELECT – FROM - WHERE句型：

SELECT　A_1，…，A_n

FROM　　R_1，…，R_m

WHERE　F

二、SELECT语句格式：

SELECT [DISTINCT] 目标表的列名(或)列表达式序列

FROM 基本表名(或)视图名序列|表引用

[WHERE 行条件表达式]

[GROUP BY 列名1序列 [HAVING 组条件表达式]]

[ORDER BY 列名2 [ASC|DESC]序列]；

整个语句的执行过程如下：

1. 读取FROM子句中基本表、视图的数据，执行笛卡尔积操作。

2. 选取满足WHERE子句中给出的条件表达式的元组。

3. 按GROUP子句中指定列的值分组，同时提取满足HAVING子句中组条件表达式的那些组。

4. 按SELECT子句中给出的列名或列表达式求值输出。

5. ORDER子句对输出的目标表进行排序,按附加说明ASC升序排列，或按DESC降序排列。

SELECT语句中:

WHERE子句称为“行条件子句”，

GROUP子句称为“分组子句”，

HAVING子句称为“组条件子句”，

ORDER子句称为“排序子句”。

在WHERE子句的行条件表达式中可使用下列运算符：

- 算术比较运算符：<，<=，>，>=，=，<>或！=；
- 逻辑运算符：AND，OR，NOT；
- 集合成员资格运算符：IN，NOT IN；
- 谓词：EXISTS，ALL，SOME，UNIQUE；
- 聚合函数：AVG，MIN，MAX，SUM，COUNT；
- 集合运算符：UNION，INTERSECT，EXCEPT。

3. 举例　　（板书举例  ）

举例： 假设学生数据库中的关系模式如下：

S（SNO, SNAME, AGE, SEX, SDEPT）

C（CNO, CNAME, CDEPT, TNAME）

SC（SNO, CNO, GRADE）

试用SQL表达下列每个查询语句。

1. 检索选修课程号为C2的学生的学号和成绩。

```
SELECT SNO,GRADE
FROM SC
WHERE CNO='C2';
```

2. 检索计算机软件、姓“李”的全体男同学的学号、姓名和出生年份。

```
SELECT SNO,SNAME,2020-AGE AS BIRTH_YEAR
FROM S
WHERE SNAME LIKE '李%'
   AND SDEPT='计算机软件'
   AND SEX='男';
```

别名

LIKE 谓词的一般形式是：

列名 LIKE 字符串常数

其中：列名的类型必须是字符串或可变字符串。

字符串常数中通配符的含义如下：

%(百分号)：表示可以与任意长度的字符串匹配。

_(下划线)：表示可以与任意单个字符匹配。

所有其他的字符只代表自己。

3. 检索选修课程号为C4的学生的学号和姓名。

第一种方法：连接查询

```
SELECT S.SNO,SNAME
FROM S,SC
WHERE S.SNO=SC.SNO
   AND CNO='C4';
```

第三种方法：使用存在量词的嵌套

```
SELECT S.SNO,SNAME
 FROM S
 WHERE EXISTS
  (SELECT * FROM SC
   WHERE S.SNO=SC.SNO
     AND SC.CNO= 'C4' );
```

子查询依赖于外层查询

第二种方法：嵌套查询①

```
SELECT S.SNO,SNAME- 外层查询
FROM S
WHERE SNO IN
  (SELECT SNO        - 内层查询
   FROM SC           (即:子查询)
   WHERE CNO='C4' );
```

第二种方法：嵌套查询 ②相关子查询)

```
SELECT S.SNO,SNAME
FROM S
WHERE 'C4' IN
  (SELECT CNO
   FROM SC
   WHERE S.SNO=SC.SNO);
```

子查询依赖于外层查询

4. 检索选修课程名为“数据库原理”的学生的学号和姓名。

```
SELECT S.SNO,SNAME      --最后在S表中找出这些学生的姓名
 FROM S
 WHERE SNO IN
       (SELECT SNO       --然后在SC表中找出选了该课程的学生的学号
        FROM SC
         WHERE CNO IN
             (SELECT CNO --首先在C表中找出数据库原理课程的课程号
               FROM  C
               WHERE CNAME='数据库原理'));
```

5. 检索选修课程号为C2和C4的学生的学号。

```
SELECT X.SNO
FROM SC X,SC Y
WHERE X.SNO=Y.SNO
   AND X.CNO='C2'
   AND Y.CNO='C4';
```

自身联接

派生表

或：

```
SELECT X.SNO
FROM (SELECT SNO FROM SC WHERE CNO='C2') X,
     (SELECT SNO FROM SC WHERE CNO='C4') Y
WHERE X.SNO=Y.SNO;
```

6. 检索没有选修C2课程的学生的姓名和年龄。

```
SELECT SNAME,AGE          ----找这样的学生
 FROM S
 WHERE NOT EXISTS
       (SELECT *           ----- 不存在
        FROM SC
         WHERE SC.SNO=S.SNO   --- 他（即找的这个学生）
           AND SC.CNO='C2');   --- 选了 'C2'课程
```

7. 检索选修了全部课程的学生的姓名。- -查询变换依据：

假设：cno：　　表示课程；

p(cno)：　表示“选修了课程cno”；

¬p(cno)：表示“没有选修课程cno”

形式化表示：

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

变换后的语义：

找这样的学生SNO，**不存在一门课程，**他**没有选修。**

7. 检索选修了全部课程的学生的姓名。-- 变换为：
找这样的学生SNO，不存在一门课程，他没有选修。

```
SELECT SNAME                    ----找这样的学生
 FROM S
 WHERE NOT EXISTS
       (SELECT *                ----不存在一门课程
        FROM C
         WHERE NOT EXISTS
                    (SELECT *
                     FROM SC
                     WHERE SC.SNO=S.SNO
                        AND SC.CNO=C.CNO));-- 选修
```

他没有

8. 检索至少选修了学生S2所选修的全部课程的学生的学号。

变换表达形式：找这样的学生SNO，对所有的课程CNO，
只要学生S2选修了，那么学生SNO也选修了这些课程CNO。

形式化表示：$(\forall cno)\ p(cno) \rightarrow q(cno)$

其中：cno：表示课程；

p(cno)：表示 “学生S2选修了课程cno”；

q(cno)：表示 “学生SNO选修了课程cno”

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

变换后的语义：找这样的学生SNO，**不存在一门课程cno，学生S2选修了，而学生sno没选。**

8.检索至少选修了学生S2所选修的全部课程的学生的学号。

```
SELECT DISTINCT SNO          ----找这样的学生
 FROM SC X
 WHERE NOT EXISTS
       (SELECT *              ----不存在一门课程
        FROM SC Y
         WHERE Y.SNO='S2'     --- 学生S2选了
           AND NOT EXISTS
               (SELECT *
                FROM SC Z
                WHERE Z.SNO=X.SNO
                  AND Z.CNO=Y.CNO));
```

而这个学生没选

8.检索至少选修了学生S2所选修的全部课程的学生的学号。

```
SELECT DISTINCT
 FROM SC X
 WHERE NOT EXIS
       (SELECT *
        FROM SC
         WHERE
           AND
```

如要在结果集中不包含学生S2的SQL语句如下：

```
SELECT DISTINCT SNO
 FROM (SELECT * FROM SC WHERE SNO<>'S2')  X
 WHERE NOT EXISTS
       (SELECT *
        FROM SC Y
         WHERE Y.SNO='S2'
           AND NOT EXISTS
                (SELECT *
                 FROM SC Z
                 WHERE Z.SNO=X.SNO
                   AND Z.CNO=Y.CNO));
```

四、聚合函数

SQL 提供了下列聚合函数：

COUNT（*）	计算元组的个数
COUNT（列名）	对一列中的值计算个数
SUM（列名）	求某一列值的总和（此列的值必须是数值）
AVG（列名）	求某一列值的平均值（此列的值必须是数值）
MAX（列名）	求某一列值的最大值
MIN（列名）	求某一列值的最小值

举例：

1. 检索年龄最大的学生的姓名和性别。

```
SELECT   SNAME,SEX
FROM   S
WHERE AGE=(SELECT MAX(AGE)
              FROM S);
```

2. 检索选修两门以上课程的学生的学号。

```
SELECT SNO,COUNT(*) COUNT
FROM SC
GROUP BY SNO HAVING COUNT(*)>2;
```

3.统计每个学生选修课程的门数（超过2门的学生才统计）。

要求显示学生的学号、姓名和选修课程门数，查询结果按门数降序排列，若门数相同，按学号升序排列：

```
SELECT S.SNO 学号,SNAME 姓名,COUNT(CNO) 课程门数
    FROM SC,S
    WHERE S.SNO=SC.SNO
      GROUP BY S.SNO,SNAME
          HAVING COUNT(*)>2
       ORDER BY 3 DESC,1;
```

SELECT语句的语义通常有三种情况（SQL标准）：

以学生表S（SNO，SNAME，AGE，SEX，SDEPT）为例说明。

第一种情况：SELECT语句中未使用分组子句,也未使用聚合操作，那么SELECT子句的语义是对查询的结果执行投影操作。如：

```
SELECT SNO,SNAME
FROM S
WHERE SEX=’男’;
```

第二种情况：SELECT语句中未使用分组子句，但在SELECT子句中使用了聚合操作，此时SELECT子句的语义是对查询结果执行聚合操作。如：

```
SELECT   COUNT(*)   count_男,   AVG(CAST(S.AGE AS INT))  avg_age
  FROM S
  WHERE SEX='男';
```

该语句是求男同学的人数和平均年龄（假设AVG为字符型）

第三种情况：SELECT语句使用了分组子句和聚合操作，此时SELECT子句的语义是对查询结果的每一分组去做聚合操作。如：

```
SELECT AGE,COUNT(*) count
       FROM S
       WHERE SEX=’男’
       GROUP BY AGE;
```

该语句是求男同学每一年龄的人数。

五. 联接操作

联接条件可在WHERE中指定,也可以在 FROM子句中指定。

在 FROM 子句中指定联接条件时，SQL2开始将联接操作符分成：联接类型、联接条件。

联接类型：决定了如何处理联接条件中不匹配的元组。

联接条件：决定了两个关系中哪些元组应该匹配。

联接类型中的OUTER字样可不写。

联接类型	联接类型说明
INNER JOIN ⋈	内联接：结果为两个联接表中匹配行的联接。
LEFT OUTER JOIN 	左外联接：结果包括“左”表（出现在 JOIN 子句的最左边）中的所有行。不包括右表中的不匹配行。
RIGHT OUTER JOIN 	右外联接：结果包括“右”表（出现在JOIN 子句的最右边）中的所有行。不包括左表中的不匹配行。
FULL OUTER JOIN 	完全外联接：结果包括所有联接表中的所有行，不论它们是否匹配
CROSS JOIN	交叉联接：结果包括两个联接表中所有可能的行组合。交叉联接返回的是两个表的笛卡儿积

A	B	C
a	b	c
b	b	f
c	a	d

关系R

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

关系S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

R ⋈ S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null

R ⋈ S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
null	e	f	g

R ⋈ S

联接条件	联接条件说明
ON　联接条件	具体列出两个关系在哪些相应属性上做联接条件比较。联接条件应写在联接类型的右边。

例1：　　统计每个学生选修课程的门数（超过2门的学生才统计）。要求显示学生的学号、姓名和选修门数，查询结果按门数降序排列，若门数相同，按学号升序排列：

```
SELECT S.SNO,SNAME,COUNT(CNO) AS COUNTER
FROM S,SC
WHERE S.SNO=SC.SNO
GROUP BY S.SNO,SNAME
HAVING COUNT(*)>2
ORDER BY 3 DESC,1;
```

```
SELECT S.SNO,SNAME,COUNT(CNO) AS COUNTER
FROM S,SC
WHERE S.SNO=SC.SNO
GROUP BY S.SNO,SNAME
HAVING COUNT(*)>2
ORDER BY 3 DESC,1;
```

内连接

```
SELECT S.SNO, S.SNAME, COUNT(SC.CNO) AS COUNTER
FROM S INNER JOIN
     SC ON SC.SNO = S.SNO
GROUP BY S.SNO, S.SNAME
HAVING (COUNT(*) > 2)
ORDER BY COUNT(SC.CNO) DESC, S.SNO
```

例2:检索选修了课程号为C3的学生的情况。

```
SELECT *
FROM  S INNER JOIN
      SC ON S.SNO = SC.SNO
WHERE (SC.CNO = 'C3')
```

为显示更多的信息：

```
SELECT *
FROM S FULL OUTER JOIN
     (SELECT * FROM SC WHERE SC.CNO='C3') SC1
       ON  S.SNO=SC1.SNO
```

例2:检索选修了课程号为C3的学生的情况。

```
SELECT *
FROM  S INNER JOIN
      SC ON S.SNO =
WHERE (SC.CNO = 'C3'
```

为显示更多的信息：

```
SELECT *
FROM S FULL OUTER JOI
     (SELECT * FROM S
       ON  S.SNO=SC1.
```

```
SELECT *
FROM  S INNER JOIN
      SC ON S.SNO = SC.SNO
WHERE (SC.CNO = 'C3')


SELECT *
FROM S FULL OUTER JOIN
    (SELECT * FROM SC WHERE SC.CNO='C3') SC1
     ON  S.SNO=SC1.SNO
```

100 %

结果 消息

	SNO	SNAME	SEX	AGE	SDEPT	LOGN	PSWD	FEES1	FEES	SNO	CNO	GRADE	POINT
1	S1	李 铭	男	19	计算机软件	S1	S1	0	360	S1	C3	85	3.7
2	S4	张 鹰	女	21	计算机软件	S4	S4	0	360	S4	C3	67	1.7
3	S5	刘竟静	女	22	计算机软件	S5	S5	0	0	S5	C3	91	4

	SNO	SNAME	SEX	AGE	SDEPT	LOGN	PSWD	FEES1	FEES	SNO	CNO	GRADE	POINT
1	S1	李 铭	男	19	计算机软件	S1	S1	0	360	S1	C3	85	3.7
2	S2	刘晓明	男	20	计算机应用	S2	S2	0	240	NULL	NULL	NULL	NULL
3	S3	李 明	男	22	计算机应用	S3	S3	0	240	NULL	NULL	NULL	NULL
4	S4	张 鹰	女	21	计算机软件	S4	S4	0	360	S4	C3	67	1.7
5	S5	刘竟静	女	22	计算机软件	S5	S5	0	0	S5	C3	91	4
6	S6	刘成刚	男	21	计算机软件	S6	S6	0	0	NULL	NULL	NULL	NULL
7	S7	王 铭	男	22	计算机应用	S7	S7	0	0	NULL	NULL	NULL	NULL
8	S8	宣明尼	女	18	计算机应用	S8	S8	0	0	NULL	NULL	NULL	NULL

§4 SQL的数据更新

SQL的数据更新包括：

数据插入、修改数据和数据删除等操作。

一、数据插入

1. 插入单个元组：

INSERT INTO 基本表名（列名表）

VALUES （元组值）

例：

```
INSERT INTO SC(SNO,CNO)
VALUES ('S3','C3')
```

2. 插入子查询的结果：

INSERT INTO 基本表名（列名表）

SELECT 查询语句；

例：

```
CREATE TABLE S_AVG_GRADE
        (SNO CHAR(4),
         AVGGRADE NUMERIC(7,2));
INSERT INTO S_AVG_GRADE
        SELECT SNO,AVG(GRADE)
        FROM SC
        GROUP BY SNO;
```

二、数据删除

删除关系中满足条件的元组语句的句法如下：

DELETE　FROM　〈表名〉

WHERE　〈条件表达式〉

举例：

例：把课程名为”数据结构”的选课从表SC中删除.

```
DELETE  FROM  SC
  WHERE CNO IN
    (SELECT CNO FROM C
      WHERE CNAME='数据结构');
```

三、数据修改

UPDATE 基本表名

SET 列名=值表达式[，列名=值表达式...]

[WHERE 条件表达式]

例：把课程名为”数据库原理”的成绩提高10%.

```
UPDATE   SC
  SET GRADE=1.1*GRADE
  WHERE CNO IN (SELECT CNO FROM C
                        WHERE CNAME='数据库原理');
```

§5 视图操作

在SQL中，外模式一级数据结构的基本单位是视图（View）。

视图是从若干基本表和（或）其他视图构造出来的表。

在创建一个视图时，系统把视图的定义存放在数据字典中，而不存储视图对应的数据，在用户使用视图时才去求对应的数据。 视图被称为“虚表”。

一、 视图的创建

CREATE VIEW 〈视图名〉（列名表）

AS 〈SELECT 查询语句〉

例1：在基本表SC上，建立一个学生学习情况视图，

内容包括：学号、选修课程门数、平均成绩。

```
CREATE VIEW  S_GRADE(SNO,C_NUM,AVG_GRADE)
       AS (SELECT SNO,COUNT(CNO),AVG(GRADE)
            FROM SC
            GROUP BY SNO);
```

例2：在基本表SC上，建立一个学生学习成绩等级视图：

```
CREATE VIEW SC_等级(SNO,SNAME,CNO,CNAME,成绩等级)
 AS   (SELECT S.SNO, SNAME, C.CNO, CNAME,
        成绩等级 = CASE
            WHEN GRADE IS NULL THEN '未登分'
            WHEN GRADE < 60 THEN '不及格'
            WHEN GRADE >= 60 AND  GRADE < 70 THEN '及格'
            WHEN GRADE >= 70 AND GRADE < 80 THEN '中'
            WHEN GRADE >= 80 AND GRADE < 90 THEN '良'
            ELSE '优' END
      FROM S INNER JOIN SC ON S.SNO = SC.SNO
                INNER JOIN  C ON SC.CNO = C.CNO)
```

例2：在基本表SC上，建立一个学生学习成绩等级视图：

```
CREATE VIEW SC_等级(SNO,SNAME,CNO,CNAME,成绩等级)
 AS   (SELECT S.SN
        成绩等级 = CA
            WHEN GR
            WHEN GR
            WHEN GR
            WHEN GR
            WHEN GR
            ELSE '优'
      FROM S INNE
          INNER JOIN  C ON SC.CNO = C.CNO)
```

DESKTOP-Q7DDM...T - dbo.SC_等级

SNO	SNAME	CNO	CNAME	成绩等级
S1	李 铭	C1	高级语言程序设计	未登分
S1	李 铭	C3	离散数学	良
S1	李 铭	C4	计算机原理	中
S1	李 铭	C6	Windows技术	及格
S1	李 铭	C8	编译原理	良
S1	李 铭	C9	系统结构	未登分
S2	刘晓明	C1	高级语言程序设计	未登分
S3	李 明	C1	高级语言程序设计	不及格
S3	李 明	C4	计算机原理	良
S4	张 鹰	C1	高级语言程序设计	及格
S4	张 鹰	C3	离散数学	及格
S5	刘竟静	C1	高级语言程序设计	及格
S5	刘竟静	C3	离散数学	优
S5	刘竟静	C5	数据库原理	中
S6	刘成刚	C1	高级语言程序设计	中

1 / 15

二、 视图的撤消

DROP VIEW 视图名

例：撤消 S_GRADE 视图，可用下列语句实现：

```
DROP VIEW S_GRADE;
```

三、 视图的查询

系统在实现对视图的查询时，根据数据字典的定义将对视图的查询转换为对基本表的查询。

例：对学生学习情况视图执行如下操作：

① SELECT * FROM S_GRADE;

相应的查询转换操作如下：

```
SELECT SNO,COUNT(CNO) AS C_NUM,AVG(GRADE) AS AVG_GRADE
 FROM SC
 GROUP BY SNO;
```

②　SELECT SNO,C_NUM

FROM S_GRADE

WHERE AVG_GRADE>80;

相应的查询转换操作如下：

SELECT SNO,COUNT(CNO) AS C_NUM

FROM SC

GROUP BY SNO

HAVING AVG(GRADE)> 80;

③
```
SELECT SNO,AVG_GRADE
FROM S_GRADE
 WHERE C_NUM >(SELECT C_NUM
                FROM S_GRADE
                WHERE SNO='S4');
```

相应的查询转换操作如下：

```
SELECT SNO,AVG(GRADE) AS AVG_GRADE
       FROM SC
       GROUP BY SNO
        HAVING COUNT(CNO)>(SELECT COUNT(CNO)
                           FROM SC
                           GROUP BY SNO
                           HAVING SNO='S4');
```

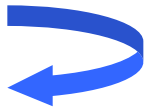
四、视图的更新操作

对于视图的更新操作(INSERT、DELET、UPDATA)有以下三条规则：

① 如果视图是从单个基本表使用选择、投影操作导出的，并且包含了基本表的主键或某个候选键，那么这样的视图称为“行列子集视图”，并且可以被执行更新操作。

② 如果在导出视图的过程中，使用了分组和聚合操作，不允许对这个视图执行更新操作。

③ 如果一个视图是从多个基本表使用联接操作导出的，通常不允许对这个视图执行更新操作。



四、视图的更新操作

对于视图的更新操作(INSERT、DELET、UPDATA)有以下三条规则：

① 如果一个视图是从多个基本表使用联接操作导出的，那么不允许对这个视图执行更新操作。

② 如果在导出视图的过程中，使用了分组和聚合操作，也不允许对这个视图执行更新操作。

③ 如果视图是从单个基本表使用选择、投影操作导出的，并且包含了基本表的主键或某个候选键，那么这样的视图称为“行列子集视图”，并且可以被执行更新操作。

在SQL2中，允许更新的视图在定义时，必须加上“WITH CHECK OPTION”短语。

例：如果定义“计算机应用”学生视图：

```
CREATE VIEW STUDENT_COMPUTER(SNO,SNAME,SEX,AGE)
    AS SELECT SNO,SNAME,SEX,AGE
       FROM S
       WHERE SDEPT=‘计算机应用’
```

该视图是从单个关系仅使用了选择和投影导出的，而且包括键SNO，因此是可以修改的。

如执行插入操作：

```
INSERT INTO STUDENT_COMPUTER
 VALUES(‘S99’, ‘王敏’, ‘男’,22);
```

系统自动会把它转换变成下列语句：

```
INSERT  INTO  S
VALUES（’S99’,‘王敏’,‘男’,22,‘计算机应用’ ）;
```

对于学生学习情况视图：

```
CREATE VIEW S_GRADE(SNO, C_NUM, AVG_GRADE)
        AS SELECT SNO, COUNT(CNO), AVG（GRADE）
            FROM SC
            GROUP BY SNO
```

执行：UPDATE S_GRADE

SET SNO='S3'

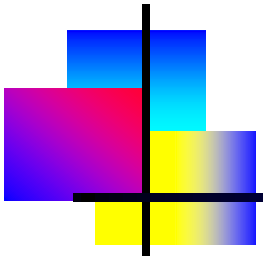
WHERE SNO='S4'；

不允许。C_NUM是对SC中的学生选修门数进行统计，在未更改SC表时，要在视图S_GRADE中更改门数，是不可能的。

执行： DELETE FROM S_GRADE

WHERE C_NUM>4；

也不允许的。在视图S_GRADE中删除选修门数在4门以上的学生元组，势必造成SC中这些学生学习元组的删除，这不一定是用户的原意，因此使用分组和聚合操作的视图，不允许用户执行更新操作。



§6　嵌入式SQL的使用技术

♥ 嵌入式 SQL

SQL语言有两种使用方式：

一种是在终端交互方式下使用，称为交互式SQL；

另一种是嵌入在高级语言的程序中使用，称为嵌入式SQL，而这些高级语言可以是C、PASCAL、COBOL或PL／I等称为宿主语言。

使用差别：SQL是基于关系数据模型的语言；

高级语言是基于基本数据类型（整型、

字符串型、记录、数组等）的语言。

譬如，SQL语句不能直接使用指针、数组等数据结构；

高级语言一般不能直接进行集合的操作。

为了能在宿主语言的程序中嵌入SQL语句，

必须作某些规定。

嵌入式SQL的实现，有两种处理方式：

1、扩充宿主语言的编译程序，使之能处理SQL语句；

2、采用预处理方式。

目前多数系统采用后一种方式。

预处理方式是：

先由预处理程序对源程序进行扫描，识别出SQL语句，并处理成宿主语言的函数调用形式；

然后再用宿主语言的编译程序把源程序编译成目标程序。

通常DBMS制造商提供一个SQL函数定义库，供编译时使用。源程序的预处理和编译的具体过程如下图所示：

宿主语言十嵌入式SQL

⇓

预 处 理 程 序

⇓

宿主语言十函数调用

⇓

宿主语言编译程 序

⇓

目标程序

一、嵌入式SQL使用时必须解决的问题

①为区分SQL语句与宿主语言语句，在所有的SQL语句前必须加上前缀标识“EXEC SQL”，并以“END EXEC”作为语句结束标志。

格式：　EXEC SQL 〈SQL语句〉 END_EXEC

结束标志在不同的宿主语言中是不同的，在C和PASCAL语言程序中规定结束标志不用END_EXEC，而使用分号“；”。

② 数据库工作单元和主程序工作单元之间的通讯

允许嵌入的SQL语句引用宿主语言的程序变量（称为共享变量）。 并规定：

- ♥ 在引用这些变量时必须在这些变量前加冒号“ ：”作为前缀标识，以示与数据库中变量有区别；
- ♥ 这些变量由宿主语言的程序定义；
- ♥ 由SQL的BEGIN DECLARE SECTION 与END DECLARE SECTION语句之间说明。

而主语言不能引用数据库中的字段变量。

♥SQL2规定，SQLSTATE是一个特殊的共享变量，

起着解释SQL语句执行状况的作用，是一个由5个字符组成的字符数组。

SQLSTATE = 0 :表示SQL语句执行成功；

SQLSTATE <> 0 :表示执行SQL语句时发生的各种特殊情况。

譬如“02000”用来表示未找到元组。

在执行一个SQL语句后，程序可以根据SQLSTATE的值转向不同的分支，以控制程序的流向。

③ 引入游标机制：

将集合操作转换为单元组处理。

SQL的执行是面向集合的，一条SQL语句原则上可以产生或处理多条记录。

宿主语言是面向记录的，一组主变量一次只能存放一条记录。

为此引入游标来协调SQL语言与主语言的不同数据处理方式。

二、与游标有关的语句：

① 定义游标语句（ DECLARE ）

游标是与某一查询结果相联系的符号名。

EXEC SQL DECLARE <游标名> CURSOR FOR

<SELECT语句>

[FOR UPDATE [OF <字段名1> <，...,字段名n>]];

END_EXEC

这是一个说明语句,定义中的SELECT语句并不立即执行。

```
EXEC SQL DECLARE scx CURSOR FOR
    SELECT SNO,CNO,GRADE
    FROM SC
    WHERE SNO= (SELECT SNO
                FROM S
                WHERE SNAME=:givensname)
     FOR UPDATE  OF  GRADE;
```

② 打开游标语句（OPEN）

打开游标语句使游标处于活动状态。与游标相应的查询语句被执行。游标指向查询结果的第一个记录之前。 句法如下：

EXEC SQL OPEN〈游标名〉END_EXEC

③ 游标推进语句（FETCH）

此时游标推进一个记录，并把游标指向的记录（称为当前行）中的值取出，送到INTO子句后相应的主变量中。句法如下：

```
EXEC SQL FETCH FROM〈游标名〉INTO〈变量表〉
END_EXEC
```

FETCH语句常用于宿主语言程序的循环结构中，并借助宿主语言的处理语句逐一处理查询结果中的一个个元组。在游标处于活动状态时,可以修改和删除游标指向的元组。

④ 关闭游标语句（CLOSE）

关闭游标，使它不再和查询结果相联系。关闭了的游标，可以再次打开，与新的查询结果相联系。句法如下：

EXEC SQL CLOSE〈游标名〉END_EXEC

三、嵌入式SQL的使用技术

(1) 在嵌入式SQL中， SQL的数据定义DDL与控制语句DCL都不需要使用游标。

它们是嵌入式SQL中最简单的一类语句，不需要返回结果数据，也不需要使用主变量。在主语言中嵌入SQL说明性语句（DECLARE）及控制语句（GRANT），只要给语句加上前缀EXEC SQL和语句结束符END_EXEC即可。在C语言中，用分号；代替END_EXEC

例：在C语言中说明共享变量：

```
EXEC SQL BEGIN DECLARE SECTION
 int grade, rise;
 char givencno[5],cname[13], tname[9] ;
 char givensno[5],sname[9],sdept[11];
 char SQLSTATE[6];
 EXEC SQL END DECLARE SECTION;
```

(2) 不涉及游标的嵌入式SQL DML语句

a. 对于INSERT、DELETE和UPDATE语句， 只要加上前缀标识“EXEC SQL”和结束标志“END_EXEC”，就能嵌入在宿主语言程序中使用。例：

① 在关系C中插入一门新的课程，各属性值已在相应的共享变量中：

```
EXEC SQL INSERTER  INTO C（CNO,CNAME,TNAME）
                VALUED(:givencno,:cname,:tname);
```

② 从关系SC中删除一个学生的所有选课，

该学生的姓名由共享变量sname提供。

```
EXEC SQL DELETE    FROM SC
                WHERE SNO=(SELECT SNO
                                FROM S
                                WHERE SNAME=:sname);
```

③ 把“数据库”课程的全部成绩增加某个值（该值由共享变量raise 提供）。

```
EXEC SQL UPDATE SC
        SET GRADE=GRADE +:rise
        WHERE CNO IN
           (SELECT CNO
             FROM C
             WHERE CNAME=‘数据库’);
```

b. 对于SELECT语句，如果已知查询结果肯定是单元组时，可直接嵌入在主程序中使用，此时在SELECI语句中增加一个INTO子句，指出找到的值应送到相应的共享变量中去。

例：在关系S中根据共享变量givensno的值检索学生的姓名和所在系。

例：在关系S中根据共享变量givensno的值检索该学生的姓名和所在系：

```
EXEC SQL   SELECT SNAME,SDEPT
                INTO :sname,:sdept
                FROM S
                WHERE SNO=:givensno;
```

此处sname,sdept,givensno 都是共享变量，已在主程序中定义，并用SQL的DECLARE语句加以说明，在引用是加上“：”作为前缀标识，以示与数据库中变量区别。

(3) 涉及游标的嵌入式SQL DML语句

a、当SELECT语句查询结果是多个元组时，此时要用游标机制把多个元组一次一个地传送给宿主语言程序处理。

例:在关系SC表中检索某学生(学生名由共享变量givensname给出)选课信息（SNO,CNO,GRADE），

该查询的C语言程序段：

```
EXEC SQL BEGIN DECLARE SECTION;
   Int grade, rise;
   Char sno[5],cno[5], givensname[9], SQLSTATE[6];
EXEC SQL END DECLARE SECTION;
EXEC SQL DECLARE scx CURSOR FOR
     SELECT SNO,CNO,GRADE
     FROM SC
     WHERE SNO= (SELECT SNO
                 FROM S
                 WHERE SNAME=:givensname)
     FOR UPDATE  OF  GRADE;
EXEC SQL OPEN scx;
```

```
While(1)
  { EXEC SQL FETCH FROM scx
                INTO :sno,:cno,:grade;
  If (SQLCA.SQLSTATE = ='02000') /*已取完查询结果中的所有元组*/
      Break;
  If (SQLCA.SQLSTATE != '0')    /*取数据出错*/
     Break;
      ...                        /*对游标所取的数据进行处理*/
      printf("%s,%s,%d",sno,cno,grade);
}
EXEC SQL CLOSE scx;
```

b. 对游标指向的元组进行修改或删除操作

当游标处于活动状态时，可以修改或删除游标指向的元组。

例：在上面的例子中，对找到的元组做如下处理：

删除不及格的选课，将60~69分的成绩增加由共享变量rise提供的值，再显示该学生的成绩信息(SNO,CNO,GRADE).

在上例中的”While(1){...}语句改为如下形式：

```
While(1)
{ EXEC SQL FECCH FROM scx
          INTO  :sno,:cno,:grade;
If (SQLCA.SQLSTATE = =’02000’) /*已经取完查询结果中的所有元组*/
   Break;
If (SQLCA.SQLSTATE ! = ’0’)   /*  取数据出错*/
    Break;
    If (grade<60)
        EXEC SQL DELETE FROM SC
                 WHERE CURRENT OF scx;
    Else
        {If (grade<70)
```

```
    {EXEC SQL UPDATE C
                SET GRADE=GRADE+:rise
                WHERE CURRENT OF scx;
                grade=grade+rise;}
        printf("%s,%s,%d",sno,cno,grade);
    }
}
```

四、动态SQL语句

在实际问题中，用户对数据库的操作有时在系统运行时才能提出来，这时要用到嵌入式SQL的动态技术才能实现。

动态SQL技术主要有两个SQL语句：

1、动态SQL预备语句：

EXEC SQL PREPARE〈动态 SQL语句名〉FROM〈共享变量或字符串〉

共享变量或字符串的值应是一个完整的SQL语句。

该语句可在程序运行时与用户的输入组合起来。

此时，这个语句并不执行。

2、动态SQL执行语句

EXEC SQL EXECUTE <动态 SQL 语句名>

动态SQL语句使用时，可以有两点改进：

(1)当预备语句中组合而成的SQL语句只需执行一次时，那么预备语句和执行语句可合并成一个语句：

EXEC SQL EXECUTE IMMEDIATE <共享变量或字符串>

(2)当预备语句中组合而成的SQL语句的条件值尚缺时，可以在执行语句中用USING短语补上：

EXEC SQL EXECUTE <动态 SQL语句名> USING <共享变量>

例①: EXEC SQL BEGIN DECLARE SECTION;

char *query;

EXEC SQL END DECLARE SECTION;

scanf（”s%”, query）; /*从键盘输入一个 SQL语句*/

EXEC SQL PREPARE que FROM : query;

EXEC SQL EXECUTE que;

该程序段表示:

从键盘输入一个SQL语句到字符数组中,

字符指针query指向字符串的第1个字符。

如果执行语句只做一次，程序段最后两个语句可合并成一个语句:

EXEC SQL EXECUTE IMMEDIATE : query;

② char *query=”UPDATE SC

SET GRADE=GRADE*1.3

WHERE SNO=? ”;

```
EXEC SQL PREPARE dynprog FROM :query;
char sno[5]=”S2”;
EXEC SQL EXECUTE dynprog USING :sno;
```

第一个char语句表示由用户组合成一个SQL语句，但有一个值（学号）还不能确定，因此用“？”表示。

第二个语句是动态SQL预备语句。

第三个语句（char语句）表示取到了学号的值。

第四个语句是动态SQL执行语句括句,“？”值到共享变量sno中取。

精读和上机、习题要求

精 读：　教材　P.69~P.100　　P.250~P.297

P.314~P.355

上 机：　1.　实验一 ……P.298

2.　实验二 …… P.300

3.　实验三 （可自选开发环境）… … P.364

4.　实验四 （可模仿示例程序）… … P.365

习 题4：　P.101　　2 、6、7 、9