



第四章 结构化查询语言SQL（1）

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。



§ 0 前导知识

关系模型及其基本术语:

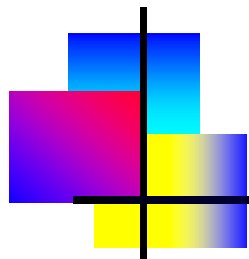
- ✚ 用二维表格表示实体集，外键表示实体间联系的模型称为关系模型；
- ✚ 关系：对应二维表格；
- ✚ 元组：表中的行；
- ✚ 属性：表中的列；



§ 0 前导知识

关键码

- + **超键：** 能唯一标识元组的属性组合（可能存在多余的属性）。
- + **候选键：** 能唯一标识元组的最小属性组合。
- + **主键：** 若一个关系中有多个候选键，则选其中的一个为关系的主键。
- + **外键：** 若一个关系R中包含有另一个关系S的主键所对应的属性组F， 则称F为R的外键。称关系S为参照关系， R为依赖关系。



§ 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。

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

-  模式的定义（数据库）
-  基本表的定义
-  索引的定义（自学）
-  视图的定义



§ 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
```

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),
```

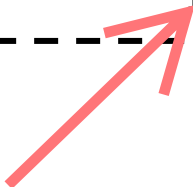
```
WEIGHT SMALLINT,
```

```
PRIMARY KEY(PNO));
```



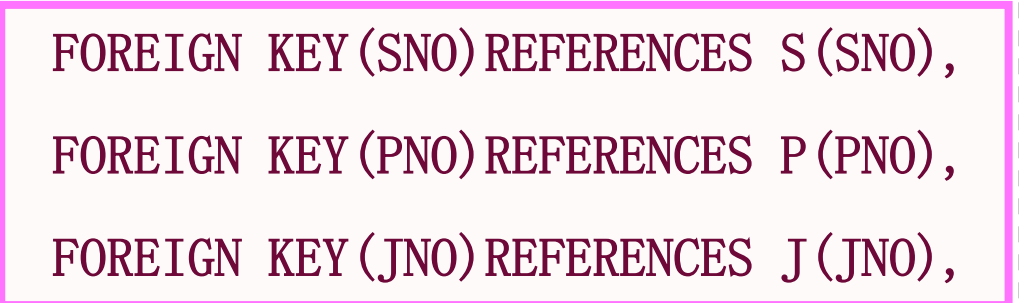

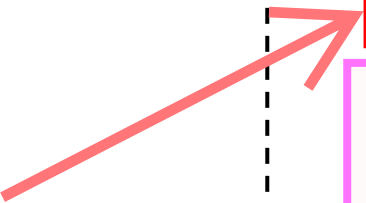
可用下列语句创建表J:

```
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),  
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(20),
```

```
SEX CHAR(2),
```

```
SDEPT CHAR(10),
```

```
PRIMARY KEY(SNO));
```

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));
```

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),

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;
```



§ 3 SQL的数据查询

SQL的数据查询(SELECT语句)是SQL的核心内容。

1. SELECT语句的来历

在关系代数中最常用的式子是下列表达式:

$$\pi A_1, \dots, A_n (\sigma_F (R_1 \times \dots \times R_m))$$

这里 R_1, \dots, R_m 为关系, F 是公式, A_1, \dots, A_n 为属性。

为此SQL设计成 **SELECT – FROM – WHERE**句型:

SELECT A_1, \dots, A_n

FROM R_1, \dots, R_m

WHERE F

2. SELECT语句格式:

SELECT [DISTINCT] 目标表的列名或列表达式序列



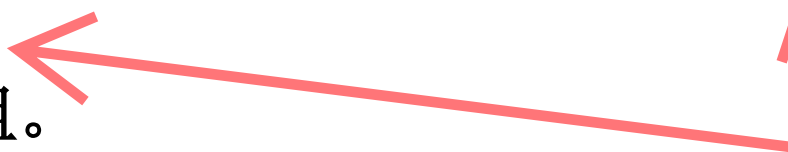


FROM 基本表名(或)视图名序列|表引用

[**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: $<$, \leq , $>$, \geq , $=$, \neq 或 \neq ;
- 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

● 单表查询

查询仅涉及一个表，是一种最简单的查询操作

- 选择表中的若干列
- 选择表中的若干元组
- 对查询结果排序
- 使用集函数
- 对查询结果分组

- 1、查询全体学生的姓名、学号、所在系。
- 2、查询全体学生的详细记录。
- 3、查全体学生的姓名及其出生年份。

别名的使用

- 4、查询选修了课程的学生学号。
- 5、查询计算机系全体学生的名单。
- 6、查询考试成绩有不及格的学生的学号。
- 7、查询年龄在20~23岁（包括20岁和23岁）之间的学生的姓名、系别和年龄。
- 8、查询年龄不在20~23岁之间的学生姓名、系别和年龄
- 9、查询信息系（IS）、数学系（MA）和计算机科学系（CS）学生的姓名和性别。

10、查询所有姓刘学生的姓名、学号和性别。

通配符

% (百分号) 代表任意长度（长度可以为0）的字符串。

例：a%b表示以a开头，以b结尾的任意长度的字符串。如acb, addgb, ab 等都满足该匹配串。

_ (下横线) 代表任意单个字符。

例：a_b表示以a开头，以b结尾的长度为3的任意字符串。如acb, afb 等都满足该匹配串。

11、查询姓“欧阳”且全名为三个汉字的学生的姓名

12、某些学生选修课程后没有参加考试，所以有选课记录，但没有考试成绩。查询缺少成绩的学生的学号和相应的课程号。

13、查询选修了3号课程的学生学号及其成绩，查询结果按分数降序排列。

- 联接操作（多表查询）

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



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



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



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。

也可以在WHERE子句中指定，一般为内联接

例1： 查询选修2号课程的学生姓名。	SELECT SNAME
SELECT SNAME	FROM SC INNER JOIN S
FROM S, SC	ON S.SNO=SC.SNO
WHERE S.SNO=SC.SNO AND CNO='2'	WHERE CNO='2'

2、检索选修了C1和C3的课程的学生的学号。

```
SELECT DISTINCT X.SNO
```

```
FROM SC X INNER JOIN
```

```
SC Y ON X.SNO = Y.SNO
```

```
WHERE (X.CNO = 'C1') AND (Y.CNO = 'C3');
```

3、查询每一门课的间接先修课（即先修课的先修课）。

```
SELECT X.Cno, Y.Cpno
```

```
FROM C X, C Y
```

```
WHERE X.Cpno = Y.Cno;
```

● 嵌套操作

例：查询选修2号课程的学生姓名。

```
SELECT SNAME  
FROM S  
WHERE SNO IN  
(SELECT SNO  
FROM SC  
WHERE CNO='2');
```

```
SELECT SNAME  
FROM S  
WHERE '2' IN  
(SELECT CNO  
FROM SC  
WHERE SNO=S.SNO)
```

```
SELECT SNAME  
FROM S  
WHERE EXISTS  
(SELECT *  
FROM SC  
WHERE SNO=S.SNO  
AND CNO='2');
```

嵌套查询分类

不相关子查询：子查询的查询条件不依赖于父查询

相关子查询：子查询的查询条件依赖于父查询

带有IN谓词的子查询

1、查询与“刘晨”在同一个系学习的学生。

```
SELECT Sno, Sname, Sdept
FROM S
WHERE Sdept IN
      (SELECT Sdept
       FROM S
       WHERE Sname= ' 刘晨 ' );
```

2、查询没有选修1号课程的学生姓名。

```
SELECT Sname  
FROM S  
WHERE SNO NOT IN  
(SELECT SNO FROM SC WHERE CNO='1')
```

3、查询选修了课程名为“信息系统”的学生学号和姓名

```
SELECT Sno, Sname  
FROM S  
WHERE Sno IN  
      (SELECT Sno  
       FROM SC  
       WHERE Cno IN  
             (SELECT Cno  
              FROM C  
              WHERE Cname= '信息系统' ));
```

带有比较运算符的子查询

- 使用范围

当能确切知道内层查询返回单值时，可用比较运算符（>，<，=，>=，<=，!=或<>）。

与**ANY**或**ALL**谓词配合使用

查询与“刘晨”在同一个系学习的学生。

```
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept =
        (SELECT Sdept
         FROM Student
         WHERE Sname= ' 刘晨  ');
```

注意：子查询一定要跟在比较符之后

带有ANY或ALL谓词的子查询

● 谓词语义

ANY: 任意一个值 **ALL:** 所有值

● 需要配合使用比较运算符

> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值
<= ANY	小于等于子查询结果中的某个值
<= ALL	小于等于子查询结果中的所有值
= ANY	等于子查询结果中的某个值
= ALL	等于子查询结果中的所有值（通常没有实际意义）
!= (或<>) ANY	不等于子查询结果中的某个值
!= (或<>) ALL	不等于子查询结果中的任何一个值

例： 查询其他系中比信息系某一学生年龄小的学生姓名和年龄

```
SELECT Sname, Sage
```

```
FROM S
```

```
WHERE Sage < ANY (SELECT Sage
```

```
FROM S
```

```
WHERE Sdept= ' IS ')
```

```
AND Sdept <> ' IS ' ;
```

/* 注意这是父查询块中的条件 */

```
SELECT Sname, Sage
```

```
FROM S
```

```
WHERE SDEPT!='IS' AND Sage < (SELECT MAX(SAGE)
```

```
FROM S
```

```
WHERE SDEPT='IS')
```


- **ANY和ALL谓词有时可以用集函数实现**
用集函数实现子查询通常比直接用ANY或ALL查询效率要高，因为前者通常能够减少比较次数
ANY与ALL与集函数的对应关系

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>= MIN
ALL	--	NOT IN	<MIN	<= MIN	>MAX	>= MAX

例：查询其他系中比信息系所有学生
年龄都小的学生姓名及年龄。

方法一：用ALL谓词

```
SELECT Sname, Sage
FROM Student
WHERE Sage < ALL
      (SELECT Sage
       FROM S
       WHERE
Sdept='IS')
      AND Sdept <>'IS';
```

方法二：用集函数

```
SELECT Sname, Sage
FROM S
WHERE Sage <
      (SELECT MIN(Sage)
       FROM S
       WHERE Sdept= 'IS ')
      AND Sdept <>'IS ';
```

带有**EXISTS**谓词的子查询

- **EXISTS**谓词
- **NOT EXISTS**谓词
- 用**EXISTS/NOT EXISTS**实现全称量词
- 用**EXISTS/NOT EXISTS**实现逻辑蕴涵

- 1. EXISTS谓词

存在量词 \exists

带有EXISTS谓词的子查询不返回任何数据，只产生逻辑真值“true”或逻辑假值“false”。

- 若内层查询结果非空，则返回真值
- 若内层查询结果为空，则返回假值

由EXISTS引出的子查询，其目标列表表达式通常都用*，因为带EXISTS的子查询只返回真值或假值，给出列名无实际意义

- 2. NOT EXISTS谓词

例： 查询所有选修了1号课程的学生姓名。

```
SELECT Sname
```

```
FROM S
```

```
WHERE EXISTS
```

```
  (SELECT *
```

```
   FROM SC
```

```
   WHERE Sno=S.Sno AND Cno= ' 1 ');
```

例:查询没有选修1号课程的学生姓名。

```
SELECT Sname
```

```
FROM S
```

```
WHERE NOT EXISTS
```

```
  (SELECT *
```

```
   FROM SC
```

```
    WHERE Sno = S.Sno AND Cno='1');
```

例：和刘晨在同一个系的学生学号、姓名。

```
SELECT Sno, Sname  
FROM S X  
WHERE EXISTS  
  (SELECT *  
   FROM S  
   WHERE Sdept = X.Sdept AND Sname = ‘刘晨’ );
```

- 用**EXISTS/NOT EXISTS**实现全称量词(难点)

SQL语言中没有全称量词 \forall (For all)

可以把带有全称量词的谓词转换为等价的带有存在量词的谓词:

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

例：查询选修了全部课程的学生姓名。

```
SELECT Sname
FROM S
WHERE NOT EXISTS
  (SELECT *
   FROM C
   WHERE NOT EXISTS
     (SELECT *
      FROM SC
      WHERE Sno= S. Sno
        AND Cno= C. Cno));
```

- 用**EXISTS/NOT EXISTS**实现逻辑蕴函(难点)
SQL语言中没有蕴函(**Implication**)逻辑运算
可以利用谓词演算将逻辑蕴函谓词等价转换为:

$$p \rightarrow q \equiv \neg p \vee q$$

例： 查询至少选修了学生95002选修的全部课程的学生号码。

解题思路：

- 用逻辑蕴涵表达：查询学号为x的学生，对所有的课程y，只要95002学生选修了课程y，则x也选修了y。

- 形式化表示：

用P表示谓词 “学生95002选修了课程y”

用q表示谓词 “学生x选修了课程y”

则上述查询为： $(\forall y) p \rightarrow q$

- 等价变换：

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

$$\equiv \neg (\exists y (\neg(\neg p \vee q)))$$

$$\equiv \neg \exists y(p \wedge \neg q)$$

- 变换后语义：不存在这样的课程y，学生95002选修了y，而学生x没有选。

```
SELECT DISTINCT Sno  
FROM SC X  
WHERE NOT EXISTS  
  (SELECT *  
   FROM SC Y  
   WHERE Y.Sno = ' 95002 ' AND  
    NOT EXISTS  
      (SELECT *  
       FROM SC Z  
       WHERE Z.Sno=X.Sno AND Z.Cno=Y.Cno));
```

•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
DISTINCT	

- 1、查询学生总人数。
- 2、查询选修了课程的学生人数。
- 3、计算1号课程的学生平均成绩。
- 4、求各个课程号及相应的选课人数。

SELECT语句的语义有三种情况（SQL标准）：

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(AGE) avg_age
```

```
FROM S
```

```
WHERE SEX='男' ;
```

该语句是求男同学的人数和平均年龄。

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 cno, count (*)
```

```
FROM SC
```

```
GROUP BY cno;
```

该语句是求每门课的选课人数。

注意：

- 通常SELECT语句中使用了分组子句就会有聚合操作。

但执行聚合操作不一定要用分组子句。

如：求男同学的人数，此时聚合值只有一个，因此不必分组。

- 但同一个聚合操作的值有多个时，必须使用分组子句。

如：求每一年龄的学生人数。此时聚合值有多个，

与年龄有关，因此必须分组。

- 1、查询学生总人数。
- 2、查询选修了课程的学生人数。
- 3、计算1号课程的学生平均成绩。
- 4、求各个课程号及相应的选课人数。
- 5、求各个课程号及相应的课程成绩在90分以上的学生人数。
- 6、查询选修了2门及以上课程的学生学号
- 7、查询选修了2门及以上课程的学生学号及选修门数，查询结果按门数降序排列，若门数相同，按学号升序排列
- 8、查询有3门以上课程在90分以上的学生的学号及90分以上的课程数。
- 9、查询所有成绩在80分以上的学生学号。
- 10、查询有成绩在90分以上的学生学号。

•集合运算

UNION

INTERSECT

EXCEPT