《数据库系统概论》实验指导书

华 中 科 技 大 学

网络空间安全学院

2019 年 11 月

实验前的准备工作

实验的环境为 Windows 操作系统。

数据库使用 Microsoft SQLServer 或者 MySQL 免费开源版。

实验前请自己下载相关的数据库管理系统(DBMS)，可以是最新版或者是任意可运行的版本。

请首先熟练掌握数据库管理系统的安装过程。Microsoft SQLServer 安装时建议选择混合模式的身份验证方式。记住系统管理员的密码。

针对 Microsoft SQLServer，要熟练掌握服务管理平台和查询分析器的界面操作。针对 MySQL 需要同时安装其可视化客户端管理工具，例如 navicat 或者 MSQL-Front 等，并熟练掌握其操作。需要先熟练掌握数据库创建、创建表、创建用户、使用不同的用户进行登录等基本操作。

最后一个实验需要进行数据库应用开发，需要先熟悉一门编程语言，例如 VC++、Delphi、PB、VB、Java、Python 等，或者各种.net 环境。

如果你准备使用 B/S 模式，还需要先熟练掌握 Tomcat、WebLogic 或者 IIS 等应用服务器的安装和应用。

如果使用 MySQL 开发 C/S 程序还需要安装其 ODBC 接口。

如果使用 Java 开发应用程序还需要熟悉 Java 的集成开发环境例如 Eclipse 或 MyEclipse 等，并安装 Java 插件。使用其他语言开发，也需要熟悉类似的集成开发环境。如果开发安卓类移动应用，需要安装安卓模拟器。

实验二　SQL 的复杂操作（4 学时）

1、实验目的

掌握 SQL 语言的数据多表查询语句和更新操作

2、实验内容

(1) 等值连接查询（含自然连接查询）与非等值连接查询
(2) 自身连接查询
(3) 外连接查询
(4) 复合条件连接查询
(5) 嵌套查询（带有 IN 谓词的子查询）
(6) 嵌套查询（带有比较运算符的子查询）
(7) 嵌套查询（带有 ANY 或 ALL 谓词的子查询）
(8) 嵌套查询（带有 EXISTS 谓词的子查询）
(9) 集合查询
(10) update 语句用于对表进行更新
(11) delete 语句用于对表进行删除
(12) insert 语句用于对表进行插入

3、实验要求

(1) 熟练掌握 SQL 的连接查询语句
(2) 熟练掌握 SQL 的嵌套查询语句
(3) 掌握表名前缀、别名前缀的用法
(4) 掌握不相关子查询和相关子查询的区别和用法
(5) 掌握不同查询之间的等价替换方法（一题多解）及限制
(6) 熟练掌握 SQL 的数据更新语句 INSERT、UPDATE、DELETE
(7) 记录实验结果，认真完成实验报告

4、实验步骤

4.1 使用上次实验室的数据库，如果没有保存，则重新建立，并输入数据。

4.2 对学生关系 Student、课程关系 Course 和选修关系 SC 进行多表查询

4.2.1 基本练习

（1）等值连接查询与自然连接查询

例如：查询每个学生及其选修课的情况。

```
SELECT Student.*, SC.*
FROM Student, SC
WHERE Student.Sno = SC.Sno; /*  一般等值连接  */
```

又如：查询每个学生及其选修课的情况（去掉重复列）。

```
SELECT Student.Sno, Sname, Ssex, Sage, Cno, Grade
FROM Student, SC
WHERE Student.Sno = SC.Sno;    /*  自然连接--特殊的等值连接  */
```

（2）自身连接查询

例如：查询每一门课的间接先修课。

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

（3）外连接查询

例如：查询每个学生及其选修课的情况（要求输出所有学生--含未选修课程的学生的情况）

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

（4）复合条件连接查询

例如：查询选修了 2 号课程而且成绩在 90 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;
```

又如：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
WHERE Student.Sno = SC.Sno AND
            SC.Cno = Course.Cno;
```

（5）嵌套查询（带有 IN 谓词的子查询）

例如：查询与“刘晨”在同一个系学习的学生的学号、姓名和所在系。

```
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept IN
            (SELECT Sdept
              FROM Student
              WHERE Sname = '刘晨');     /*  解法一*/
```

可以将本查询中的 IN 谓词用比较运算符‘=’来代替：

```
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept =
            (SELECT Sdept
              FROM Student
              WHERE Sname = '刘晨');     /*  解法二*/
```

也可以使用自身连接完成以上查询：

```
SELECT s1.Sno, s1.Sname, s1.Sdept
FROM Student s1, Student s2
WHERE s1.Sdept = S2.Sdept AND
              s2.Sname = '刘晨';          /*  解法三*/
```

还可以使用 EXISTS 谓词完成本查询：

```
SELECT Sno, Sname, Sdept
FROM Student S1
WHERE EXISTS
      (SELECT *
        FROM Student S2
        WHERE S2.Sdept=S1.Sdept AND S2.Sname='刘晨'); /*  解法四*/
```

又如：查询选修了课程名为“信息系统”的学生号和姓名。

```
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'
                          )
            );
```

也可以使用连接查询来完成上述查询：

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

（6）嵌套查询（带有比较运算符的子查询）
例如：找出每个学生超过他所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);
```

（7）嵌套查询（带有 ANY 或 ALL 谓词的子查询）
例如：查询其他系中比计算机系某个学生年龄小的学生的姓名和年龄。

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

本查询也可以使用聚集函数来实现：

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

又如：查询其他系中比计算机系所有学生年龄都小的学生的姓名和年龄。

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

也可以使用聚集函数来实现：

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

（8）嵌套查询（带有 EXISTS 谓词的子查询）
例如：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
                 WHERE Sno=Student.Sno AND Cno='1');
```

又如：查询所有未选修 1 号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
              (SELECT *
                 FROM SC
                 WHERE Sno=Student.Sno AND Cno='1');
```

可以使用带有 EXISTS 谓词的子查询实现全称量词或蕴涵逻辑运算功能：
例如：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));
```

又如：查询至少选修了学生 200215122 选修的全部课程的学生号码。

```
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));
```

（9）集合查询

例如：查询计算机系的学生以及年龄不大于 19 岁的的学生。

```
SELECT *
FROM Student
WHERE Sdept='CS'
UNION        /*并集运算*/
SELECT *
FROM Student
WHERE Sage<=19;
```

可以改用多重条件查询：

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

又如：查询既选修了课程 1 又选修了课程 2 的学生（交集运算）。

```
SELECT Sno
FROM SC
WHERE Cno='1'
INTERSECT          /*交集运算*/
SELECT Sno
FROM SC
WHERE Cno='2';
```

可以使用嵌套查询：

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

思考：能不能改用多重条件查询？

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

再如：查询计算机系的学生与年龄不大于 19 岁的学生的差集。

```
SELECT *
FROM Student
WHERE Sdept='CS'
EXCEPT              /*差集运算*/
SELECT *
FROM Student
WHERE Sage<=19;
```

可以改用多重条件查询：

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

(10) update 语句用于对表进行更新

例如：将信息系所有学生的年龄增加 1 岁。

```
UPDATE Student
SET Sage= Sage+1
WHERE Sdept=' IS '
```

(11) delete 语句用于对表进行删除

例如：删除学号为 95019 的学生记录。

```
DELETE
FROM Student
WHERE Sno='95019'
```

(12) insert 语句用于对表进行插入

例如：插入一条选课记录('95020'，'1 ')。

```
INSERT
 INTO SC(Sno，Cno)
 VALUES (' 95020 '，' 1 ')
```

4.2.2 扩展练习（要求写出并执行 SQL 语句完成以下各种操作，记录查询结果）

（1）查询每门课程及其被选情况（输出所有课程中每门课的课程号、课程名称、选修该课程的学生学号及成绩--如果没有学生选择该课，则相应的学生学号及成绩为空值）。

（2）查询与“张立”同岁的学生的学号、姓名和年龄。（要求使用至少 3 种方法求解）

（3）查询选修了 3 号课程而且成绩为良好（80~89 分）的所有学生的学号和姓名。

（4）查询学生 200215122 选修的课程号、课程名

（思考：如何查询学生 200215122 选修的课程号、课程名及成绩？）

（5）找出每个学生低于他所选修课程平均成绩 5 分以上的课程号。（输出学号和课程号）

（6）查询比所有男生年龄都小的女生的学号、姓名和年龄。

（7）查询所有选修了 2 号课程的学生姓名及所在系。

（8）使用 update 语句把成绩为良的学生的年龄增加 2 岁，并查询出来。

（9）使用 insert 语句增加两门课程：C 语言和人工智能，并查询出来

（10）使用 delete 语句把人工智能课程删除，并查询出来。