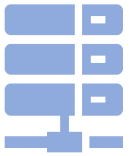
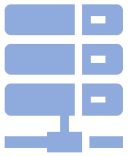
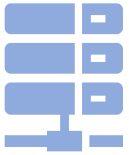


第三章 SQL语言



查询语言

- 主要内容：
  - 单表查询
  - 聚集和分组
  - 多表查询
  - 子查询
  - 集合查询



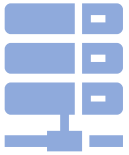
多表查询

- 连接查询
  - 在From子句中涉及多个表的查询
    - ✓ 一般会在Where子句中给出用来连接两个表的条件，称为连接条件或连接谓词，其一般格式：

      [<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>

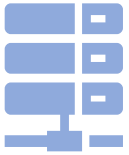
      比较运算符：=、>、<、>=、<=、!=

      [<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>
    - ✓ 连接谓词中的列名称为连接字段。连接条件中的各连接字段类型必须是可比的，但不必是相同的



多表查询

- 连接查询
  - SQL中连接查询的主要类型
    - ✓ 广义笛卡尔积
    - ✓ 等值连接(含自然连接)
    - ✓ 非等值连接查询
    - ✓ 自身连接查询
    - ✓ 外连接查询
    - ✓ 复合条件连接查询



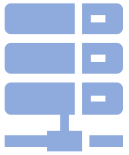
多表查询

- 连接查询
  - 广义笛卡尔积
    - ✓ 不带连接谓词的连接

```
SELECT  Student.* ,  SC.*
        FROM     Student, SC
```

    - ✓ 在SQL-92标准中广义笛卡尔积被称为交叉连接（CROSS JOIN），其一般格式是：

```
SELECT <属性或表达式列表>
        FROM <表名>CROSS JOIN <表名>
```



多表查询

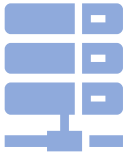
- 连接查询
  - 等值连接（包含自然连接）
    - ✓ Where子句中的连接运算符为=号的连接操作

[<表名1>.]<列名1> = [<表名2>.]<列名2>

任何子句中引用表1和表2中同名属性时，都必须加表名前缀。引用唯一属性名时可以加也可以省略表名前缀。

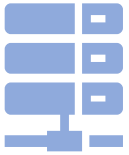
[例32] 查询每个学生及其选修课程的情况。

```
SELECT  Student.sno，sname, cno
FROM    Student，SC
WHERE  Student.Sno = SC.Sno;
```



多表查询

- 连接查询
  - 自然连接
    - ✓ 等值连接的一种特殊情况，把目标列中重复的属性列去掉
  - 非等值连接
    - ✓ 连接运算符不是=号的连接



多表查询

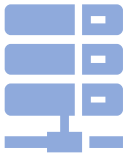
- 连接查询
  - 在SQL92标准中传统的连接操作被称为内连接（INNER JOIN），其一般格式是：

    SELECT <属性或表达式列表>

    FROM <表名> [INNER] JOIN <表名>

    ON <连接条件>

    [WHERE <限定条件>]
  - 上式中INNER可以省略，这里用ON短语指定连接条件，用WHERE短语指定其它限定条件。



多表查询

- 连接查询
  - 自身连接（自连接）
    - ✓ 一个表与其自己进行连接
    - ✓ 需要给表的两个副本起别名以示区别
    - ✓ 由于所有属性名都是同名属性，因此必须使用别名前缀

[例33] 查询每一门课的间接先修课（即先修课的先修课）

```
SELECT  FIRST.Cno，SECOND.Cpno
FROM  Course  AS FIRST，Course AS SECOND
WHERE FIRST.Cpno = SECOND.Cno;
```

多表查询

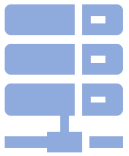
- 连接查询
  - 自身连接（自连接）

FIRST表（Course表）

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

SECOND表（Course表）

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4



多表查询

- 连接查询
  - 外连接（Outer Join）
    - ✓ 普通连接操作只输出满足连接条件的元组；外连接操作以指定表为连接主体，将主体表中不满足连接条件的元组一并输出

LEFT OUTER JOIN 或 LEFT JOIN

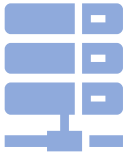
RIGHT OUTER JOIN 或 RIGHT JOIN

FULL OUTER JOIN 或 FULL JOIN

```
SELECT <属性或表达式列表>
    FROM <表名> [LEFT | RIGHT | FULL] [OUTER] JOIN <表名>
    ON <连接条件>
    [WHERE <限定条件>]
```

Mysql不支持全外连接

解决方法：Left join Union Right join



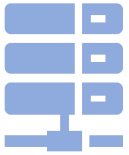
多表查询

- 连接查询
  - 外连接（Outer Join）

[例34] 查询每个学生及其选修课程的情况包括没有选修课程的学生----用外连接操作

```
SELECT  Student.Sno，Sname，Ssex，
                    Sage，Sdept，Cno，Grade
From  Student Right Outer Join SC
   on Student.Sno =SC.Sno;
```

(不能写成：Where Student.Sno=Sc.Sno)



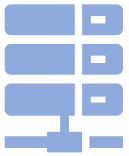
多表查询

- 连接查询
  - 复合条件连接

    WHERE子句中含多个连接条件时，称为复合条件连接

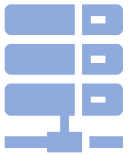
[例34] 查询选修2号课程且成绩在90分以上的所有学生的学号、姓名

```
SELECT Student.Sno, student.Sname
FROM    Student, SC
WHERE Student.Sno = SC.Sno AND
                                    /* 连接谓词*/
        SC.Cno= ' 2   ' Or   /* 其他限定条件 */
    Not SC.Grade > 90;        /* 其他限定条件 */
```



查询语言

- 主要内容：
  - 单表查询
  - 聚集和分组
  - 多表查询
  - 子查询
  - 集合查询

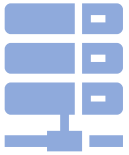


子查询

- 子查询
  - 一个SELECT-FROM-WHERE语句称为一个查询块
  - 通常将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中（称为嵌套查询）

```
SELECT Sname                    外层查询/父查询
FROM Student
WHERE Sno IN
    (SELECT Sno                 内层查询/子查询
     FROM SC
     WHERE Cno= ' 2 ') ;
```

  - ✓ 层层嵌套方式反映了 SQL语言的结构化
  - ✓ 子查询中不能使用ORDER BY子句 (Why?)

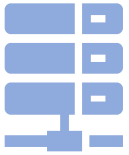


子查询

- 子查询分类：
  - 子查询可以插入到From子句中，做为临时表使用
    - 临时表需起个别名来引用
    - 不是所有数据库产品都支持临时表

```
SELECT   IS.Sno, Sname, Cno
        FROM    SC, (SELECT Sno, Sname from Student
WHERE  Sdept=  ‘ IS  ‘) as SIS
        WHERE SC.Sno=SIS.Sno
```

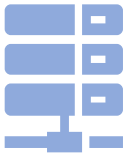
  - 子查询甚至可以插入到Select子句中
    - 确保返回结果为单列单行
    - 多数数据库不支持



子查询

- 子查询分类：
  - 不相关子查询
    - 由里向外逐层处理。即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件
  - 相关子查询

    子查询的查询条件依赖于父查询
    - 首先取外层查询中表的第一个元组，根据它与内层查询相关的属性值处理内层查询，若WHERE子句返回值为真，则取此元组放入结果表；
    - 然后再取外层表的下一个元组；
    - 重复这一过程，直至外层表全部检查完为止。



子查询

- 引出子查询的谓词
  - 带有IN谓词的子查询
  - 带有比较运算符的子查询
  - 带有ANY或ALL谓词的子查询
  - 带有EXISTS谓词的子查询

子查询

- 带有IN谓词的子查询

[例35] 查询与“刘晨”在同一个系学习的学生。

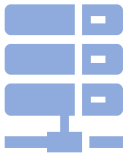
此查询可以分步来完成

① 确定“刘晨”所在系名（是‘IS’系）

```
SELECT  Sdept  FROM    Student
 WHERE  Sname= ' 刘晨   ‘;
```

② 查找所有在IS系学习的学生。

```
SELECT   Sno，Sname，Sdept
FROM     Student
WHERE  Sdept= ' IS ';
```



子查询

■带有IN谓词的子查询

[例36] 查询与“刘晨”在同一个系学习的学生。

也可以直接构造嵌套查询：

将第一步查询嵌入到第二步查询的条件中

```
SELECT Sno，Sname，Sdept FROM Student
WHERE Sdept  IN
      (SELECT Sdept FROM Student
       WHERE Sname=  ‘刘晨’ );
```

此查询为不相关子查询

子查询

■带有IN谓词的子查询

父查询和子查询中的表均可以定义别名，简化书写

```
SELECT Sno，Sname，Sdept
 FROM   Student  S1
 WHERE  S1.Sdept  IN
       (SELECT Sdept
        FROM    Student  S2
        WHERE S2.Sname=  ‘刘晨’ );
```

子查询

■带有IN谓词的子查询

复杂一点的例子

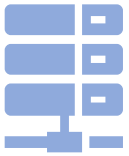
[例37]. 查询选修了课程名为“信息系统”的学生学号和姓名

```
SELECT Sno，Sname
FROM    Student
WHERE Sno  IN
        (SELECT Sno
         FROM    SC
         WHERE  Cno IN
            (SELECT Cno
            FROM Course
            WHERE Cname=   ‘信息系统’ ));
```

③ 最后在Student关系中取出Sno和Sname

② 然后在SC关系中找出选修了3号课程的学生学号

① 首先在Course关系中找出“信息系统”的课程号，结果为3号



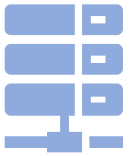
子查询

■带有IN谓词的子查询

与上例等价的连接查询

[例37]. 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   Student，SC，Course
        WHERE Student.Sno = SC.Sno  AND
              SC.Cno = Course.Cno AND
              Course.Cname= ‘信息系统’ ；
```

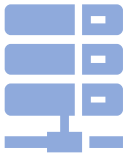


子查询

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

假设一个学生只可能在一个系学习，并且必须属于一个系，则在[例36]中可以用 = 代替IN

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

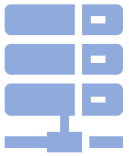


子查询

■子查询一定要跟在比较符之后

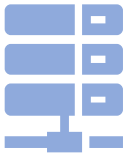
错误的例子：

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



子查询

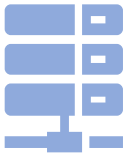
- 带有ANY或ALL谓词的子查询
  - 谓词语义
    - ✓ ANY：任意一个值
    - ✓ ALL：所有值
  - 与比较运算符配合使用



子查询

■带有ANY或ALL谓词的子查询

> 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



子查询

■带有ANY或ALL谓词的子查询

[例38] 查询其他系中比信息系任意一个(其中某一个)学生年龄小的学生姓名和年龄

```
SELECT Sname，Sage
FROM    Student
WHERE Sage < ANY (SELECT  Sage
                  FROM    Student
                  WHERE Sdept= ' IS ')
      AND Sdept <> ' IS ' ;
          /* 注意这是父查询块中的条件 */
```

子查询

■带有ANY或ALL谓词的子查询

- 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

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

ANY与ALL与集函数的对应关系

子查询

■带有ANY或ALL谓词的子查询

[例39] 用集函数实现 [例38] 的查询

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

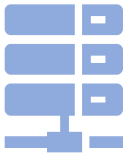
子查询

■带有ANY或ALL谓词的子查询

[例40] 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 Student
         WHERE Sdept= ' IS ')
      AND Sdept <> ' IS ’ ;
```



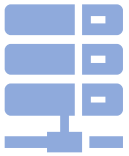
子查询

■带有ANY或ALL谓词的子查询

[例40] 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。

方法二：用集函数

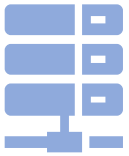
```
SELECT Sname，Sage
FROM Student
WHERE Sage <
        (SELECT MIN(Sage)
         FROM Student
         WHERE Sdept= ' IS ')
    AND Sdept <>' IS ’ ;
```



子查询

■带有Exists谓词的子查询

- 1. EXISTS谓词
  - 存在量词 ∃
  - 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



子查询

- 带有Exists谓词的子查询

A:

a	b
1	2
2	3

B:

a	c
1	2
3	3

Select * from A where exists

(select * from B where B.a=A.a)

1:逐行考察A中元组

A中第一行元组, a=1

2:考察EXISTS子句里的查询，看其是否选中了元组

该子句此时相当于:

Select * from B where a=1

选中第一行元组

因此EXISTS子句返回TRUE

子查询

■带有Exists谓词的子查询

•**A**

a	b
1	2
2	3

•**B**

a	c
1	2
3	3

•Select * from A where exists

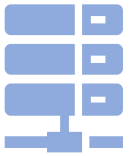
(select * from B where B.a=A.a)

•3:当Exists子句返回TRUE时，最外层查询所考察的元组被选中

•因此A表第一行被选中了

•同理考察第二行，不被选中。因此最终结果是:

a	b
1	2

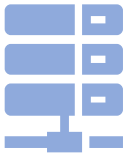


子查询

■带有Exists谓词的子查询

[例41] 查询所有选修了1号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE EXISTS
  (SELECT *
   FROM SC          /*相关子查询*/
   WHERE Sno=Student.Sno AND
Cno= ' 1 ');
```



子查询

■带有Exists谓词的子查询

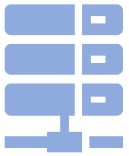
[例41] 查询所有选修了1号课程的学生姓名。

思路分析：

本查询涉及Student和SC关系。

在Student中依次取每个元组的Sno值，用此值去检查SC关系。

若SC中存在这样的元组，其Sno值等于此Student.Sno值，并且其Cno= '1'，则取此Student.Sname送入结果关系。



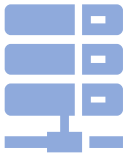
子查询

■带有Exists谓词的子查询

- Exists谓词也可以用于不相关子查询

select * from a where exists (select * from b)

- 一些带EXISTS或NOT EXISTS谓词的子查询不能被其他形式的子查询等价替换
- 所有带IN谓词、比较运算符、ANY和ALL谓词的子查询都能用带EXISTS谓词的子查询等价替换。

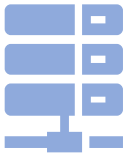


子查询

■带有Exists谓词的子查询

[例36] 中的IN查询可以用带EXISTS谓词的子查询替换：

```
SELECT Sno，Sname，Sdept
FROM Student S1
WHERE EXISTS
    SELECT *
    FROM Student S2
    WHERE S2.Sdept = S1.Sdept AND
        S2.Sname = ' 刘晨 ';
```



子查询

■用EXISTS/NOT EXISTS实现全称量词

SQL语言中没有全称量词∀ （For all）

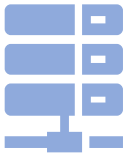
可以把带有全称量词的谓词转换为等价的带有存在量词的谓词：

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

只能用EXISTS谓词实现该查询

[例42] 查询选修了全部课程的学生姓名。

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



子查询

■用EXISTS/NOT EXISTS实现逻辑蕴函

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：

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

只能用EXISTS谓词实现该查询

[例43] 查询至少选修了学生95002选修的全部课程的学生号码

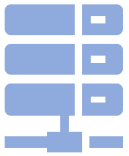
●用逻辑蕴函表达：查询学号为x的学生，对所有的课程y，只要95002学生选修了课程y，则x也选修了y。

●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: $(\forall y)\ p \rightarrow q$



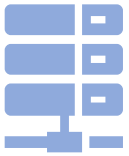
子查询

- 用EXISTS/NOT EXISTS实现逻辑蕴函

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

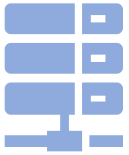
变换后语义：不存在这样的课程y，学生95002选修了y，而学生x没有选。

```
SELECT DISTINCT Sno FROM SC SCX
        WHERE NOT EXISTS
            (SELECT * FROM SC SCY
             WHERE SCY.Sno = ' 95002 '  AND
                NOT EXISTS
                    (SELECT * FROM SC SCZ
                     WHERE SCZ.Sno=SCX.Sno AND
                                SCZ.Cno=SCY.Cno));
```



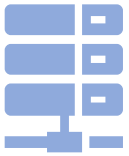
查询语言

- 主要内容：
  - 单表查询
  - 聚集和分组
  - 多表查询
  - 子查询
  - 集合查询



查询语言

- 集合查询
  - 标准SQL直接支持的集合操作种类
    - ✓ 并操作 (subquery) UNION (subquery)
  - 一般商用数据库支持的集合操作种类
    - ✓ 并操作 (subquery) UNION (subquery)
    - ✓ 交操作 (subquery) INTERSECT (subquery)
    - ✓ 差操作 (subquery) EXCEPT (subquery)

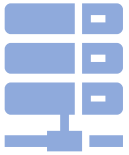


查询语言

■ 集合查询

[例44] 查询计算机科学系的学生及年龄不大于19岁的学生。

```
SELECT *
        FROM Student
        WHERE Sdept= 'CS'
 UNION
SELECT *
        FROM Student
        WHERE Sage<=19;
```



查询语言

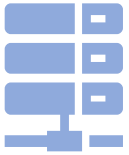
■ 集合查询

[例45] 查询计算机科学系选修95001号课程的学生的学号。

```
SELECT Sno
    FROM Student
    WHERE Sdept= 'CS'
Intersect
SELECT Sno
    FROM SC
    WHERE Cno= ‘95001’ ;
```

MySQL不支持Intersect

用什么办法等效实现？



查询语言

■ 集合查询

[例46] 查询计算机科学系没有选修95001号课程的学生的学号。

```
SELECT Sno
    FROM Student
    WHERE Sdept= 'CS'
 Except
SELECT Sno
    FROM SC
    WHERE Cno= ‘95001’ ;
```

MySQL不支持Except

用什么办法等效实现？