第三章 关系数据库标准语言SQL

3.4 数据查询

回顾

SELECT 语句的完整句法：

```
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1> [ HAVING <条件表达式> ] ]
  [ ORDER BY <列名2> [ ASC|DESC ] ];
```

连接（一般格式）：

- [<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>
- [<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>

等值连接 / 自然连接；自身连接、嵌套连接

3.4.3 嵌套查询

- 在WHERE子句中包含一个形如SELECT-FROM-WHERE的查询块，此查询块称为子查询或嵌套查询，包含子查询的语句称为父查询或外部查询。
- 嵌套查询可以将一系列简单查询以层层嵌套的方式构成复杂查询，增强了查询能力，充分体现了SQL“结构化”的特点。
- 嵌套查询在执行时由内向外处理，每个子查询在上一级外部查询处理之前完成，父查询要用到子查询的结果。
- 有些嵌套查询可以用连接运算替代。
- 子查询的限制：不能使用Order by子句。

嵌套查询的应用场景

- 一个查询块是对关系集合进行搜索找出满足资格的元组。对目标关系中成员t的判断可能会出现情况：

  （1）该成员t是否属于某个select结果集合R；

  t∈R 是否成立？（属于运算）

  （2）该成员是否比某个select集中所有成员或至少一个成员大或小；

  t ＞ R中所有或某个成员是否成立？ （比较运算）

  （3）该成员是否能使得集合逻辑式成立；

  t是否能使得逻辑命题L成立？ 其中：L是一个通过t求出的集合逻辑式。

  L（R(t)） （逻辑运算）

3.4.3 嵌套查询

1. 当子查询的返回值只有一个时，可以使用比较运算符（=，>，<，>=，<=，!=）将父查询和子查询连接起来。

[例] 查询与学号为95003的学生同系的学生学号和姓名。

```
SELECT sno, sname   //父查询
FROM STUDENT
WHERE sdept = (SELECT sdept
FROM STUDENT
 WHERE sno= '95003' );
```

本例改为：

```
SELECT sno, sname
   FROM STUDENT
    WHERE
     ( SELECT sdept
        FROM STUDENT
      WHERE sno= '95003' );
```

对吗？

3.4.3 嵌套查询

当子查询的返回结果不止一个，而是一个集
时可以使用IN、ANY、ALL或EXISTS谓词

2. 带有IN谓词的子查询

[例] 查询与“李冬”同系的学生学号和姓名（可能有同名）。

```
SELECT Sno, Sname
FROM STUDENT
WHERE sdept IN
 (SELECT Sdept
  FROM STUDENT
  WHERE sname= ‘李冬’ );
```

子查询的查询条件不依赖于父查询——不相关子查询

本例用自身连接：

```
SELECT  a.Sno，a.Sname，a.Sdept
FROM  Student a，Student b
WHERE a.Sdept = b.Sdept  AND
b.Sname = ‘李冬';
```

3.4.3 嵌套查询

[例] 找出每个学生超过他选修课程平均成绩的课程号。

```
SELECT Sno，Cno
  FROM  SC  x
  WHERE Grade >=(SELECT AVG(Grade)
                 FROM  SC y
                  WHERE y.Sno=x.Sno);
```

子查询的查询条件与父查询相关——相关子查询

- 可能的执行过程：

1. 从外层查询中取出SC的一个元组x，将元组x的Sno值传送给内层查询。
2. 执行内层查询，得到值88（近似值），用该值代替内层查询，得到外层查询。
3. 执行父查询，得到x对应的满足条件的集合；
4. 外层查询取出下一个元组重复做上述1至3步骤，直到外层的SC元组全部处理完毕。

带有比较运算符的子查询问题分析

- 相关嵌套查询效率低，如何提高？ 将相关查询改为无关查询。

[例] 对于上例，先从选课表中找出每个学生选课平均成绩，
再从选课表中找出所选课程成绩大于平均成绩的学生

```
SELECT sno, AVG(grade) avg_grade
FROM  SC
as Y;
SELECT sno, cno
FROM  SC, Y
WHERE SC.sno=Y.sno AND SC.grade>Y.avg_grade;
```

子查询执行方式

子查询分为非相关子查询和相关子查询。二者执行方式不同：

- 非相关子查询的执行顺序是：
  - 首先执行子查询；
  - 父查询所涉及的所有元组都与子查询的查询结果进行比较，以确定查询结果集合。
- 相关子查询的执行顺序是：
  - 首先选取父查询表中的一个元组，内部的子查询利用此元组中相关的属性值进行查询；
  - 然后父查询根据子查询返回的结果判断此行是否满足查询条件。如果满足条件，则把该行放入父查询的查询结果集合中。
  - 重复执行上述过程，直到处理完父查询表中的所有元组。

由此可以看出，非相关子查询只执行一次；而相关子查询的执行次数是由父查询表的行数决定的。

3.4.3 嵌套查询

- **课堂练习：查询平均成绩高于“王军”同学平均成绩的学生姓名和学号；**

```
select sno,sname
from student s join sc on sc.sno=s.sno
group by s.sno,sname
having avg(grade)>(select avg(grade)
                   from sc,student
                   where sc.sno=student.sno
                     and sname='王军');
```

```
select sno,sname from student
where sno in (select sno from sc
      group by sno having avg(grade)>(
         select avg(grade) from sc
         where sno=(select sno from student
                    where sname='王军'))
      )
```

```
select sno,sname from student
where (select avg(grade) from sc as
sc1 where sc1.sno=student.sno)
   >
  (select avg(grade) from sc as
sc2,student s2 where
sc2.sno=s2.sno and sname='王军')
```

3.4.3 嵌套查询

问题：在嵌套查询情形二中，若需要判断关系中元组t与一个集合的“任意一个”或“所有”是否满足某个关系式，该如何解决?

3. 带有ANY（SOME）或ALL谓词的子查询

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

IN 等价于 = SOME
NOT IN 等价于 < > ALL

3.4.3 嵌套查询

[例]：查询其他系中比信息系某一学生的年龄小的学生姓名及年龄。

```
SELECT sname, sage
 FROM STUDENT
WHERE  sage < ANY
        (SELECT sage
             FROM STUDENT
             WHERE sdept =  'IS' )
AND sdept <>  'IS' ;
```

ANY或ALL谓词可用集函数或IN谓词等价表示：

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

执行过程：

1.RDBMS执行此查询时，首先处理子查询，找出IS系中所有学生的年龄，构成一个集合;

2. 处理父查询，找所有不是IS系且年龄小于上述集合中某个值的学生。

带有ANY或ALL谓词的子查询

表3.5 ANY、ALL谓词与聚集函数、IN谓词的等价转换关系

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

3.4.3 嵌套查询

4.带有EXISTS谓词的子查询

- EXISTS表示存在量词。带有EXISTS的子查询不返回任何实际数据，它只得到逻辑值“真”或“假”。
- 当子查询的的查询结果集合为非空时，外层的WHERE子句返回真值，否则返回假值。NOT EXISTS与此相反。
- 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。

[例] 查询选修了1号课程的学生姓名。

```
SELECT sname FROM STUDENT
WHERE EXISTS
   (SELECT * FROM  SC
   WHERE SC.Sno=Student.Sno AND Cno= ‘1’ );
```

等价于：

```
SELECT Sname
    FROM Student, SC
WHERE Student.Sno=SC.Sno
        AND SC.Cno= '1';
```

带有EXISTS谓词的子查询示例

[例] 查询与“刘晨”在同一个系学习的学生。

1）用EXISTS函数判断任一个学生t，其院系与刘晨院系是否相同

2）从学生表中，搜索让上述逻辑式为真的元组

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

注：S1是所有学生元组变量，S2是所有院系元组变量。

3.4.3 嵌套查询

- 不同形式的查询间的替换：
  - 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
- 用EXISTS/NOT EXISTS实现全称量词(难点)
  - 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：

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

3.4.3 嵌套查询

[例] 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))
```

所有课程，所求学生选之

⇔

不存在任何一门课程，所求学生没有选之

在student中求满足下列条件的sname:

在course中不存在这样的课程，SC中没有该学生(sno)的该课程(cno)的成绩记录。

3.4.3 嵌套查询

[例]查询至少选修了学生200215122选修的全部课程的学生号码。

解题思路

- 用逻辑
  200215
- 形式化
  用p表示
  用q表示
  则上述查
- 等价变
  (∀
- 变换后
  有选。

■用NOT EXISTS谓词表示：

```
SELECT DISTINCT Sno
  FROM SC SCX
  WHERE NOT EXISTS
        (SELECT *
         FROM SC SCY
         WHERE SCY.Sno = ' 200215122 '  AND
               NOT EXISTS
               (SELECT *
                FROM SC SCZ
                WHERE SCZ.Sno=SCX.Sno AND
                      SCZ.Cno=SCY.Cno));
```

Ex没

3.4.4 集合查询

- 集合操作的种类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT
- 参加集合操作的各查询结果的列数必须相同；对应项的数据类型也必须相同

3.4.4 集合查询

[例] 查询计算机科学系的学生及年龄不大于19岁的学生。

方法一：

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

方法二：

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

- UNION：将多个查询结果合并起来时，系统自动去掉重复元组。
- UNION ALL：将多个查询结果合并起来时，保留重复元组

3.4.4 集合查询

[例] 查询选修课程1的学生集合与选修课程2的学生集合的交集

方法一：

```
SELECT Sno
  FROM SC
  WHERE Cno=' 1 '
  INTERSECT
  SELECT Sno
  FROM SC
  WHERE Cno='2 '
```

方法二：

```
SELECT Sno FROM SC
    WHERE Cno=' 1 ' AND Sno IN
        (SELECT Sno
          FROM SC
          WHERE Cno=' 2 ');
```

3.4.4 集合查询

[例] 查询计算机科学系的学生与年龄不大于19岁的学生的差集。

方法一：

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

方法二：

```
SELECT *
   FROM Student
   WHERE Sdept= 'CS' AND
Sage>19;
```

3.4.5 基于派生表的查询

- 当子查询出现在FROM子句中，这时子查询生成临时派生表（derived table）成为主查询的查询对象。

- [例] 找出每个学生超过他选修课程平均成绩的课程号

```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade) FROM SC
          Group by Sno)
         AS Avg_sc(avg_sno, avg_grade)
Where SC.sno = avg_sno
      AND grade > avg_grade;
```

必须为派生关系指定别名

SELECT综合实例

假设银行数据库关系模式为：

Branch=(Bname, Bcity, Bassets)

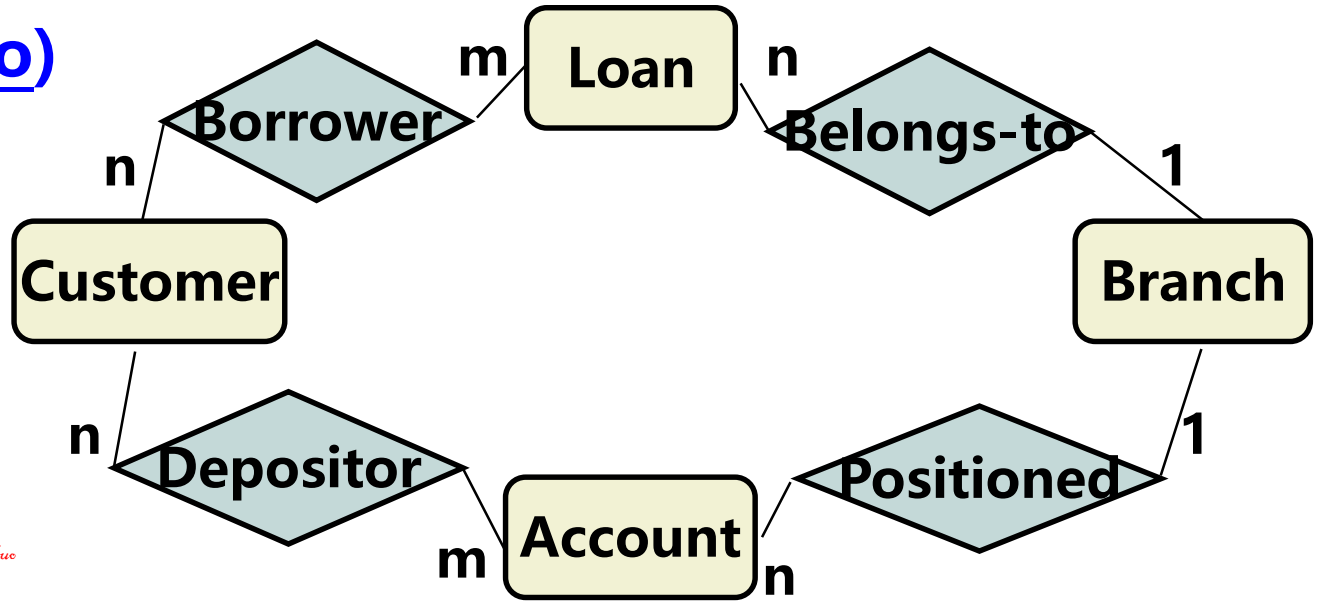
Customer=(Cno, Cname, Cstreet, Ccity)

Loan=(Lno, Bname, amount)

Borrower=(Cno, Lno)

Account=(Ano, Bname, balance)

Depositor=(Cno, Ano)



SELECT综合实例

银行数据库关系模式为：

Branch=(Bname, Bcity, Bassets)
Customer=(Cno, Cname, Cstreet, Ccity)
Loan=(Lno, Bname, amount)
Borrower=(Cno, Lno)
Account=(Ano, Bname, balance)
Depositor=(Cno, Ano)

- 思考题：
  1） 找出在“Perry”银行有贷款的客户姓名及贷款数；
  2） 找出资产至少比位于Brooklyn的某一家支行高的支行名；
  3）找出银行中在Perry银行既有贷款又有账户的客户姓名；
  4）找出平均余额最高的支行；
  5）找出住在Harrison且在银行中至少有三个账户的客户的平均余额；
  6）找出在Brooklyn的所有支行都有账户的客户；

SELECT综合实例

银行数据库关系模式为：
Branch=(Bname, Bcity, Bassets)
Customer=(Cno, Cname, Cstreet, Ccity)
Loan=(Lno, Bname, amount)
Borrower=(Cno, Lno)
Account=(Ano, Bname, balance)
Depositor=(Cno, Ano)

- 思考题解答：

1、找出在“Perry”银行有贷款的客户姓名及贷款数；

解法一：整体法 关键词（客户、有贷款、贷款）

1）找from customer ⋈ borrower ⋈ loan

2）用where过滤 Bname="perry"

3）用select投影 Cname,amount

```
SELECT  Cname,amount
    FROM   customer C, borrower B, loan L
    WHERE  C.Cno=B.Cno and B.Lno = L.Lno and
Bname="perry"
```

SELECT综合实例

- 1、找出在“Perry”银行有贷款的客户姓名及贷款数；
- 解法二：分步法

1）找出在perry有贷款的所有Cno集合

用一个select块从borrower ⋈ loan 中查询；

2）从customer中选择其Cno IN 1）结果集的元组

```
select  Cname,amount
    from   customer C
    where  C.Cno  IN  (select  Cno
                       from    borrower B, loan L
                       where  B.Lno = L.Lno
                          and Bname="perry");
```

SELECT综合实例

- 解法三：相关法

1）用exists函数构造一个能够判断任意一个顾客元组t，t.Cno是否在perry银行有贷款集合中的逻辑函数；

2）对customer中每个元组，调用exists判断

```
select  Cname,amount
      from   customer C
      where  EXISTS  (select  *
                      from   borrower B, loan L
                      where  C.Cno = B.Cno
                             and B.Lno = L.Lno
                             and Bname="perry");
```

SELECT综合实例

- 思考题解答：

2、找出资产至少比位于Brooklyn的某一家支行多的支行名；

解法一：整体法 关键词（一家支行、位于Brooklyn的支行）

1）找from branch(角色T) × branch(角色S)

2）用where过滤 T.assets>S.assets
and S.city="Brooklyn"

3）用select投影 T.Bname

```
select  T.Bname
   from   branch T, branch S
   where  T.assets > S.assets
            and S.city =  'Brooklyn' ;
```

SELECT综合实例

2、找出资产至少比位于Brooklyn的某一家支行多的支行名；

解法二：分步法

1）找出位于Brooklyn的支行的总资产集合，用一个select块从branch中查询；

2）对上述结果用集合函数any()求出集合的任意一个

3）从branch中选择其总资产大于any()函数返回值的元组

```
select  Bname
from   branch
where  assets > any(select  assets
                    from   branch
                    where  city='Brooklyn');
```

SELECT综合实例

- 思考题解答：

3、找出银行中在Perry银行既有贷款又有账户的客户姓名；

解法一：使用交运算

1）找出在Perry银行有贷款的客户

2）找出在Perry银行有账户的客户

3）用intersect求交集

```
(select  distinct Cno
 from   borrower B, loan L
 where  B.Lno=L.Lno and Bname="Perry")
  intersect
(select  distinct Cno
 from   depositor D, account A
 where  D.Ano=A.Ano and Bname="Perry");
```

SELECT综合实例

3、找出银行中在Perry银行既有贷款又有账户的客户姓名；

- **解法二：** 1）找出在Perry银行有贷款的客户（集合S1）
  2）找出在Perry银行有账户的客户（集合S2）
  3）对于集合S1中每个元组，判断是否属于S2

```
select  distinct Cno
from    borrower, loan
where   B.Lno=L.Lno and Bname="Perry"
        and Cno IN (select  distinct Cno
                    from    depositor, account
                    where   D.Ano=A.Ano
                            and Bname="Perry");
```

或 (Bname,Cno) IN (select distinct Bname,Cno
from depositor, account
where D.Ano=A.Ano);

SELECT综合实例

- 3、找出银行中在Perry银行既有贷款又有账户的客户姓名；
- 解法三：相关法

1）用exists函数构造一个能够判断任意一个顾客元组t，t.Cno是否在perry银行有账户的逻辑函数；

2）对borrower,loan中每个在perry银行有贷款元组，调用exists判断该元组是否有账户

```
select  Cname
      from   borrower B, loan L
      where  L.Bname=’ Perri' and B.Lno=L.Lno and
             EXISTS  (select  *
                      from   depositor D, account A
                      where  B.Cno = D.Cno
                             and D.Ano = A.Ano
                             and A.Bname="perry");
```

SELECT综合实例

银行数据库关系模式为：
Branch=(Bname, Bcity, Bassets)
Customer=(Cno, Cname, Cstreet, Ccity)
Loan=(Lno, Bname, amount)
Borrower=(Cno, Lno)
Account=(Ano, Bname, balance)
Depositor=(Cno, Ano)

- 思考题解答

4、找出平均余额最高的支行；

分步法：

1）找出每个银行的平均余额集合，用一个select块和集函数avg()从account中查询；

2）对上述结果用集合函数all()求出平均余额的所有

3）从account中选择平均余额大于all()函数返回值的元组

```
select  Bname
from   account
group by  Bname
having (avg(balance) >=all(select  avg(balance)
                           from   account
                           group by  Bname));
```

SELECT综合实例

- 思考题解答：

5、找出住在Harrison且在银行中至少有三个账户的客户的平均余额；

解法：整体法 关键词（客户、有账户、账户余额）

1）找from customer ⋈ depositor ⋈ account

2）用where过滤 Ccity=“Harrison” ，得出住在Harrison且在银行中有账户的客户

3）用group分组 求出每个Harrison客户的账户数

4）用having 对每个分组过滤

```
select  Cname, avg(balance)
from   customer C, depositor D, account A
where  C.Cno=D.Cno and D.Ano = A.Ano  and Ccity="Harrison"
group by D.Cno
having  count(distinct D.Ano)>=3)
```

SELECT综合实例

6、找出在Brooklyn的所有支行都有账户的客户;

方法一：双重否定。

```
select  distinct S.Cno
from   Depositer S
where   not exists ( select  *
                from   Branch B
                where   Bcity= ‘Brooklyn’  and not exists
                    ( select  *
                      from   depositor T, account R
                      where   T.Ano=R.Ano
                              and  S.Cno=T.Cno
                              and B.Bname= R.Bname ));
```

不存在Brooklyn的某家支行，该客户在该支行没有账户

SELECT综合实例

6、找出在Brooklyn的所有支行都有账户的客户;

方法二：集合A包含集合B 等价于 not exists (B - A);

```
select  distinct  Cno
   from   depositor S
   where   not exists (( select  Bname
                         from   branch
                         where   Bcity='Brooklyn')
                         except
                        ( select  Bname
                        from   depositor T, account R
                        where   T.Ano=R.Ano
                                and  S.Cno=T.Cno));
```

不存在Brooklyn的某家支行，该客户在该支行没有账户

课堂测试题

考虑下面关系数据库 employee (ename, street, city)
works (ename, cname, salary)
company (cname, city)
manages (ename, manager-name)

用SQL语言表示下面查询?

1）找出“first bank corporation”的所有员工姓名。
2）找出“first bank corporation”的所有员工姓名和居住城市。
3）找出“first bank corporation”的所有收入在10000元以上员工的姓名和居住城市。
4）找出所有居住地与工作的公司在同一个城市的员工姓名。
5）找出与其经理居住在同一个城市、同一街道的所有员工姓名。
6）找出不在“first bank corporation”工作的所有员工姓名。
7）找出比“small bank corporation”的所有员工收入都高的所有员工姓名。
8）假设公司可以在几个城市部署分部。找出在“small bank corporation”公司所在的各个城市都有分部的公司。
9）假设公司可以在几个城市。找出位于“small bank corporation”所在的各个城市的所有公司。