数据库系统原理

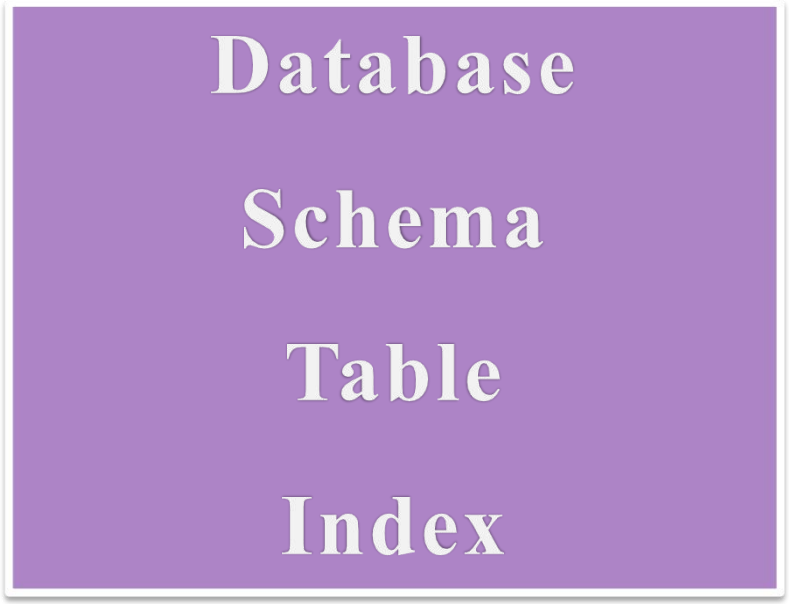
教程：数据库系统概论（第5版）

结合：CMU 15-445/645 INTRO TO DATABASE SYSTEMS

华中科技大学 计算机学院

左琼

第三章 关系数据库标准语言SQL



3.4 数据查询

- DML语言

  DBMS给上层提供的DML语言主要包括表的增、删、查，其中查询语句是核心DML语句。

[例] 查询选修2号课程且成绩在90分以上的所有学生。

关系代数：

$$\pi_{student.sno,\ sname}(\sigma_{cno='2' \wedge grade>90}(Student \bowtie SC)))$$

等价的SQL语句：

```
SELECT   Student.Sno, Sname
  FROM   Student, SC
  WHERE  Student.Sno = SC.Sno AND
         Cno= '2' AND Grade > 90;
```

3.4 数据查询

1. SELECT 语句的基本句法：

在关系代数中最常用的式子是下列表达式：

$$\pi_{A1,\ldots,An}(\sigma_F(R1\times\ldots\times Rm))$$

这里 R1,…,Rm为关系，F是公式，A1,…,An为属性。

针对上述表达式，SQL语言使用下列句型来表示：

```
SELECT  A1,…,An
FROM  R1,…,Rm
WHERE   F
```

此句型是从关系代数表达式演变过来的，但WHERE子句中的条件表达式 F 要比关系代数中公式更灵活。

3.4 数据查询

2. 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 ] ];
```

注：

- 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。

3.4 数据查询

3. SELECT 语句的一般执行过程

- 读取FROM子句中基本表及视图的数据，并执行笛卡尔积操作；
- 选取其中满足WHERE子句中条件表达式的元组；
- 按GROUP BY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；
- 按ORDER BY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。

查询语句的语法虽然看上去简单易用，实际上却是SQL语言中最难掌握的语句。主要难点在于查询条件的正确表达。

3.4 数据查询

3.4.1 单表查询

3.4.2 连接查询

3.4.3 嵌套查询

3.4.4 集合查询

3.4.5 基于派生表的查询

3.4.1 单表查询

单表查询 —— 只涉及到一张表的查询。

- SELECT子句的<目标列表达式>可以为：
  - 列名[,列名]…；
  - *
  - 算术表达式；
  - 字符串常量；
  - 函数；
  - 列别名

1. 选择表中的若干列

[例] 查询全体学生的学号与姓名（指定列）。

```
SELECT Sno，Sname FROM Student；
```

[例] 查询全部学生的所有信息。

```
SELECT * FROM STUDENT;
```

1. 选择表中的若干列

- 问题：若希望的查询结果表中属性无法直接用SELECT得出，但可以通过运算得出，如何处理这种派生属性?

[例] 列分别为算术表达式、字符串常量、函数的示例，并使用列别名改变查询结果的列标题:

```
SELECT Sname NAME, 'Year of Birth: ' BIRTH,
        2000-Sage BIRTHDAY, LOWER(Sdept) DEPARTMENT
FROM Student;
```

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth	2000	cs
刘晨	Year of Birth	2001	cs

3.4.1 单表查询

2. 选择表中的若干元组

- 问题：对一个关系SELECT后，结果关系中可能出现重复元组，实现中，为提高效率，SELECT并不消除重复元组。

（1） 消除取值重复的行

如果没有指定DISTINCT关键词，则缺省为ALL。

[例] 查询选修了课程的学生学号。

```
SELECT Sno  FROM SC;
```

等价于：

```
SELECT ALL  Sno  FROM SC;   //结果中可以有多个相同元组
```

[例] 查询全部被选课程的课程号（去除重复值）。

```
SELECT  DISTINCT  Cno  FROM  SC;
```

2. 选择表中的若干元组

（2）查询满足条件的元组

- 如何从表中选择指定元组?

  对应于关系代数运算σ_P，SQL提供where子句解决表元组的选择。

- 格式：WHERE <条件表达式>

<条件表达式>是包含属性名的逻辑表达式P，通过P对元组进行筛选。

查 询 条 件	谓　词
比　较	=，>，<，>=，<=，!=，<>，!>，!<； NOT+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空　值	IS NULL，IS NOT NULL
多重条件(逻辑运算)	AND，OR，NOT

2. 选择表中的若干元组

（2）查询满足条件的元组

[例] 查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT Sno
  FROM  SC
  WHERE Grade<60;   //比较大小
```

[例] 查询年龄不在20~23岁之间的学生姓名、系别和年龄。

```
SELECT Sname，Sdept，Sage
   FROM    Student
   WHERE Sage NOT BETWEEN 20 AND 23;
```

//确定范围：闭区间

确定集合

- SQL中提供了元素与集合之间的比较运算符:
- 谓词： x IN <值表>，x NOT IN <值表>

其中 <值表> 是一个集合，从关系代数的角度看， 它是一个代数式，从SQL角度看，它是一个SELECT语句。

[例] 查询信息系（IS）、数学系（MA）和计算机系（CS）学生的姓名和性别。

```
SELECT Sname，Ssex
FROM  Student
WHERE Sdept IN ( 'IS'，'MA'，'CS' );
```

[例] 查询既不是信息系、数学系，也不是计算机科学系的学生的姓名和性别。

```
SELECT Sname，Ssex
FROM Student
WHERE Sdept NOT IN ( 'IS'，'MA'，'CS' );
```

字符匹配

- 为什么提供字符匹配运算符?

  关系数据库支持对集合的运算，实际应用中，往往需要从集合中找出类似于某个条件的元组，即模糊查询。SQL的字符匹配运算符为解决这类问题而提出。

- 谓词:

  [NOT] LIKE ‘<匹配串>’ [ESCAPE ‘<换码字符>’]

- 通配符:

  SQL规定符号百分号%及下划线_具有其他含义

  百分号% 代表任意长度的字符串

  下划线_ 代表任意一个字符

  <匹配串> 为可以含有通配符的字符串

  ESCAPE 是将百分号% 或下划线_转回其本意

字符匹配示例

[例] 查询所有姓刘学生的姓名、学号和性别。

```
SELECT Sname，Sno，Ssex FROM Student
        WHERE Sname LIKE '刘%' ;
```

[例] 查询姓“欧阳”且全名为三个汉字的学生的姓名。

```
SELECT Sname FROM Student
        WHERE Sname LIKE '欧阳_ _';
```

[例] 查询DB_Design课程的课程号和学分。

```
SELECT Cno，Ccredit FROM Course
    WHERE Cname LIKE 'DB\_Design' ESCAPE '\';
```

[例] 查询以“DB_”开头，且倒数第3个字符为 i的课程的详细情况。

```
SELECT * FROM Course
     WHERE Cname LIKE 'DB\_%i_ _' ESCAPE '\';
    // ESCAPE ' \ ' 表示 “ \ ” 为换码字符
```

2. 选择表中的若干元组

（2）查询满足条件的元组

[例] 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。

```
SELECT Sno，Cno FROM SC
WHERE Grade IS NULL;
```

Notice: “IS” 不能用 “=” 代替。

- 多重条件查询：

  逻辑运算符： AND和 OR来联结多个查询条件

  - AND的优先级高于OR
  - 可以用括号改变优先级
  - 可用来实现多种其他谓词： IN， BETWEEN…AND

多重条件查询

- 逻辑运算符：AND和OR可以用来将多个简单查询条件复合成更加复杂的条件

[例] 查询信息系、数学系和计算机系20岁以下的学生的姓名和性别。

```
SELECT Sname，Ssex
 FROM Student
 WHERE Sdept IN (‘IS’，‘MA’，‘CS’) AND Sage<20;
```

可改为:

```
SELECT Sname，Ssex
 FROM Student
 WHERE （Sdept=‘IS’ OR  Sdept=‘MA’ OR
          Sdept= ‘CS’ ） AND Sage<20;
```

优先级：NOT > AND > OR

3. ORDER BY子句

ORDER BY 子句：

- 可以按一个或多个属性列排序。如：按(系别，年龄)排序；
- 升序：ASC；降序：DESC；缺省值为升序。

[例] 列出选修了课程号为 ‘C6’ 的所有学生的学号和成绩，并按分数的降序排列。

```
SELECT  sno, grade
  FROM    SC
  WHERE  cno='C6'
  ORDER  BY  grade  DESC;
```

注：当排序列含空值null时，SQLServer、Mysql认为空值最小，ASC：null排最前，DESC：反之。而Oracle认为空值最大，ASC：null排最后，DESC：反之。

4. 聚集函数

- SQL聚集函数：
  - 计数

    COUNT（[DISTINCT|ALL] *）：统计元组个数

    COUNT（[DISTINCT|ALL] <列名>）：统计一列中值的个数
  - 计算总和

    SUM（[DISTINCT|ALL] <列名>）
  - 计算平均值

    AVG（[DISTINCT|ALL] <列名>）
  - 最大最小值

    MAX（[DISTINCT|ALL] <列名>）

    MIN（[DISTINCT|ALL] <列名>）

集函数只能用于Select子句和Having子句中。

4. 聚集函数

[例] 查询学生总人数。

```
SELECT COUNT(*) FROM  Student;
```

[例] 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT Sno) FROM SC;
```

[例] 查询选修3号课程学生的最高分、最低分和平均成绩。

```
SELECT  MAX(grade), MIN(grade), AVG(grade)
FROM  SC
WHERE  cno=  '3'  ;
```

// 当集函数遇到NULL时，除COUNT(*),都跳过空值而只处理非空值。

5. GROUP BY子句

- 分组是按某（些）列的值对查询的结果进行分类。
- 格式： GROUP BY A1，A2，...,An

  其中：Ai为属性名

  按GROUP子句中指定的列的值进行分组，值相等的为一组。
- HAVING子句可以对分组后的结果作进一步的筛选。
- 当查询语句中有GROUP子句时，集函数作用的对象是一个分组的结果，而不是整个查询的结果。

[例] 求各个课程号及相应的选课人数。

```
SELECT Cno，COUNT(Sno) FROM SC
    GROUP BY Cno;
```

Cno	COUNT（Sno）
1	22
2	34
3	44
4	33
5	48

5. GROUP BY子句

- 注：分组后，一些详细信息可能损失，不能出现在SELECT结果中。
- 例如，下面的查询

```
SELECT Sno, Grade, COUNT(*)
    FROM SC
    WHERE Cno='2'
    GROUP BY Grade;
```

Sno	Cno	Grade
1	1	95
2	1	96
3	1	NULL
4	1	NULL

Sno	Grade	Count(*)
1	95	1
2	96	1
???	NULL	2

将出错，想一想，为什么？

- 一般来说，分组查询的SELECT目标列中只允许出现聚集函数和GROUP BY子句中出现过的列。

5. GROUP BY子句

□ 按分组计算结果排序

[例] 列出每门课程号及相应的选修人数，按选修人数的多少，从高到低排序。

```
SELECT   Cno, Count(Sno)
 FROM     SC
 GROUP BY   Cno
 ORDER BY   Count(Sno)   DESC;
```

Cno	Count(Sno)
3	95
1	48
2	33

单表查询

- 特殊语法* 求Top n

[例] 求1号课程的最高成绩,及取得最好成绩的学生学号。

Mysql

```
SELECT  Sno, Grade  FROM  SC
        WHERE   Cno='1'
        ORDER BY  Grade  DESC
        LIMIT 1;
```

SQL Server

```
SELECT TOP 1  WITH TIES sno, grade
FROM  SC
WHERE    cno='1'
ORDER    BY    grade  DESC;
```

等价于：

嵌套查询

```
SELECT
Sno, Grade  FROM  SC
WHERE  Grade
            = (SELECT MAX(Grade) FROM SC WHERE Cno = '1')
          AND  Cno = '1' ;
```

5. GROUP BY子句

- 问题： 有时，对于一个分组以后的结果集合希望使用限定条件选择部分分组，则可以使用Having 子句。
- 格式： Having P; P是谓词
- 注意： 由于HAVING 子句中的谓词P是在分组以后起作用的，因此P中可以使用聚集函数。

[例] 查询选修了三门以上课程的学生学号。

```
SELECT Sno FROM SC
    GROUP BY Sno
    HAVING COUNT(*)>3;
```

注意：统计结果和组别属性不一定非要出现在查询列表中，但非统计函数及非组别属性，一定不允许出现在查询列表中

5. GROUP BY子句

[例] 查询至少有3门课程成绩在90分以上的学生的学号及（90分以上的）课程数。

```
SELECT   Sno, COUNT(*)
 FROM   SC
 WHERE  Grade > =90
 GROUP BY  Sno
 HAVING   COUNT(*) > 3
```

- 注：WHERE子句与HAVING子句的区别: 作用对象不同。
  - WHERE子句作用于基本表或视图，从中选择满足条件的元组；
  - HAVING短语作用于组，从中选择满足条件的组。

5. GROUP BY子句

- 带CASE子句的HAVING*

[例] 列出至少有三门课程成绩90分以上的学生学号及其平均成绩。

```
SELECT  Sno, AVG(grade) as Avg_grade
  FROM  SC
  GROUP  BY  Sno
  HAVING  SUM ( CASE WHEN Grade >= 90 THEN  1 ELSE  0 END ) >= 3
```

Mysql

```
HAVING SUM ( IF ( Grade >= 90, 1,0 ) ) >= 3
```

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；自连接、嵌套连接

3.4.2 连接查询

- 连接查询：同时涉及多个表的查询
- 连接条件或连接谓词：用来连接两个表的条件

- 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>

- 连接字段：连接谓词中的列名称
  - 连接条件中的各连接字段类型必须是可比的，但名字不必是相同的

3.4.2 连接查询

1. 等值连接与非等值连接查询

等值连接：连接运算符为=

[例] 查询每个学生及其选修课程的情况。

```
SELECT Student.*, SC.* FROM Student, SC
    WHERE Student.Sno = SC.Sno;
```

查询结果：

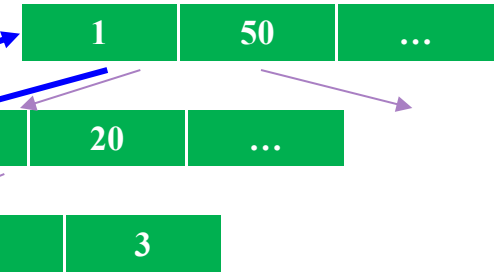
Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
200215121	李勇	男	20	CS	200215121	1	92
200215121	李勇	男	20	C			
200215121	李勇	男	20	C			
200215122	刘晨	女	19	C			
200215122	刘晨	女	19	CS			

改为自然连接：

```
SELECT Student.*, Cno, Grade
    FROM Student, SC
    WHERE Student.Sno = SC.Sno;
```

连接操作的执行方法

- 嵌套循环法(NESTED-LOOP)
- 排序合并法(SORT-MERGE)
- 索引连接法(INDEX-JOIN)
- 哈希法(HASH-JOIN)



Student

Sno	Sname	Ssex	Sage	Sdept
1	李勇	男	19	CS
2	刘晨	女	18	CS
3	王敏	女	18	MA

SC

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88
2	2	90
3	1	80

```
SELECT Sname, Cno, Grade
    FROM  Student, SC
    WHERE Student.Sno = SC.Sno;
```

2. 连接查询

[例] 求“数据结构”课程成绩大于85分的学生姓名和成绩，结果按成绩降序排列。

```
SELECT   STUDENT.sname，grade
FROM     STUDENT，SC，COURSE
WHERE    STUDENT.sno = SC.sno  AND
         SC.cno = COURSE.cno   AND
         COURSE.cname = '数据结构'  AND
         SC.grade > 85
ORDER BY  grade DESC;
```

连接条件

筛选条件

谁先做？

注：当不同的表中有同名属性时，属性名前要用表名限定。

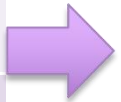
2.自身连接

一个表与其自己进行连接，称为表的自身连接。

[例] 求同时选修了1号课程和2号课程的学生学号。

```
SELECT  a.sno
FROM  CS a, CS b
WHERE  a.sno = b.sno  AND
       a.cno =  '1'  AND b.cno =  '2' ;
```

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88
2	2	90
3	1	80



a.Sno	a.Cno	a.Grade	b.Sno	b.Cno	b.Grade
1	1	92	1	1	92
1	1	92	1	2	85
1	2	85	1	1	92
1	2	85	1	2	85
……					

自身连接 （续）

[例] 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 FIRST, Course SECOND
WHERE FIRST.Cpno = SECOND.Cno;
```

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

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

查询结果：

Cno	Cpno
1	7
3	5
4	NULL
5	6
7	NULL

连接

```
select student.sno,sname,cname,grade
from sc
join student on student.sno = sc.sno
join course on  course.cno = sc.cno;
```

- SQL中表的连接有两种表示方法：
- 方法1：在FROM子句中指明进行连接的表名，在WHERE子句中指明连接的列名及其连接条件，如前面的例子。
- 方法2：利用关键字JOIN进行连接。具体分为以下几种：
  - INNER JOIN ： 返回符合连接条件的记录；
  - LEFT [OUTER] JOIN：返回符合连接条件的数据行以及左边表中不符合条件的数据行，此时右边数据行以NULL来显示，称为左连接；
  - RIGHT [OUTER] JOIN：返回符合连接条件的数据行以及右边表中不符合条件的数据行，此时左边数据行以NULL来显示，称为右连接；
  - FULL [OUTER] JOIN：返回符合连接条件的数据行以及左边表和右边表中不符合条件的数据行，此时缺乏数据的数据行会以NULL来显示；
  - CROSS JOIN：返回笛卡儿积。
  - 通过关键词ON与JOIN相对应，指明连接条件。

3. 外连接

- LEFT JOIN、RIGHT JOIN与 FULL JOIN统称为外连接。可用来显示不满足连接条件的元组。
- 某些查询要求只能用外连接来表达。

Sno	Sname
1	李勇
2	刘晨
3	王敏

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88

［例］查询所有学生的学号、姓名、所选课程的课程号以及这门课的成绩（没有选修任何课程的同学信息也要求被查询出来，相应的选课信息显示为空）。

```
SELECT STUDENT.sno, sname, cno, grade
FROM   STUDENT  LEFT OUTER JOIN  SC
ON STUDENT.SNO=SC.SNO;
```

Student.Sno	Sname	Cno	Grade
1	李勇	1	92
1	李勇	2	85
2	刘晨	1	88
3	王敏	NULL	NULL

3.4.3 嵌套查询

什么是嵌套查询?

- 一个SELECT-FROM-WHERE语句称为一个查询块；将一个查询块嵌套在另一个查询块的 WHERE子句或HAVING短语的条件中的查询称为嵌套查询，相当于在SELECT中调用另一段SELECT。

【例】查询选修了2号课程的学生姓名。

```
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') ;
```

- 子查询不能使用ORDER BY子句；
- 层层嵌套方式反映了 SQL语言的结构化；
- 有些嵌套查询可以用连接运算替代。

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) 属于运算：该成员t是否属于某个select结果集合R；IN谓词

$t \in R$ 是否成立?

(2) 比较运算：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?

(3) 逻辑运算：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)）

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     //子查询
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
这时可以使用IN、ANY、ALL或EXISTS谓词

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';
```

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

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

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88
2	2	90
3	1	80

如何实现?

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, 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

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；

3.4.3 嵌套查询

- 问题：在嵌套查询情形二中，若需判断关系中元组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

- 谓词语法：

  ANY(R)： R的任意一个值

  ALL(R)： R所有值

- 谓词语义：与关系运算符配合使用

  > ANY(R)：至少比R...的某一个大

  > ALL(R)：比R...所有大

> 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

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。本质上是一个返回值为“真”/“假”的集函数，用于判断一个集合是否为空。 EXISTS（R），R为非空则返回真。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。
- 带EXISTS的子查询的用法：
  - 不同形式的查询间的替换: 带IN谓词、比较运算符、ANY和ALL谓词的子查询 与 带EXISTS谓词的子查询 等价替换
  - 用EXISTS/NOT EXISTS实现全称量词(难点)
  - 用EXISTS/NOT EXISTS实现逻辑蕴函(难点)

3.4.3 嵌套查询

- 不同形式的查询间的替换示例

[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'  );
```

item in (select attribute from T)

等价于：

```
exists (select *  from  T
        where  attribute = item  )
```

等价于：

方法1：

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

方法2：

```
SELECT  Sname
 FROM   Student
 WHERE  Sno IN
      （SELECT Sno
        FROM SC
        WHERE Cno='1');
```

3.4.3 嵌套查询

```
SELECT Sname，Sage
FROM  Student
WHERE Sage < ANY (SELECT  Sage
                  FROM   Student
                  WHERE Sdept= 'CS' )
                  AND Sdept <> ‘CS’ ;
```

□ 不同形式的查询间的替换示例

[例2] 查询其他系中比计算机科学系某一学生年龄

【分析】

1）用EXISTS函数判断任一个学生t，其年龄比计算机系任一学生年龄小

2）从其他系学生表中，搜索让上述逻辑式为真的元组

```
SELECT Sname，Sage
FROM  Student S1
WHERE EXISTS (SELECT * FROM Student S2
                    WHERE Sdept = 'CS' AND S1.Sage < S2.Sage)
              AND Sdept <> 'CS';
```

注：S1是其他系学生元组变量，S2是计算机系元组变量。

3.4.3 嵌套查询

[实例] 查询2015-3-20,2015-5-1分别

- 2015-3-20 9:00pm – 次日 2:00am
  中国地质大学(28991,7202)
- 2015-5-1 9:00pm – 次日 2:00am:
  湖北大学(28961,10522)

主叫 Oid	被叫 Tid	开始时间 start	时长 period	区域 lac	基站 cel

```
select oid,tid,start,period,lac,cell
from  calling  A
where lac = 28991 and
        cell = 7202 and
        start  between '2015-3-20 21:00' and
                       '2015-3-21 2:00' and
        exists  (
                select * from calling B
                where A.oid = B.oid  and
                        lac =  28961 and
                        cell = 10522 and
                        start between '2015-5-1 21:00'
                                and   '2015-5-2 2:00'
```

3.4.3 嵌套查询

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：

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

□ 用EXISTS/NOT EXISTS实现全称量词示例

[例] 查询选修了全部课程的学生姓名。

【分析】“选修了全部课程” ⇔ “没有一门课他没有选”

```
SELECT sname  FROM STUDENT
WHERE  NOT EXISTS          //不存在学生t没选任一门课c
      (SELECT *
       FROM  COURSE
       WHERE  NOT EXISTS          //学生t 选修了一门课c，为“假”，即：t没选课程c
        (SELECT *
           FROM   SC
           WHERE  sno=STUDENT.sno AND          //学生t 选修了一门课c
                  cno=COURSE.cno))
```

3.4.3 嵌套查询

用EXISTS/NOT EXISTS实现逻辑蕴含示例

[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：

- 用逻辑蕴函表达：查询学号为x的学生，对所有的课程y，只要200215122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200215122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200215122选修了y，而学生x没有选。

3.4.3 嵌套查询

- [续上例]查询至少选修了学生200215122选修的全部课程的学生号码。
- 不存在这样的课程y，学生200215122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 = ' 200215122 '  AND
               NOT EXISTS
               (SELECT *
                FROM SC SCZ
                WHERE SCZ.Sno=SCX.Sno AND
                      SCZ.Cno=SCY.Cno));
```