数据库系统原理

第3章　关系数据库语言2

分级通关平台，交流

基础篇

第1章 绪论
第2章 关系数据库*3
第3章 标准语言SQL*3
第4章 数据库安全性
第5章 数据库完整性
实验1

设计与应用开发篇

第6章 关系数据理论*2
第7章 数据库设计
实验2
第8章 数据库编程

系统篇

第9章 *关系查询处理和优化
实验3
第10章 数据库恢复技术
第11章 并发控制
实验4

内容提要

关系数据库标准语言SQL-1

1 SQL概述

2 学生-课程数据库

3 数据定义

4 数据查询—单表

SQL-2

4 数据查询—多表

SQL-3

5 数据更新

6 空值的处理

7 视图

4. 数据查询—多表

1 单表查询

2 连接查询

3 嵌套查询

4 集合查询

5基于派生表的查询

select * from 表1, 表2, ……;　然后加条件；

数据查询—多表

- 语句格式

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>]…|(SELECT 语句)

[AS]<别名>

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]];

select * from 表名；　然后加条件；

连接查询

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

```
select * from T1,T2
  where [<表名1>.]<列名1>  <比较运算符>  [<表名2>.]<列名2>
  order by b1;
```

连接查询（等值与非等值连接查询）

- 等值连接：连接运算符为=

[例 3.49] 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
201215121	李勇	男	20	CS	201215121	1	92
201215121	李勇	男	20	CS	201215121	2	85
201215121	李勇	男	20	CS	201215121	3	88
201215122	刘晨	女	19	CS	201215122	2	90
201215122	刘晨	女	19	CS	201215122	3	80

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

学生-课程 数据库

学生-课程模式 S-T：学生关系Student、课程关系Course和选修关系SC

Student

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

Course

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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

SC

学号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

连接查询(外连接)

- 外连接：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

[例 3.49] 查询每个学生及其选修课程的情况

```
SELECT Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade
    FROM  Student  LEFT OUT JOIN SC ON
    (Student.Sno=SC.Sno);
```

查询结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
201215121	李勇	男	20	CS	1	92
201215121	李勇	男	20	CS	2	85
201215121	李勇	男	20	CS	3	88
201215122	刘晨	女	19	CS	2	90
201215122	刘晨	女	19	CS	3	80
201215123	王敏	女	18	MA	NULL	NULL
201215125	张立	男	19	IS	NULL	NULL

连接查询（等值与非等值连接查询）

- 一条SQL语句可以同时完成选择和连接查询，这时WHERE子句是由连接谓词和选择谓词组成的复合条件。

[例 3.51]查询选修2号课程且成绩在90分以上的所有学生的学号和姓名。

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

连接查询(自身连接)

- 自身连接：一个表与其自己进行连接
  - 需要给表起别名以示区别
  - 由于所有属性名都是同名属性，因此必须使用别名前缀

[例 3.52]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;
```

查询结果：

Cno	Pcno
1	7
3	5
5	6

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

FIRST表（Course表）

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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）

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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

连接查询（多表连接）

- 多表连接：两个以上的表进行连接

[例3.54]查询每个学生的学号、姓名、选修的课程名及成绩

```
SELECT Student.Sno, Sname, Cname, Grade
  FROM    Student, SC, Course    /*多表连接*/
  WHERE Student.Sno = SC.Sno
                AND SC.Cno = Course.Cno;
```

连接查询

- 连接查询的多表连接过程是怎样的？
- 如何提高连接查询的效率？

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

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

```
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 ');
```

  - 上层的查询块称为外层查询或父查询
  - 下层查询块称为内层查询或子查询

嵌套查询

- SQL语言允许多层嵌套查询
  - 即一个子查询中还可以嵌套其他子查询
  - 子查询不能使用ORDER BY子句
- 不相关子查询：子查询的查询条件不依赖于父查询
  - 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
  - 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

嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY或ALL谓词的子查询

4.带有EXISTS谓词的子查询

嵌套查询（IN谓词）

[例 3.55] 查询与“刘晨”在同一个系学习的学生。（分步来完成）

① 确定“刘晨”所在系名

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

结果为： CS

② 查找所有在CS系学习的学生。

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

结果为：

Sno	Sname	Sdept
201215121	李勇	CS
201215122	刘晨	CS

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。

嵌套查询（IN谓词）

[例 3.55] 查询与“刘晨”在同一个系学习的学生。

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

用自身连接完成查询要求

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

嵌套查询（IN谓词）

[例 3.56]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，为3号

用连接查询实现：

```
SELECT Sno,Sname   FROM    Student,SC,Course
WHERE Student.Sno = SC.Sno  AND
            SC.Cno = Course.Cno AND
            Course.Cname='信息系统';
```

嵌套查询（比较运算符）

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

在[例 3.55]中，由于一个学生只可能在一个系学习，则可以用 = 代替IN：

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

嵌套查询（比较运算符）

[例 3.57]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);
```

嵌套查询（ ANY, ALL）

- 使用ANY或ALL谓词时必须同时使用比较运算

语义为：

> ANY	大于结果中的某个值	> ALL	大于结果中的所有值
< ANY	小于结果中的某个值	< ALL	小于结果中的所有值
>= ANY	大于等于某个值	>= ALL	大于等于所有值
<= ANY	小于等于某个值	<= ALL	小于等于所有值
= ANY	等于结果中的某个值	=ALL	等于结果中的所有值
!=（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		

嵌套查询（ ANY, ALL）

[例 3.58] 查询非计算机科学系中比计算机科学系任意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= ' CS ')
        AND Sdept <> 'CS ' ;          /*父查询块中的条件 */
```

用聚集函数实现[例 3.58]

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

嵌套查询（ ANY, ALL）

[例 3.59] 查询非计算机科学系中比计算机科学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= ' CS ')
  AND Sdept <> ' CS ';
```

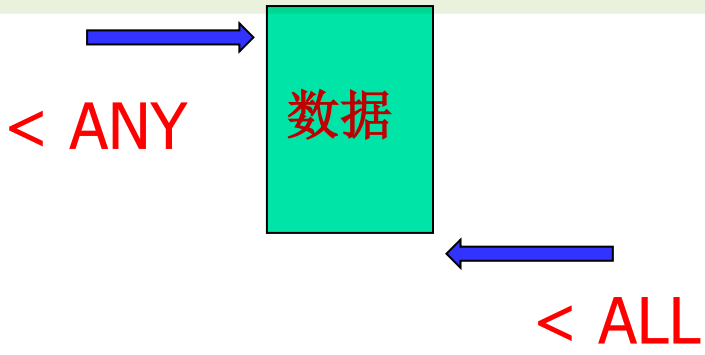
方法二：用聚集函数

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

嵌套查询（ ANY, ALL）

表3.7 ANY（或SOME），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



嵌套查询（ EXISTS）

- 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，则外层的WHERE子句返回真值
    - 若内层查询结果为空，则外层的WHERE子句返回假值
  - 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。
- NOT EXISTS谓词
  - 若内层查询结果非空，则外层的WHERE子句返回假值
  - 若内层查询结果为空，则外层的WHERE子句返回真值

嵌套查询（ EXISTS）

[例 3.60]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
- 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送入结果表

```
SELECT Sname      FROM Student
WHERE EXISTS
          (SELECT *    FROM SC
           WHERE Sno=Student.Sno AND Cno= ' 1 ');
```

嵌套查询（ EXISTS）

- 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

[例 3.55]查询与“刘晨”在同一个系学习的学生。

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');
```

嵌套查询（ EXISTS）

[例 3.62] 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
                 )
        );      /* 不存在没有修的课程 */
```

嵌套查询（ EXISTS）

[例 3.63]查询至少选修了学生201215122选修的全部课程的学生号码。

- 不存在这样的课程y，学生201215122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 = ' 201215122 '  AND
                     NOT EXISTS
                     (SELECT *
                      FROM SC SCZ
                      WHERE SCZ.Sno=SCX.Sno AND
                                SCZ.Cno=SCY.Cno));
```

嵌套查询（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

嵌套查询（集合查询）

[例 3.64] 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;
```

- 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

嵌套查询（集合查询）

[例 3.65] 查询选修了课程1或者选修了课程2的学生。

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
UNION
SELECT Sno
FROM SC
WHERE Cno= ' 2 ';
```

[例3.66] 查询计算机科学系的学生与年龄不大于19岁的学生的交集。

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

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

嵌套查询（集合查询）

[例 3.67]查询既选修了课程1又选修了课程2的学生。

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

[例3.67]也可以表示为：

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

嵌套查询（派生表）

- 子查询还可以出现在FROM子句中，生成的临时派生表（Derived Table）成为主查询的查询对象

[例3.57]找出每个学生超过他自己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_sc.avg_sno
      and SC.Grade >=Avg_sc.avg_grade
```

嵌套查询（派生表）

- 如果子查询中没有聚集函数，派生表可以不指定属性列，子查询SELECT子句后面的列名为其缺省属性。

[例3.60]查询所有选修了1号课程的学生姓名，可以用如下查询完成：

```
SELECT Sname
FROM   Student,
 (SELECT Sno FROM SC WHERE Cno=' 1 ') AS SC1
       WHERE  Student.Sno=SC1.Sno;
```

小结

- SELECT语句格式

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>]…|（SELECT 语句)

[AS]<别名>

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]];

select * from 表名;　然后加条件;