



# UNIT 7 复杂查询



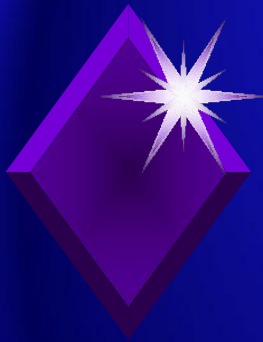
# 本讲主要目标

---



学完本讲后，你应该能够：

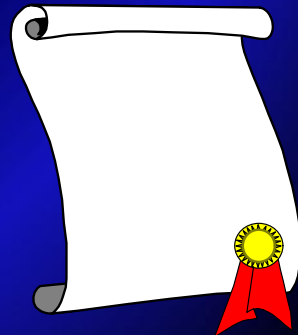
- 1、多个表放在FROM子句中，代表多表进行笛卡尔积。
- 2、出现在另一个SELECT语句之内的SELECT语句形式被称为子查询。两种子查询：相关子查询和不相关子查询
- 3、学会使用谓词IN、ANY、ALL和EXISTS；
- 4、学会使用UNION运算；
- 5、学会使用NOT EXISTS实现关系代数的除运算；
- 6、高级SQL语句的语法不统一，不适用于所有的数据库系统；
- 7、INTERSECT和EXCEPT是高级SQL提供的两个运算符，直接支持关系代数的“交”和“差”操作；
- 8、高级SQL的FROM子句中可以包含子查询或连接表；
- 9、SELECT语句的能力限制。



# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





# DreamHome 租赁数据库

---

DreamHome 案例的部分关系模式：

- ❁ **Branch** (branchNo, street, city, postcode)
- ❁ **Staff** (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
- ❁ **PropertyForRent** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
- ❁ **Client** (clientNo, fName, lName, telNo, prefType, maxRent)
- ❁ **PrivateOwner** (ownerNo, fName, lName, address, telNo)
- ❁ **Viewing** (clientNo, propertyNo, viewDate, comment)
- ❁ **Registration** (clientNo, branchNo, staffNo, dateJoined)



## DreamHome租赁数据库实例:

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005





## PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 8SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

## Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk



### PrivateOwner

ownerNo	fName	lName	address	telNo	eMail	password
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	jkeogh@lhh.com	*****
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	cfarrel@gmail.com	*****
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	tinam@hotmail.com	*****
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	tony.shaw@ark.com	*****

### Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	no dining room
CR56	PG36	28-Apr-13	

### Registration

clientNo	branchNo	staffNo	dateJoined
CR76	B005	SL41	2-Jan-13
CR56	B003	SG37	11-Apr-12
CR74	B003	SG37	16-Nov-11
CR62	B007	SA9	7-Mar-12



# 一个学生-课程数据库

S

学号 S#	姓名 SN	性别 SE	年龄 SA	所在系 SD
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

SC

学号 S#	课程号 C#	成绩 G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

C

课程号 C#	课程名 CN	先行课 CP#	学分 CC
C1	数据库	C5	4
C2	数学		2
C3	信息系统	C1	4
C4	操作系统	C6	3
C5	数据结构	C7	4
C6	数据处理		2
C7	PASCAL语言	C6	4





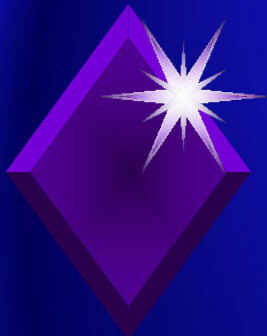
# 一、连接查询（见教材P84-88）

---

## 1、连接查询

—— 当查询的结果列来自多个表时，完成查询要求多个表进行连接操作。在FROM子句中将出现多个表，表名之间用逗号分隔。

➤ 完成关系代数的连接操作。



# 一、连接查询

---

## ➤ 语法：

**FROM** TableName [[**AS**] alias] [, ...]

**WHERE** join condition

◆ **表别名** alias 可在列名有歧义时用来指定列名，也可以用来作为表名的简写

◆ **连接条件** join condition 一般的格式是

[<表名1>. ]<列名1> <比较运算符> [<表名2>. ]<列名2>

也可以是下面的形式

[<表名1>. ]<列名1>BETWEEN[<表名2>. ]<列名2>AND[<表名2>. ]<列名3>



# 一、连接查询

---

## 2、连接运算的计算过程：

- ①形成FROM子句中指定表的笛卡尔积
- ②如果存在WHERE子句，对笛卡尔积运用查找条件进行行过滤
- ③执行SELECT子句，完成投影操作
- ④如果指定DISTINCT，则消除结果中的冗余行
- ⑤如果存在ORDER BY子句，则根据要求对查询结果排序



# 一、连接查询

## 3、简单连接

例7.1 列出查看过房产的所有客户的姓名及所提的意见

```
SELECT  c.clientNo, fName, lName,  
        propertyNo, comment  
FROM    Client c, Viewing v  
WHERE   c.clientNo = v.clientNo;
```

clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

类似于关系代数的等值连接



# 一、连接查询

## 4、排序连接结果

**例7.2** 对每一个分支机构，列出管理房产的职员姓名、编号及其正在管理的房产，并对结果按分支机构、员工编号、房产编号排序

```
SELECT    s.branchNo, s.staffNo, fName,
          lName, propertyNo
FROM      Staff s, PropertyForRent p
WHERE     s.staffNo = p.staffNo
ORDER BY  s.branchNo, s.staffNo, propertyNo;
```

branchNo	staffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14





# 一、连接查询

## 5、按多个列分组

**例7.3** 找出每一位职员管理的房产的数量，以及该员工所在分支机构的编号

```
SELECT    s.branchNo, s.staffNo,  
COUNT(*) AS myCount  
FROM      Staff s, PropertyFoRent p  
WHERE     s.staffNo = p.staffNo  
GROUP BY  s.branchNo, s.staffNo  
ORDER BY  s.branchNo, s.staffNo;
```

branchNo	staffNo	my count
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1



# 一、连接查询

## 6、等值连接与自然连接

例7.4 查询每个学生及其选修课程的情况。

```
SELECT S.*, SC.*  
FROM S, SC  
WHERE S.S# = SC.S#;
```

等值连接，卡  
氏积连接

例7.5 查询每个学生及其选修课程的情况。

```
SELECT S.*, SC.C#, SC.G  
FROM S, SC  
WHERE S.S# = SC.S#;
```

去掉重复的属性，  
保留其它属性列，  
自然连接



# 一、连接查询

## 7、自身连接 (P86)

—— 一个表与其自己进行连接

例7.6 查询每一门课的间接先修课。

```
SELECT  FIRST.C#, SECOND.CP#  
FROM    C FIRST, C SECOND  
WHERE   FIRST.CP# = SECOND.C#;
```

表别名



# 一、连接查询

---

## 8、同时完成连接和选择

**例7.7** 查询选修了“C1”且成绩在90分以上的学生的姓名和学号

```
SELECT  S.SN, S.S#  
FROM    S, SC  
WHERE   S.S# = SC.S# AND  
        SC.C# = 'C1' AND G>90;
```



## 一、连接查询

---

**例7.8** 查询选修了“95001”所选修的某门课程的学生学号

```
SELECT  DISTINCT SC1.S#  
FROM    SC SC1, SC SC2  
WHERE   SC1.C# = SC2.C# AND SC2.S# = '95001';
```

表别名





# 一、连接查询

---

## 9、多表连接

**例7.9** 查询每个学生的学号、姓名、选修的课程名和成绩。

```
SELECT  S. S#, SN, CN, G
FROM    S, SC, C
WHERE   S. S# = SC. S# AND SC. C#=C. C#;
```



# 一、连接查询

## 10、SQL标准提供了下列可选的方式来指定连接

**前例7.5** 查询每个学生及其选修课程的情况。

```
SELECT S.*, SC.C#, SC.G  
FROM S JOIN SC ON S.S# = SC.S#;
```

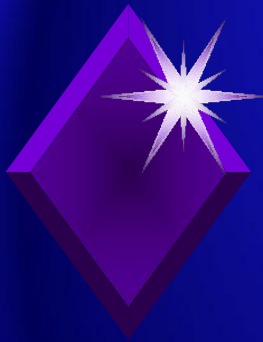
join...on :最为灵活,  
可以指明连接的条件

```
SELECT S.*, SC.C#, SC.G  
FROM S JOIN SC USING (S#) ;
```

natural join:指明了两表进  
行自然连接, 并且连接是基于  
两表中所有同名字段的

```
SELECT S.*, SC.C#, SC.G  
FROM S NATURAL JOIN SC ;
```

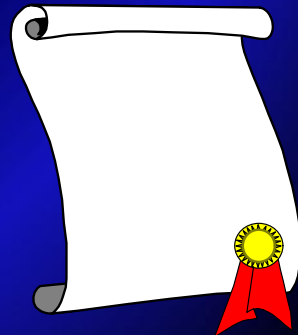
join...using:用于两表有同名字段  
但数据类型不同, 或者使用多个同名  
字段中的某一个做等值连接



# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





## 二、子查询 (见教材P88-93)

主查询/父查询

### 1、什么是子查询?

出现在另一个SELECT语句之内的SELECT语句形式被称为**子查询**。采用了子查询的查询称为**嵌套查询**。

内查询

**例7.10** 查询同时选修了课程“C2”和课程“C3”的学生的学号。

```
SELECT S#  
FROM SC  
WHERE C#='C2' AND  
       S# IN (SELECT S#  
              FROM SC  
              WHERE C#='C3');
```



## 二、子查询

### 2、子查询出现在SELECT的什么位置？

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〉 [, 〈列名〉 ]...  
[HAVING 〈内部函数表达式〉] ]  
[ORDER BY 〈列名〉 [ASC | DESC] [, 〈列名〉 [ASC | DESC]]...]

- SELECT语句中不能像关系代数一样，任意地将一个SELECT语句嵌入另一个SELECT语句；
- FROM子句中的表不能是SELECT语句的结果；
- 子查询可以被使用在外部SELECT语句的WHERE和HAVING子句中

● 子查询也可以出现在 INSERT, UPDATE, 和 DELETE语句中





## 二、子查询

---

### 3、子查询应遵循如下规则

- ① ORDER BY子句不能用于子查询
- ② 子查询总是括在圆括号中，作为表达式的一部分出现在条件比较运算符的右边，并且可以有选择的跟在IN, SOME (ANY), ALL和EXIST等谓词后面。
  - ◆带有比较运算符的子查询
  - ◆带有IN谓词的子查询
  - ◆带有ANY (SOME) 或ALL子查询
  - ◆带有EXIST的子查询



## 二、子查询

- ③ 子查询select列表必须由单个列名或表达式组成，除非子查询使用了关键字EXISTS
  - ◆ 返回单个值
  - ◆ 返回单个列、多个行
  - ◆ 返回多个列，多个行（带EXISTS）
- ④ 默认情况下，子查询中列名取自子查询的FROM子句中给定的表，也可以通过限定列名的办法指定取自外查询的FROM子句中的表
  - ◆ 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）



## 二、子查询

### 4、带有比较运算符的子查询

——子查询返回单个值时可以用比较运算符

例7.11 查询与‘刘晨’在同一个系学习的学生。

```
SELECT S#, SN, SD
FROM S
WHERE SD =
    ( SELECT SD
      FROM S
      WHERE SN = '刘晨'
    );
```

S

学号 S#	姓名 SN	性别 SE	年龄 SA	所在系 SD
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

```
SELECT S#, SN, SD
FROM S
WHERE SD = 'IS' ;
```

学号 S#	姓名 SN	所在系 SD
95002	刘晨	IS
95004	张立	IS



## 二、子查询

**例7.12** 列出个人工资高于平均工资的所有职员。

```
SELECT staffNo, fName, lName, position,  
FROM Staff  
WHERE salary >  
      (SELECT AVG(salary)  
       FROM Staff);
```

可以使用  
集合函数

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

❗ 不能写成 WHERE salary > AVG(salary)



## 二、子查询

**例7.13** 列出个人工资高于平均工资的所有职员，并求出多于平均数的值。

```
SELECT staffNo, fName, lName, position,  
       salary - (SELECT AVG(salary) FROM Staff) AS SalDiff  
FROM   Staff  
WHERE  salary >  
       (SELECT AVG(salary)  
        FROM   Staff);
```

staffNo	fName	lName	position	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00





## 二、子查询

### 5、带有IN谓词的子查询

--- 子查询返回单个值或单个列多个行时可以用IN

前例7.11 查询与‘刘晨’在同一个系学习的学生

```
SELECT  S#, SN, SD
FROM    S
WHERE   SD IN
        ( SELECT  SD
          FROM    S
          WHERE   SN = '刘晨'
        ) ;
```

带IN谓词的子查询是指父查询和子查询之间用IN谓词连接，判断某个属性列值是否在子查询的结果中



## 二、子查询

**例7.14** 查询选修了课程  
“C2”的学生的学号和姓名

```
SELECT S#, SN
FROM S
WHERE S# IN
    ( SELECT S#
      FROM SC
      WHERE C# = 'C2'
    ) ;
```

S

学号 S#	姓名 SN	性别 SE	年龄 SA	所在系 SD
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

SC

学号 S#	课程号 C#	成绩 G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

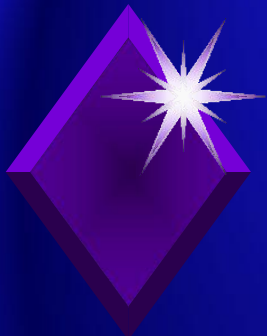


## 二、子查询

### 6、多层嵌套

**例7.15** 列出由位于“163 Main St”的分支机构的职员经营的房产。

```
SELECT propertyNo, street, city, postcode, type,  
        rooms, rent  
FROM    PropertyForRent  
WHERE   staffNo IN  
        (SELECT staffNo  
         FROM    Staff  
         WHERE   branchNo =  
                 (SELECT branchNo  
                  FROM    Branch  
                  WHERE   street = '163 Main St' ));
```



## 二、子查询

---

### 7、相关子查询和不相关子查询

#### (1) 不相关子查询

—— 子查询没有接受任何输入数据的情况下向外层的SELECT语句传递一个行集，即内层的子查询完全独立于外层的SELECT语句。

**不相关子查询的概念性执行顺序是：**先执行内层子查询，然后执行外层。

注：前面介绍的子查询中的例子均为不相关子查询。



## 二、子查询

**前例7.14** 查询选修了课程“C2”的学生的学号和姓名（第1种方法）

```
SELECT S#, SN
FROM S
WHERE S# IN
    ( SELECT S#
      FROM SC
      WHERE C# = 'C2'
    ) ;
```

S

学号 S#	姓名 SN	性别 SE	年龄 SA	所在系 SD
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

SC

学号 S#	课程号 C#	成绩 G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80





## 二、子查询

---

### (2) 相关子查询

—— 一个要使用外层SELECT语句所提供的数据的子查询

**相关子查询的执行过程：**假定内层子查询使用外层SELECT语句的变量X，则对于X的每一个取值，都执行一次内层子查询。



## 二、子查询

---

**例7.16** 找出每个学生超过他自己已选课程平均成绩的课程号

```
SELECT  S#, C#  
FROM    SC x  
WHERE   G >=  
        ( SELECT AVG(G)  
          FROM    SC y  
          WHERE   x.S# = y.S#  
        ) ;
```

定义SC表的  
别名X

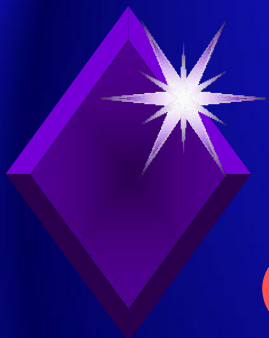


## 二、子查询

---

**前例7.14** 查询选修了课程“C2”的学生的学号和姓名  
(第2种方法)

```
SELECT S#, SN
FROM S
WHERE 'C2' IN
    ( SELECT C#
      FROM SC
      WHERE S.S# = SC.S#
    ) ;
```



## 二、子查询

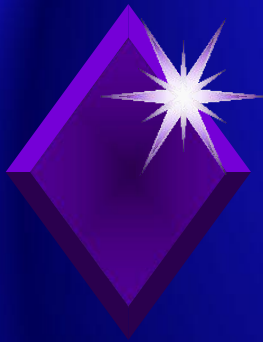
### (3) 相关和不相关子查询执行过程比较

- 不相关子查询执行过程：

- ① 执行子查询，其结果不被显示，而是传递给外部查询，作为外部查询的条件使用。
- ② 执行外部查询，并显示整个结果。

- 相关子查询执行过程：

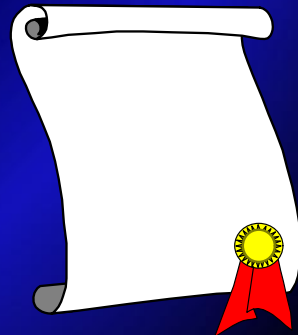
- ① 从外层查询中取出一个元组，将元组相关列的值传给内层查询。
- ② 执行内层查询，得到子查询操作的值。
- ③ 外查询根据子查询返回的结果或结果集得到满足条件的行。
- ④ 然后外层查询取出下一个元组重复做步骤1-3，直到外层的元组全部处理完毕。



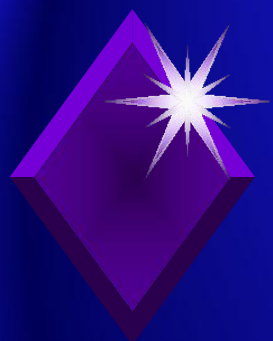
# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力







### 三、量化比较谓词 (见教材P93-95)

1、量化谓词用于产生单个列的子查询

2、量化谓词的通用形式

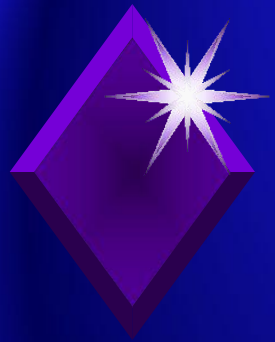
$\text{expr } \theta \{ \text{SOME} \mid \text{ANY} \mid \text{ALL} \} (\text{Subquery})$

其中  $\theta \in \{<, <=, =, <>, >, >=\}$

◆该形式中的SOME与ANY含义相同

◆量化谓词的定义：

- ❖ 对于 $\theta$ ，两个等价的谓词 $\text{expr } \theta \text{ SOME } (\text{Subquery})$ 和 $\text{expr } \theta \text{ ANY } (\text{Subquery})$ 为真，当且仅当至少存在一个由子查询返回的元素 $s$ ， $\text{expr } \theta s$ 为真；
- ❖ 对于 $\theta$ ， $\text{expr } \theta \text{ ALL } (\text{Subquery})$ 为真，当且仅当对每一个由子查询返回的元素 $s$ ， $\text{expr } \theta s$ 为真；



## 三、量化比较谓词

---

### 3、使用SOME的实例

**例7.17 (1)**      查询其它系中比信息系某一学生年龄小的学生姓名和年龄

```
SELECT SN, SA
FROM S
WHERE SA < SOME
      (SELECT SA
       FROM S
       WHERE SD = 'IS'
      )
AND SD <> 'IS';
```



### 三、量化比较谓词

---

#### 4、使用ALL实例

例7.17 (2) 查询其它系比信息系所有学生年龄小的学生的姓名和年龄。

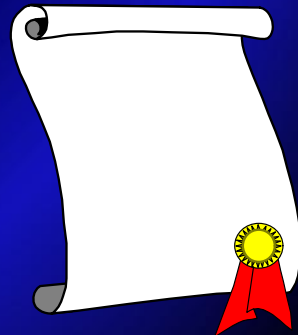
```
SELECT SN, SA
FROM S
WHERE SA < ALL
      (SELECT SA
       FROM S
       WHERE SD = 'IS'
      )
AND SD <> 'IS';
```



# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





## 四、EXISTS谓词 (见教材P95-97)

---

### 1、EXISTS谓词的通用形式

—— 测试被子查询检索到的行集(子查询可以返回多行多列)是否为空

**通用形式为：** [NOT] EXISTS (Subquery)

**谓词 EXISTS (Subquery) 为真当且仅当子查询返回一个非空的集合；**

**谓词 NOT EXISTS (Subquery) 为真当且仅当子查询返回的集合为空；**





## 四、EXISTS谓词

---

### 2、EXISTS谓词的使用

**前例7.14** 查询所有选修了 ‘C2’号课程的学生学号和姓名。（第3种方法）

```
SELECT S#,SN
FROM S
WHERE EXISTS
    (SELECT *
     FROM SC
     WHERE S# = S.S# AND C# = 'C2'
    ) ;
```

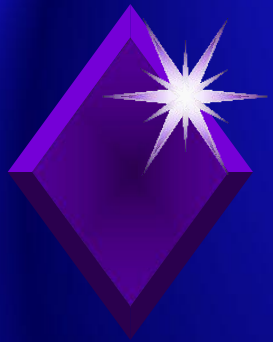


## 四、EXISTS谓词

### 3、用NOT EXISTS谓词实现关系代数的差运算

例7.18 查询没有选修 ‘C1’号课程的学生姓名。

```
SELECT SN
FROM S
WHERE NOT EXISTS
    (SELECT *
     FROM SC
     WHERE S# = S.S# AND C# = 'C1'
    ) ;
```



## 四、EXISTS谓词

---

### 4、EXISTS带来新的功能

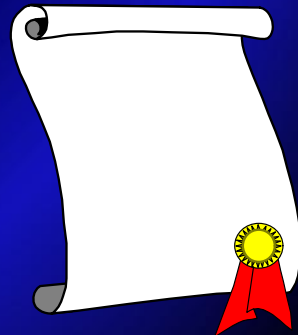
一些带EXISTS的子查询不能被其它形式的子查询等价替换，但**所有带IN、比较运算符、ANY和ALL谓词**的子查询都能用带EXISTS的子查询等价替换。



# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





## 五、SQL的过多等价形式

### 前例7.14 查询选修了课程“C2”的学生的姓名学号

#### 语句一：

```
SELECT S#, SN
FROM S
WHERE S.S# IN
      ( SELECT S#
        FROM SC
        WHERE C# = 'C2') ;
```

#### 语句二：

```
SELECT S#, SN
FROM S
WHERE 'C2' IN
      ( SELECT C#
        FROM SC
        WHERE S.S# = SC.S#
      ) ;
```

#### 语句三：

```
SELECT S#, SN
FROM S
WHERE EXISTS
      ( SELECT *
        FROM SC
        WHERE S.S#=SC.S#
          AND C# = 'C2') ;
```

#### 语句四：

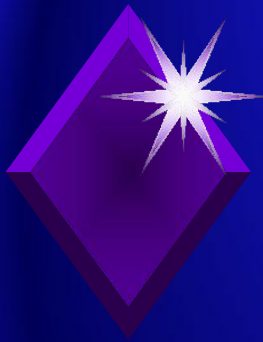
```
SELECT S#, SN
FROM S
WHERE S.S# = SOME
      ( SELECT S#
        FROM SC
        WHERE C# = 'C2') ;
```

#### 语句五：

```
SELECT S#, SN
FROM S, SC
WHERE S.S# = SC.S#
      AND C# = 'C2';
```

思考：还有什么方法？

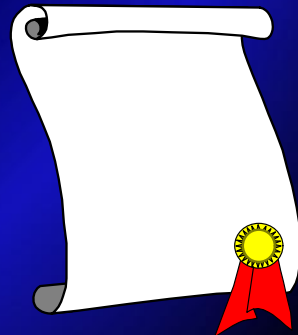




# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





## 六、FOR ALL 条件

---

(参考教材P93-97)

### 1、SQL语言没有全称量词

- 全称量词可以转换为等价的带有存在量词的谓词：

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

- 用NOT EXISTS实现全称量词的查询
- 用NOT EXISTS实现关系代数的除运算



## 六、FOR ALL 条件

### 2、用NOT EXISTS谓词实现全称量词的查询

例7.19 查询选修了全部课程的学生姓名。

语义转换：查询这样的学生 $x$ ，没有一门课程 $y$ 是 $x$ 不选修的

查询  
学生 $x$

```
SELECT  SN
FROM    S
WHERE NOT EXISTS
```

不存在  
课程 $y$

```
(SELECT  *
FROM    C
WHERE NOT EXISTS
```

$x$ 不选修  
课程 $y$

```
(SELECT  *
FROM    SC
WHERE   S# = S.S#
        AND C# = C.C#)
```

```
) ;
```



## 六、FOR ALL 条件

### 3、用NOT EXISTS谓词实现关系代数的除运算

**例7.20 查询至少选修了学生‘95002’选修的全部课程的学生号码**

**语义转换：**查询学号为x的学生，对所有的课程y，只要95002选修了课程y，则x也选修了y。

**形式化：** p ---- 学生95002选修了课程y

q ---- 学生x选修了课程y

则上述查询为：  $(\forall y)p \rightarrow q$

**谓词演算转换：**

$$\begin{aligned}(\forall y)p \rightarrow q &\equiv \neg(\exists y(\neg(p \rightarrow q))) \equiv \neg(\exists y(\neg(\neg P \vee q))) \\ &\equiv \neg(\exists y(P \wedge \neg q))\end{aligned}$$

**表达的语义：**查询学号为x的学生，不存在这样的课程y，学生95002选修了y，而学生x没有选。

## 六、FOR ALL 条件

查询  
学生x

```
SELECT  S#  
FROM    S  
WHERE NOT EXISTS
```

不存在  
课程y

```
( SELECT  *  
  FROM    C  
  WHERE EXISTS
```

95002选  
修了y

```
(SELECT  *  
  FROM    SC  SCX  
  WHERE SCX.C# = C.C#  AND SCX.S#= '95002' )  
AND NOT EXISTS
```

X没有  
选修y

```
(SELECT  *  
  FROM SC SCY  
  WHERE SCY.C# = C.C# AND SCY.S#=S.S#) );
```

## 六、FOR ALL 条件

另一个SELECT语句为：

查询  
学生x

```
SELECT DISTINCT S#  
FROM SC SCX  
WHERE NOT EXISTS
```

不存在  
课程y

```
( SELECT *  
FROM SC SCY  
WHERE SCY.S# = '95002'
```

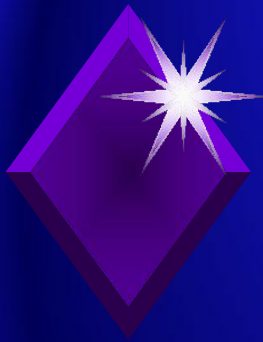
95002选  
修了y

```
AND NOT EXISTS
```

x没有  
选修y

```
(SELECT *  
FROM SC SCZ  
WHERE SCZ.S# = SCX.S#  
AND SCZ.C# = SCY.C#) ) ;
```

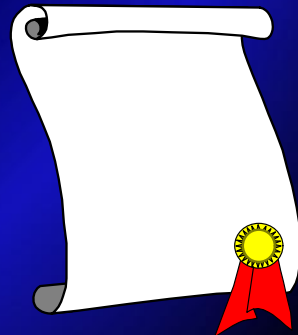


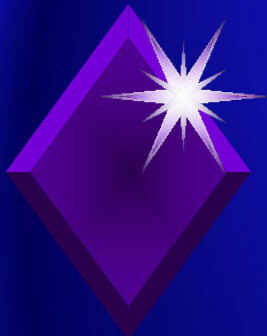


# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





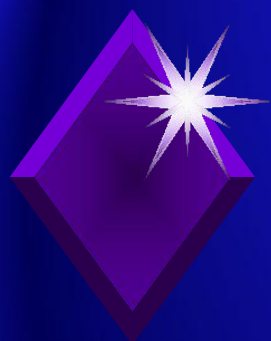
# 七、高级SQL语句

---

(参考教材P97-99)

## 1、高级SQL语法概述

- ①高级SQL语法不统一、不适用于所有的数据库系统
- ②大多数高级语法对某些关系查询提供了新的解法
- ③高级SQL提供“并”、“交”和“差”运算
- ④高级SQL对FROM子句进行了扩充



## 七、高级SQL语句

### 2、“并”、“交”“差”运算符

(1) UNION、INTERSECT和EXCEPT的高级SQL子查询形式

Subquery {UNION [ALL] | INTERSECT [ALL]  
| EXCEPT [ALL] Subquery }

- ◆ Q1 UNION Q2表示子查询Q1结果与Q2结果的并；
- ◆ Q1 INTERSECT Q2表示子查询Q1结果与Q2结果的交；
- ◆ Q1 EXCEPT Q2表示子查询Q1结果与Q2结果的差；

标准SQL没有提供集合“交”操作和集合“差”操作，可以用其它方法来实现。见P98-99

本节介绍MYSQL中的UNION、INTERSECT和EXCEPT



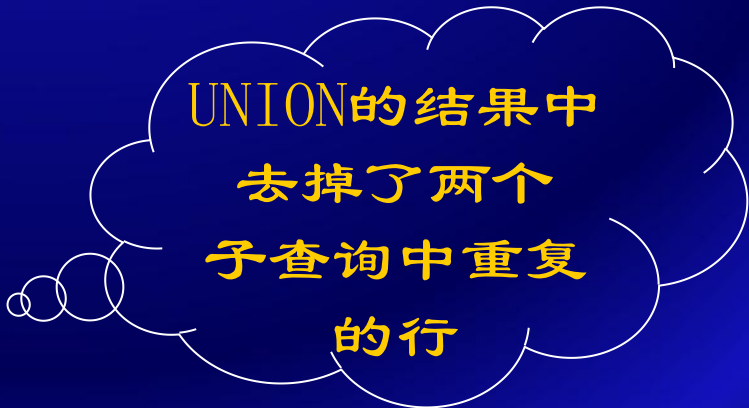
# 七、高级SQL语句

## (2) 使用UNION

◆ UNION运算实现关系代数的“并”运算

例7.21 查询计算机系的学生及年龄大于19岁的学生

```
SELECT *  
FROM S  
WHERE SD = 'CS'  
UNION  
SELECT *  
FROM S  
WHERE SA > 19;
```



UNION的结果中  
去掉了两个  
子查询中重复  
的行



## 七、高级SQL语句

### ◆ UNION ALL的使用

Q:= Q1 UNION [ALL] Q2

前例7.21 查询计算机系的学生及年龄大于19岁的学生

```
SELECT *  
FROM S  
WHERE SD = 'CS'
```

UNION ALL

```
SELECT *  
FROM S  
WHERE SA > 19;
```

对于两个子  
查询中的公  
共行，UNION  
结果将包含  
两个相同的行

UNION的结果中  
保留了两个  
子查询中重复  
的行





## 七、高级SQL语句

### (2) 使用INTERSECT

◆ INTERSECT 运算实现关系代数的“交”运算

**例7.22 查询既选修了课程“C1”又选修了课程“C2”的学生的学号与姓名**

```
SELECT S.S#, SN  
FROM S, SC  
WHERE S.S# = SC.S# AND C# = 'C1'
```

选修C1的学  
生集合

**INTERSECT**

```
SELECT S.S#, SN  
FROM S, SC  
WHERE S.S# = SC.S# AND C# = 'C2'
```

选修C2的学  
生集合





## 七、高级SQL语句

### ◆ INTERSECT ALL的使用

$Q := Q1 \text{ INTERSECT } [ALL] Q2$

假定X在Q1结果中出现M次，在Q2结果中出现N次

- INTERSECT不使用ALL时，

若M或N为0，则Q中X出现的次数= 0，否则=1

- INTERSECT使用ALL时，Q中X出现的次数=MIN (M, N) ；

假定Q1的查询结果为 {a, a, a, b, b, c, d} ,

Q2的查询结果为 {a, a, b, b, b, c, e}

则  $Q1 \text{ INTERSECT ALL } Q2$  的结果为 {a, a, b, b, c}

$Q1 \text{ INTERSECT } Q2$  的结果为 {a, b, c}



## 七、高级SQL语句

### (3) 使用EXCEPT

◆ EXCEPT 运算实现关系代数的“差”运算

**例7.23** 查询没选修“95001”所选修的任何课程的学生  
的学号。

SELECT S#

全体学生  
集合

FROM S

EXCEPT

SELECT SC1.S#

选修了95001所  
选修某课程的学生  
集合

FROM SC SC1, SC SC2

WHERE SC1.C# = SC2.C# AND SC2.S# = '95001';



## 七、高级SQL语句

---

### ◆ EXCEPT ALL的使用

$Q := Q1 \text{ EXCEPT } [ALL] Q2$

假定X在Q1结果中出现M次，在Q2结果中出现N次

- EXCEPT不使用ALL时，

若M不为0且N为0，则Q中X出现的次数 = 1，否则=0

- EXCEPT使用ALL时，Q中X出现的次数 = M-N，若M-N为负数，则看作0

假定Q1的查询结果为 {a, a, a, b, b, c, d},

Q2的查询结果为 {a, a, b, b, b, c, e}

则 Q1 EXCEPT ALL Q2 的结果为 {a, d}

Q1 EXCEPT Q2 的结果为 {d}



## 七、高级SQL语句

---

### 3、高级SQL中的FROM子句

#### (1) 基本SQL的FROM语法形式

**FROM** tableref {, tableref...}

tableref ::= tablename [[AS] corr\_name]



## 七、高级SQL语句

### (2) 高级SQL中的FROM子句语法形式

**FROM** tableref {, tableref...}

简单形式

子查询作为  
表使用

tableref::=

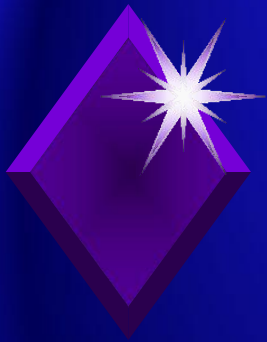
tablename [[AS] corr\_name [(colname {, colname...})]]

| (subquery) [AS] corr\_name [(colname {, colname...})]

| tableref1 [INNER | {LEFT | RIGHT | FULL}] [OUTER] JOIN tableref2

ON search\_condition | USING (colname {, colname...})

连接表



## 七、高级SQL语句

---

### (3) 简单形式

—— 用带括号的列名序列为从一个表 (FROM子句中的表) 中检索到的所有列重新命名

tablename [[AS]corr\_name[(colname{, colname...})]]

#### 例7.24 查询年龄大于19岁的学生学号和姓名

```
SELECT Student_number, Student_name
FROM    S AS S1  (Student_number, Student_name)
WHERE   SA > 19;
```





## 七、高级SQL语句

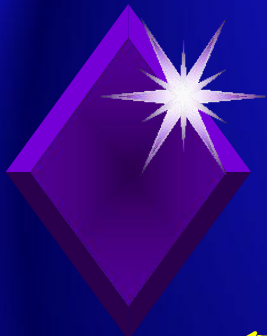
### (4) 子查询当作表使用

—— 将子查询放在FROM子句中,使得一个子查询或SELECT语句可以自由地检索另一个子查询的结果

(subquery) [AS] corr\_name [(colname {, colname...})]

#### 例7.25 查询至少有30人选修的课程的信息

```
SELECT C.C#, CN, CP#, CC
FROM (SELECT C#
      FROM SC
      GROUP BY C#
      HAVING COUNT(S#) >= 30) X, C
WHERE X.C# = C.C#;
```



## 七、高级SQL语句

---

### 例7.26 查询学生的最高成绩的平均值

```
SELECT AVG (t. x)
FROM
    (SELECT S#, MAX (G) AS x
     FROM SC
     GROUP BY S#) t;
```



## 七、高级SQL语句

---

前例7.14 查询选修了课程 ‘C2’ 的学生的姓名学号

语句六:

```
SELECT S#, SN
FROM S, (SELECT S#
        FROM SC
        WHERE C#= 'C2' ) AS P
WHERE S.S#=P.S#;
```



## 七、高级SQL语句

### (5) 特殊的连接结果当作表使用

tableref1[INNER|{LEFT|RIGHT|FULL}][OUTER]] JOIN tableref2 ON  
search\_condition|USING (colname{, colname...})

**功能**(连接概念参见教材P84-87)

INNER JOIN 即 JOIN 内连接

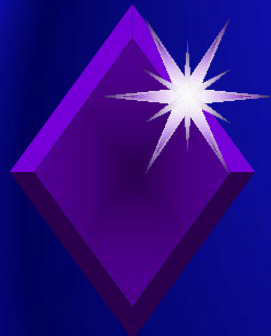
FULL OUTER JOIN 外连接

LEFT OUTER JOIN 左外连接

RIGHT OUTER JOIN 右外连接

● **用ON search\_condition形式**指定所要参加连接的列之间应该满足的条件

● **用USING子句**指定两表中参加连接的列名集合 (简化ON)



## 七、高级SQL语句

---

### ➤ JOIN 内连接

**例7.27 查询选修过课程的学生学号、姓名、课程号及成绩**

#### ❖ 用ON search\_condition

```
SELECT S.S#, SN, C#, G  
FROM S JOIN SC ON S.S# = SC.S#;
```

#### ❖ 用USING子句

```
SELECT S.S#, SN, C#, G  
FROM S JOIN SC USING (S#) ;
```



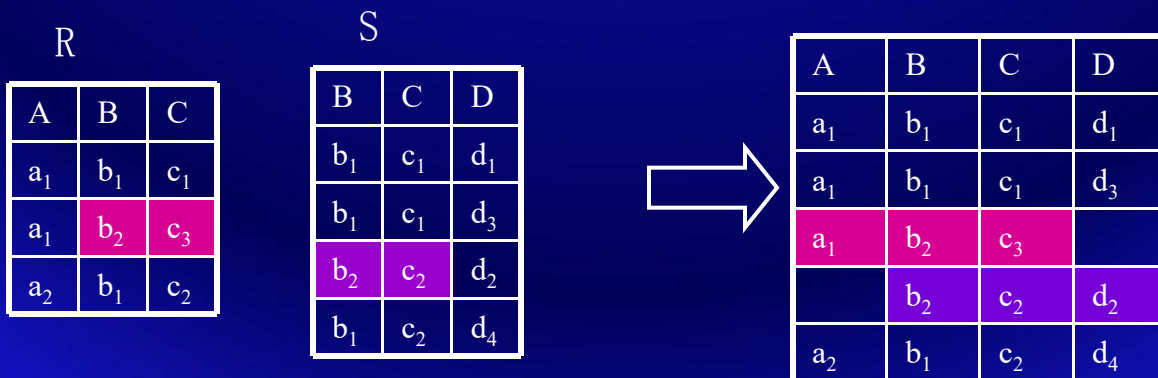
## 七、高级SQL语句

### ➤ FULL OUTER JOIN 外连接

实现关系代数  $R \bowtie_O S$

SELECT \*

FROM R FULL OUTER JOIN S USING (B, C) ;





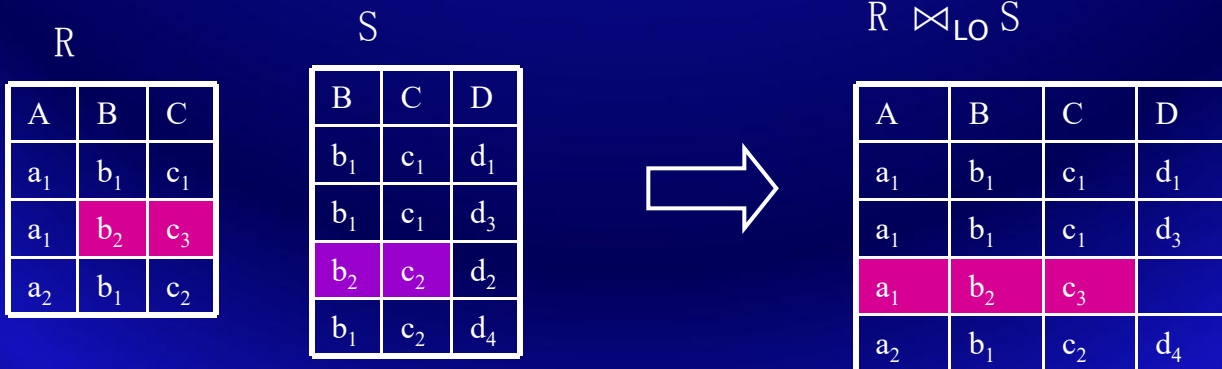
## 七、高级SQL语句

### ➤ LEFT OUTER JOIN 左外连接

实现关系代数  $R \bowtie_{LO} S$

SELECT \*

FROM R LEFT OUTER JOIN S USING (B, C) ;



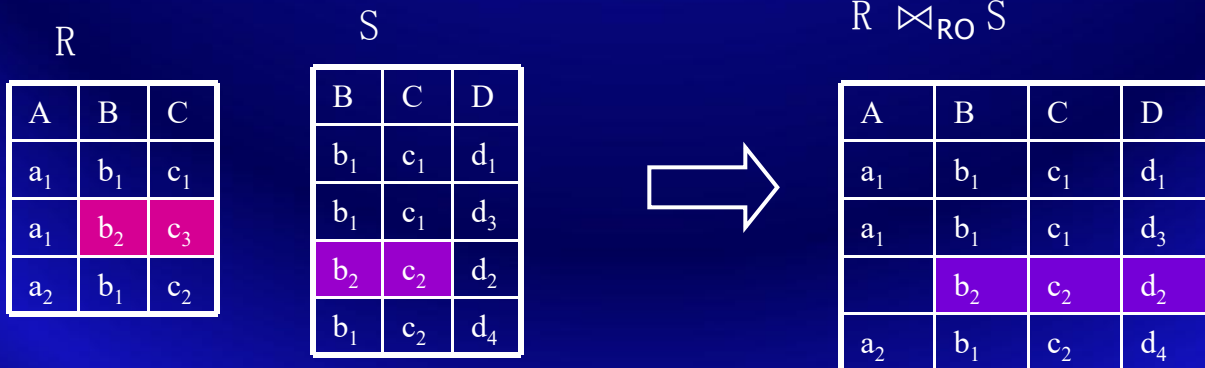
## 七、高级SQL语句

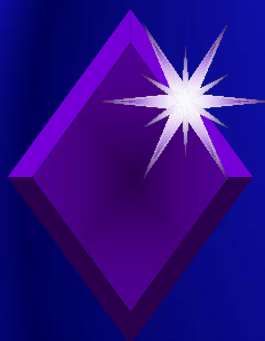
### ➤ RIGHT OUTER JOIN 右外连接

实现关系代数  $R \bowtie_{RO} S$

SELECT \*

FROM R RIGHT OUTER JOIN S USING (B, C) ;

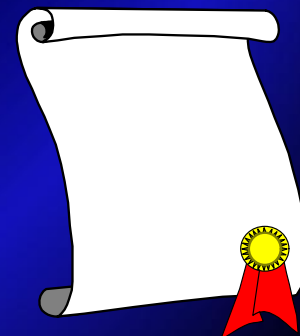




# 本讲主要内容

---

- 一. 连接查询
- 二. 子查询
- 三. 量化比较谓词
- 四. EXISTS谓词
- 五. SQL的过多等价形式
- 六. FOR ALL条件
- 七. 高级SQL语句
- 八. SELECT语句的表达能力





## 八、Select语句的能力

---

### 1、过程性语言与非过程性语言

➤ **过程性语言** (procedural language)

用该语言编写的程序应写明完成某项任务的**有序指令序列**

怎么做

➤ **非过程性语言** (non-procedural language)

用该语言编写的程序直接**描述**了所期望的结果。

做什么



## 八、Select语句的能力

---

- **非过程性语言编写程序 包含了两个方面：**
  - ❖ 必须指明要做什么，而不必说明怎么做
  - ❖ 程序的各语句之间不存在需要程序员来考虑的隐含顺序



# 八、Select语句的能力

---

## 2、非过程性语言便于处理即席查询

- **即席查询** (ad hoc query)  
—— 源于紧急的需求并且不可能用预先编好的程序来解决的查询
- **实际应用环境下，既存在例行查询，也存在即席查询**
- **非过程性语言既可以通过预先编制应用程序满足用户的例行查询要求，也便于用户临时处理即席查询要求**





## 八、Select语句的能力

### 3、SELECT语句的非过程性

例：求选修了‘C2’号课程的学生姓名。

用SQL表达：

```
SELECT S.SN
FROM S, SC
WHERE S.S# = SC.S# AND SC.C# = 'C2';
```

SELECT语句的非过程化程度比关系代数高

可以用多种等价的关系代数表达式表达：

Q1 := ((S  $\times$  SC) where S.S#=SC.S# and SC.C#='C2') [SN]

Q2 := (S  $\bowtie$  SC) where SC.C#='C2') [SN]

Q3 := (S  $\bowtie$  (SC where SC.C#='C2')) [SN]



## 八、Select语句的能力

---

### 4、图灵能力

- **图灵能力** (Turing power)  
—— 能执行有限长度的简单过程性程序的机器可以执行任何算法的计算过程
- **没有一种非过程性语言可能具备图灵能力**
- **可通过将数据库语言与过程性语言结合，来提供图灵能力** —— **嵌入式数据库语言**



## 八、Select语句的能力

---

### 5、基本SELECT语句的有限能力

➤ **缺少某些集合函数并不允许集合函数嵌套**

**例 求学生成绩表中成绩的中值 (median)**

**例 求每个学生总学分的平均学分**

```
SELECT AVG (SELECT SUM (CC)
              FROM SC, C
              WHERE SC.C# = C.C#
              GROUP BY S#) ;
```



基本SQL  
不允许子查询  
出现在集合  
函数的内部



## 八、Select语句的能力

---

### ➤ 基本SQL不能创建的报表

**例 求学生的平均成绩，并按照平均成绩低于60分、60-79、80-100分类统计学生人数**

- ❖ 不可能由基本SQL语句产生
- ❖ 可以用高级SQL功能实现：CAST表达式+FROM包含子查询
- ❖ 可以由过程性编程语言产生



## 八、Select语句的能力

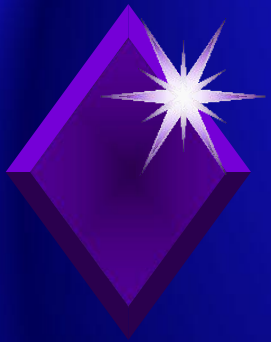
- 不能实现传递闭包 (transitive closure)

例 求课程“C3”的所有必须先修的课程

$C3 \rightarrow C1 \rightarrow C5 \rightarrow C7 \rightarrow C6$

C

课程号 C#	课程名 CN	先行课 CP#	学分 CC
C1	数据库	C5	4
C2	数学		2
C3	信息系统	C1	4
C4	操作系统	C6	3
C5	数据结构	C7	4
C6	数据处理		2
C7	PASCAL语言	C6	4



## 八、Select语句的能力

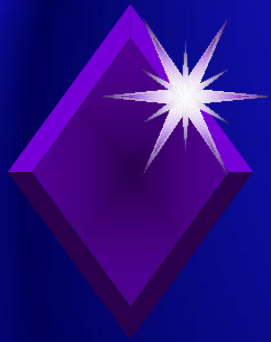
---

### ➤ 布尔条件的有限能力

Gerard Salton —— 文本检索领域的创始人之一列举大量的例子说明：布尔条件不能提供解决某些重大问题的能力

**例 查询平均成绩排名前20位的学生的学号**





*Questions?*





# 本讲主要目标

---



学完本讲后，你应该能够：

- 1、多个表放在FROM子句中，代表多表进行笛卡尔积。
- 2、出现在另一个SELECT语句之内的SELECT语句形式被称为子查询。两种子查询：相关子查询和不相关子查询
- 3、学会使用谓词IN、ANY、ALL和EXISTS；
- 4、学会使用UNION运算；
- 5、学会使用NOT EXISTS实现关系代数的除运算；
- 6、高级SQL语句的语法不统一，不适用于所有的数据库系统；
- 7、INTERSECT和EXCEPT是高级SQL提供的两个运算符，直接支持关系代数的“交”和“差”操作；
- 8、高级SQL的FROM子句中可以包含子查询或连接表；
- 9、SELECT语句的能力限制。



# 问题讨论

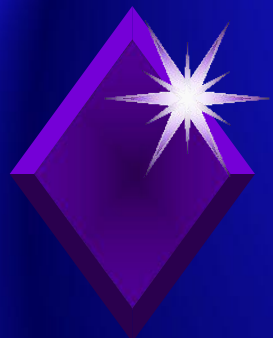
---

1、什么是相关子查询？什么是不相关子查询？举一个分别用相关子查询和不相关子查询实现的查询实例；并比较它们的概念性执行过程。你觉得哪一种子查询的实现效率更高？

2、一些带EXISTS的子查询不能被其它形式的子查询等价替换，但所有带IN、比较运算符、ANY和ALL谓词的子查询都能用带EXISTS的子查询等价替换。请你找两个不能替换的EXISTS子查询；也找几个用IN、比较运算符、ANY和ALL谓词的子查询，用EXISTS子查询实现。

3、你所使用的关系数据库管理系统的哪些SQL语句属于高级SQL语句？





# 练习

教材：《数据库系统原理教程》（第2版）

P112

1) 2

2) 4

