数据库系统原理实验指导书

邢振祥

电子与信息工程系计算机应用教研室

2010-11-18

《数据库系统原理》为计算机科学与技术、软件工程和网络工程专业核心课程，是一门实用性较强的课程，也是计算机学科的基础课程之一。本课程的教学目的是使学生了解和熟悉数据库系统的实现原理和应用开发基础。要求学生了解数据库特别是关系数据库的基本概念，发展历史，和发展动向；理解关系数据库的查询优化，规范化理论，并发控制，安全性完整性控制等相关理论；掌握关系代数、sql 语言使用，数据库设计，开发，运行和维护等相关技术。

本课程总学时为 48 学时，其中理论课程学时为 40 学时，实验学时为 8 学时，实验内容安排如下：

1．创建数据库和表（2 学时）

2．查询数据库（2 学时）

3．创建和使用视图、索引（2 学时）

4．实现数据完整性和安全性（2 学时）

实验一　基本表的定义、删除与修改（2 课时，验证）

一、实验目的

熟练掌握基本表的定义、删除与修改，为后继学习作准备。

二、实验要求

1．了解并掌握 SQL 查询分析器及企业管理器的使用;

2．掌握基本表的定义、删除与修改。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

数据表的定义、删除及修改。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．验证如下例题：

表 1.1 关系 Students

Sno	Sname	Ssex	Sage	Sdept
S01	王建平	男	21	自动化
S02	刘华	女	19	自动化
S03	范林军	女	18	计算机
S04	李伟	男	19	数学
S05	黄河	男	18	数学
S06	长江	男	20	数学

表 1.2 关系 Courses

Cno	Cname	Pre_Cno	Credits
C01	英语		4
C02	数据结构	C05	2
C03	数据库	C02	2
C04	DB_设计	C03	3
C05	C++		3
C06	网络原理	C07	3
C07	操作系统	C05	3

表 1.3 关系 Reports

Sno	Cno	Grade
S01	C01	92
S01	C03	84
S02	C01	90
S02	C02	94
S02	C03	82
S03	C01	72
S03	C02	90
S04	C03	75

1 定义基本表

例 1.1　建立表 1.1 所示的学生表 Students，每个属性名的意义为 Sno-学号、Sname-姓名、Ssex-性别、Sage-年龄、Sdept-所在系。这里要求 Sno 和 Sname 不能为空值，且取值唯一。

```
CREATE TABLE Students                          /*列级完整性约束条件*/
    (Sno    CHAR(5) NOT NULL,                  /* Sno 不能为空值*/
    Sname    CHAR(20) NOT NULL,                /*Sname 不能为空值*/
    Ssex    CHAR(2),
    Sage    INT,
    Sdept    CHAR(15),
    CONSTRAINT un_Sno UNIQUE(Sno),             /* Sno 取值唯一的约束*/
    CONSTRAINT un_Sname UNIQUE(Sname));    /* Sname 取值唯一的约束*/
```

说明：在 Microsoft SQL Server 2000 的查询分析器(Query Analyzer)中使用单条 SQL 语句，其末尾不需要分号“;”作为命令结尾标记。通常，SQL Server 2000 对大多数末尾带有分号的 SQL 命令都能顺利执行，但对少数的 SQL 命令，末尾若带分号，则 SQL Server 2000 会给出错误信息提示。 比如，若在**例 1.59** 的 SQL 命令末尾加上一个分号“;”，SQL Server 2000 就会出现“Incorrect syntax near ';'”的提示，虽然 SQL Server 2000 实际上已经执行了该命令。

例 1.1-1 建立表 1.2 所示的课程表 Courses，其属性名意义分别为 Cno-课程号, Cname-课程名, Pre_Cno-先修课程号, Credits-学分。

```
CREATE TABLE Courses
   (Cno    CHAR(5) NOT NULL,                  /* Cno 不能为空值*/
    Cname    CHAR(20) NOT NULL,                 /*Cname 不能为空值*/
    Pre_Cno CHAR(5),
    Credits    INT,
    CONSTRAINT un_Cno UNIQUE(Cno));            /*Cno 取值唯一的约束*/
```

例 1.1-2 建立表 1.3 所示的成绩表 Reports。其中的属性名意义分别为 Sno-学号，Cno-课程号和 Grade-考试成绩。

```
 CREATE TABLE Reports
    ( Sno    CHAR(5) NOT NULL,                  /* Sno 不能为空值*/
      Cno    CHAR(5) NOT NULL,                  /* Cno 不能为空值*/
      Grade    INT,
      CONSTRAINT Sno_Cno UNIQUE(Sno,Cno));     /*Sno+Cno 取值唯一的约束*/
```

2 修改基本表

例 1.2 向基本表 Students 中增加“入学时间”属性列，其属性名为 Sentrancedate，数据类型为 DATETIME 型。

```
ALTER TABLE Students ADD Sentrancedate DATETIME;
```

例 1.3 将 Sage(年龄)的数据类型改为 SMALLINT 型。

```
ALTER TABLE Students ALTER COLUMN Sage SMALLINT;
```

例 1.4 删除 Sname(姓名)必须取唯一值的约束。

```
ALTER TABLE Students DROP CONSTRAINT un_Sname;
```

注意：SQL Server 2000 增加了删除属性的命令。比如，删除属性列 Sentrancedate 的命令为：

```
ALTER TABLE Students DROP COLUMN Sentrancedate;
```

说明：（1）为了保证后面例子能够顺利运行，请大家一定将属性列 Sentrancedate 从 Students 表中删除。

（2）为了调试 SQL 语句方便，这里没有在表 Reports 中增加参照完整性约束，甚至没有定义主键。等本章学完后，第 7 章的实验系统就是把这些约束全部加上了。

3 删除基本表

例 1.5 删除 Students 表。

```
DROP TABLE  Students;
```

说明：此表删除后，请立即用例 1.1 将其建立起来，以便后面的例子使用。

4 向表中添加元组

例 1.6 将一个学生元组(S01，王建平，男，21，计算机)添加到基本表 Students 中。

```
INSERT
INTO  Students
VALUES ('S01','王建平','男',21,'自动化');
```

说明：(1)请读者用这个命令将其余 5 个学生的元组也添加到基本表 Students 中。

(2)向 Courses 表插入元组(‘C01’,’英语’,’’,4)的命令为：

```
INSERT
INTO  Courses
VALUES ('C01','英语','',4);
```

请大家将其余 6 门课程的信息插入 Courses 表中。

例 1.7 将学习成绩的元组(‘S01’, ’C01’)添加到基本表 Reports 中。

```
INSERT
INTO  Reports(Sno, Cno)
VALUES (‘S01’,’C01’);
```

说明：请大家用这个命令将其余 7 个选课元组也添加到基本表 Reports 中.

实验二　建立与删除索引（2 课时，验证）

一、实验目的

熟练掌握索引的建立与删除的方法。

二、实验要求

1．掌握建立索引的二种方法，即在基本表中建立和用命令方式建立；

2．掌握删除索引的方法。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

索引的建立与删除。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．验证如下例题：

1 建立索引

例 1.8 为学生选课数据库中的 Students，Courses，Reports 三个表建立索引。其中 Students 表按 Sno(学号)升序建唯一索引，Courses 表按 Cno(课程号)升序建唯一索引，Reports 表按 Sno(学号)升序和 Cno(课程号)号降序建唯一索引。其语句为：

```
CREATE UNIQUE INDEX Stu_Sno ON Students(Sno);
CREATE UNIQUE INDEX Cou_Cno ON Courses(Cno);
CREATE UNIQUE INDEX Rep_Scno ON Reports(Sno ASC, Cno DESC);
```

例 1.9 在基本表 Students 的 Sname(姓名)和 Sno(学号)列上建立一个聚簇索引，而且 Students 中的物理记录将按照 Sname 值和 Sno 值的升序存放。其语句为：

```
CREATE CLUSTERED INDEX Stu_Sname_Sno ON Students(Sname, Sno);
```

2 删除索引

例 1.10 删除基本表 Reports 上的 Rep_SCno 索引。

```
DROP INDEX Reports.Rep_Scno;
```

实验三 sql 数据查询(2 课时，综合)

一、实验目的

熟练掌握查询语句的使用。

二、实验要求

1．掌握查询语句的一般格式;

2．掌握无条件、有条件查询及查询结果排序与分组。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

数据的各种查询方法。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．综合练习如下例题：

1 无条件查询

例 1.11 查询全体学生的详细记录。这是一个无条件的选择查询，其命令为：

```
SELECT　*                              /*这里的“*”等价于 ALL*/
```

```
FROM    Students;
```

其结果为表 1.3 中的全部数据。

例 1.12 查询全体学生的姓名(Sname)、学号(Sno)、所在系(Sdept)。这是一个无条件的投影查询，其命令为：

```
SELECT    Sname, Sno, Sdept
FROM    Students;
```

例 1.13 查询全体学生的姓名(Sname)、出生年份及学号(Sno)。由于 SELECT 子句的<目标列表达式>不仅可以是表中的属性列，也可以是表达式，故可以查询经过计算的值。其命令为：

```
SELECT    Sno, Sname, 2001-Sage
FROM    Students;
```

例 1.14 查询全体学生的姓名、出生年份和学号，要求用小写字母表示学号中的字母。其命令为：

```
SELECT    Sname, 'Birth:' Title, 1996-Sage BirthYear, LOWER(Sno) Lsno
FROM    Students;
```

例 1.15 查询选修了课程的学生学号。其命令为：

```
SELECT DISTINCT Sno
FROM    Reports;
```

2 条件查询

例 1.16 查询数学系全体学生的学号(Sno)和姓名 (Sname)。其命令为：

```
SELECT    Sno, Sname
FROM    Students
WHERE    Sdept='数学';
```

例 1.17 查询所有年龄在 18~22 岁(包括 18 岁和 22 岁)之间的学生姓名(Sname)及年龄(Sage)。其命令为：

```
SELECT    Sname, Sage
FROM    Students
WHERE    Sage>=18 AND Sage<=22;
```

例 1.18 查询年龄在 18~22 岁(包括 18 岁和 22 岁)之间的学生姓名(Sname)及年龄(Sage)。其命令为：

```
SELECT    Sname, Sage
FROM    Students
WHERE    Sage    BETWEEN 18 AND 22;
```

例 1.19 查询年龄不在 18-22 岁之间的学生姓名(Sname)及年龄(Sage)。其命令为：

```
SELECT    Sname, Sage
FROM    Students
WHERE    Sage    NOT    BETWEEN    18 AND 22;
```

例 1.20 查询自动化系、数学和计算机系学生的学号(Sno)、姓名(Sname)和性别(Ssex)。其命令为：

```
SELECT  Sno, Sname, Ssex
FROM  Students
WHERE  Sdept  IN ('自动化', '数学', '计算机');
```

等价于：

```
SELECT  Sname, Ssex
FROM  Students
WHERE  Sdept='自动化' OR Sdept='数学' OR Sdept='计算机';
```

例 1.21 查询既不是信息系、数学系、也不是计算机系的学生的姓名 (Sname)和性别(Ssex)。其命令为：

```
SELECT  Sname, Ssex
FROM  Students
WHERE  Sdept  NOT  IN ('自动化', '数学', '计算机');
```

例 1.22 查询所有姓刘的学生的姓名(Sname)、学号(Sno)和性别(Ssex)。其命令为：

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

例 1.23 查询姓“刘”且全名为 4 个汉字的学生的姓名(Sname)和所在系(Sdept)。其命令为：

```
SELECT  Sname, Sdept
FROM  Students
WHERE  Sname  LIKE  '刘____';
```

例 1.24 查询所有不姓刘的学生姓名(Sname)和年龄(Sage)。

```
SELECT  Sname, Sage
FROM  Students
WHERE  Sname NOT LIKE '刘%';
```

例 1.25 查询课程名为“DB_设计”的课程号(Cno)和学分(Credits)。其命令为：

```
SELECT  Cno, Credits
FROM  Courses
WHERE  Cname LIKE 'DB\_设计' ESCAPE '\';
```

例 1.26 查询以"DB_"开头，且倒数第 2 个汉字字符为“设”的课程的详细情况。其命令为：

```
SELECT  *
FROM  Courses
WHERE  Cname LIKE 'DB\_%设__'ESCAPE'\';
```

例 1.27 假设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。试查询缺少成绩的学生的学号(Sno)和相应的课程号(Cno)。其命令为：

```
SELECT  Sno, Cno
```

```
FROM    Reports
WHERE   Grade IS NULL;
```

例 1.28 查询所有有成绩的学生学号(Sno)和课程号(Cno)。其命令为：

```
SELECT Sno, Cno
FROM    Reports
WHERE   Grade IS NOT NULL;
```

3 查询结果排序

例 1.29 查询选修了 C03 号课程的学生的学号(Sno)和成绩(Grade)，并按成绩降序排列。其命令为：

```
SELECT   Sno, Grade
FROM    Reports
WHERE   Cno='C03'
ORDER BY Grade DESC;
```

例 1.30 查询全体学生情况，查询结果按所在系的系名(Sdpet)升序排列，同一系中的学生按年龄(Sage)降序排列。其命令为：

```
SELECT   *
FROM    Students
ORDER BY Sdept, Sage DESC;
```

4 集函数的使用

例 1.31 查询学生总人数。其命令为：

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

例 1.32 查询选修了课程的学生人数。其命令为：

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

例 1.33 计算选修 C01 号课程的学生平均成绩。其命令为：

```
SELECT   AVG(Grade)
FROM    Reports
WHERE   Cno='C01';
```

例 1.34 查询选修 C01 号课程的学生最高分数。其命令为：

```
SELECT   MAX(Grade)
FROM    Reports
WHERE   Cno='C01';
```

5 查询结果分组

例 1.35 求各个课程号(Cno)及相应的选课人数。其命令为：

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

例 1.36　查询选修了 3 门或 3 门以上课程的学生学号(Sno)。其命令为：

```
SELECT    Sno
FROM    Reports
GROUP    BY Sno
HAVING    COUNT(Cno)>
```

实验四　连接、嵌套和集合查询(2 课时，综合)

一、实验目的

熟练掌握连接、嵌套和集合查询的使用。

二、实验要求

1．掌握连接、嵌套和集合查询语句的一般格式；

2．掌握连接、嵌套和集合查询的各种使用方法。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

各种连接、嵌套和集合查询方法。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．综合练习如下例题：

1．连接查询

（1）不同表之间的连接查询

例 1.37　查询每个学生及其选修课程的情况。

本查询实际上是涉及 Students 与 Reports 两个表的连接操作。这两个表之间的联系是通过公共属性 Sno 实现的，因此，其操作命令为：

```
SELECT    Students.*, Reports.*
FROM    Students, Reports
WHERE    Students.Sno = Reports.Sno；
```

说明：若在以上等值连接中把目标列中重复的属性列去掉则为自然连接，其命令为

```
SELECT    Students.Sno, Sname, Ssex, Sage, Sdept, Cno, Grade
```

```
FROM    Students, Reports
WHERE   Students.Sno= Reports.Sno;
```

例 1.38 查询每个学生的学号(Sno)、姓名(Sname)、选修的课程名(Cname)及成绩(Grade)。

本查询涉及到三个表的连接操作，完成该查询的 SQL 语句如下：

```
SELECT    Students.Sno, Sname, Cname, Grade
FROM    Students, Reports, Courses
    WHERE    Students.Sno= Reports.Sno AND Reports.Cno=Courses.Cno;
```

（2）自身连接

例 1.39 查`询每一门课的间接先修课(即先修课的先修课)。

在 Courses 表关系中，只有每门课的直接先修课信息，而没有先修课的先修课。要得到这个信息，必须先对一门课找到其先修课，再按此先修课的课程号，查找它的先修课程。这就需要要将 Courses 表与其自身连接。为方便连接运算，这里为 Courses 表取两个别名分别为 A，B。则完成该查询的 SQL 语句为：

```
SELECT    A.Cno, A.Cname, B.Pre_Cno
FROM    Courses    A, Courses    B
WHERE    A.Pre_Cno =B.Cno;
```

（3）外连接

例 1.40 把例 1.37 中的等值连接改为左连接。该左连接操作在 SQL Server 2000 中的命令格式为：

```
SELECT Students.Sno, Sname, Ssex, Sdept, Cno, Grade
FROM Students
LEFT JOIN Reports ON
Students.Sno= Reports.Sno;
```

说明：以上左连接操作也可以用如下的右连接操作代替，其结果完全一样。

```
SELECT Students.Sno, Sname, Ssex, Sdept, Cno, Grade
FROM Reports
RIGHT JOIN Students ON
Reports.Sno=Students.Sno;
```

2．嵌套查询

（1）带谓词 IN 的嵌套查询

例 1.41 查询选修了编号为“C02”的课程的学生姓名(Sname)和所在系(Sdept)。

```
SELECT    Sname, Sdept
FROM    Students
WHERE    Sno    IN
        (SELECT    Sno
        FROM    Reports
```

```
        WHERE    Cno='C02');
```

例 1.42 查询与“李伟”在同一个系学习的学生学号(Sno)、姓名(Sname)和系名(Sdept)。

该查询可构造嵌套查询实现，其 SQL 语句如下：

```
SELECT    Sno, Sname, Sdept
FROM    Students
WHERE    Sdept    IN
        (SELECT    Sdept
         FROM    Students
         WHERE    Sname=’李伟’);
```

说明：本例中的查询也可以用自身连接来完成，其 SQL 语句如下：

```
SELECT    A.Sno , A.Sname , A.Sdept
FROM    Students A , Students B
WHERE    A.Sdept=B.Sdept AND B.Sname=’李伟’;
```

例 1.43 查询选修了课程名为“数据结构”的学生学号(Sno)和姓名(Sname)。

本查询涉及学号、姓名和课程名(Cname)三个属性。学号和姓名存放在 Students 表中，课程名的存放在 Courses 表中，但 Students 与 Courses 两个表之间没有公共属性，必须通过 Reports 表建立它们之间的联系。所以本查询实际上涉及三个关系的连接操作。

```
SELECT    Sno, Sname                          /* ③最后在 Studenst 关系中 */
FROM    Students                               /* 取出 Sno 和 Sname        */
WHERE    Sno    IN
       (SELECT    Sno                         /*② 然后在 SC 关系中找出 */
        FROM    Reports                       /*选修了 3 号课程的学生学号*/
        WHERE    Cno    IN
              (SELECT    Cno                   /*① 首先在 Courses 关系中    */
               FROM    Courses               /*找出“数据结构”的课程号，*/
               WHERE    Cname = ‘数据结构’));    /*结果为 C02 号              */
```

说明：本查询同样可以用连接查询实现：

```
SELECT    S.Sno, Sname
FROM    Students S, Reports R, Courses    C
WHERE    S.Sno=R.Sno AND R.Cno=C.Cno AND C.Cname='数据结构';
```

（2）带有比较运算符的嵌套查询

例 1.44 将例 1.42 改为带有比较运算符的嵌套查询。由于一个学生只可能在一个系学习，因此子查询的结果是一个值，因此可以用=代替 IN，其 SQL 语句如下：

```
SELECT    Sno , Sname, Sdept
FROM    Students
```

```
WHERE    Sdept =
         (SELECT    Sdept
          FROM    Students
          WHERE    Sname='李伟');
```

（3）带谓词 ANY 或 ALL 的嵌套查询

例 1.45 查询非自动化系的不超过自动化系所有学生的年龄的学生姓名(Sname)和年龄(Sage)。其查询命令为

```
SELECT    Sname, Sage
FROM    Students
WHERE Sdept<>'自动化'
        AND Sage<=ALL (SELECT    Sage
                FROM    Students
                WHERE    Sdept= '自动化');
```

说明：本查询也可以用集函数来实现。其 SQL 语句如下：

```
SELECT    Sname, Sage
FROM    Students
WHERE Sdept<>'自动化'
        AND Sage<= (SELECT    MIN(Sage)
                FROM    Students
                WHERE    Sdept='自动化');
```

（4）带谓词 EXISTS 的嵌套查询

例 1.46 查询所有选修了编号为“C01”课程的学生姓名(Sname)和所在系(Sdept)。

本查询的 SQL 语句是：

```
SELECT    Sname, Sdept
FROM    Students
WHERE    EXISTS
         (SELECT    *
          FROM    Reports
           WHERE    Sno=Students.Sno AND Cno='C01');
```

例 1.47 将例 1.42 改为带谓词 EXISTS 的查询，其 SQL 语句如下

```
SELECT    Sno, Sname, Sdept
FROM    Students A
WHERE    EXISTS
         (SELECT *
         FROM    Students B
```

```
        WHERE    B.Sdept=A.Sdept AND B.Sname='李伟');
```

例 1.48 查询选修了所有课程的学生姓名(Sname)和所在系。

由于没有全称量词，可将题目的意思转换成等价的用存在量词的形式：查询这样的学生，没有一门课程是他不选修的。其 SQL 语句为：

```
SELECT    Sname, Sdept
FROM    Students
WHERE    NOT EXISTS
          (SELECT    *
           FROM    Courses
           WHERE    NOT EXISTS
                    (SELECT *    FROM    Reports    WHERE    Sno=Students.Sno
                      AND    Cno=Courses.Cno));
```

3．集合查询

例 1.49 查询计算机科学系的学生或年龄不大于 20 岁的学生信息。

```
SELECT    *
FROM    Students
WHERE    Sdept='计算机'
UNION
SELECT    *
FROM    Students
WHERE    Sage<=20;
```

例 1.50 查询数学系的学生且年龄不大于 20 岁的学生的交集，这实际上就是查询数学系中年龄不大于 20 岁的学生。

```
SELECT    *
FROM    Students
WHERE    Sdept='数学'    AND    Sage<=20;
```

例 1.51 查询数学系的学生与年龄不大于 20 岁的学生的差集。

本查询的等价说法是，查询数学系中年龄大于 20 岁的学生。

```
SELECT    *
FROM    Students
WHERE    Sdept='计算机' AND Sage>20;
```

实验五 SQL 的数据更新(2 课时，验证)

一、实验目的

熟练掌握 sql 数据插入、修改和删除的使用。

二、实验要求

1．掌握 sql 数据插入、修改和删除语句的一般格式；

2．掌握 sql 数据插入、修改和删除使用方法。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

sql 数据的插入、修改和删除。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．验证如下例题：

1 插入数据

例 1.52　设数据库中已有一个关系 History_Student，其关系模式与 Students 完全一样，试将关系 Students 中的所有元组插入到关系 History_Student 中去，其 SQL 命令为：

```
INSERT
INTO　History_Student
SELECT　*
FROM　Students;
```

2 修改数据

例 1.53　将学号为“S03”的学生年龄改为 22 岁，即要修改满足条件的一个元组的属性值。

```
UPDATE　Students
SET Sage=22
WHERE　Sno=’S03’;
```

例 1.54 将所有学生的年龄增加 1 岁。即要修改多个元组的值。

```
UPDATE　Students
SET　Sage=1+Sage;
```

例 1.55　将数学系所有学生的成绩置零。

由于学生所在系的信息在 Students 表中，而学习成绩在 Reports 表中，因此，可以将 SELECT 子查询作为 WHERE 子句的条件表达式。故该更新要求的 SQL 命令为：

```
UPDATE Reports
SET　Grade=0
WHERE　‘数学’=
　　(SELECT Sdept
```

```
        FROM    Students
        WHERE   Students.Sno=Reports.Sno);
```

3 删除数据

例 1.56 删除学号为“S04”的学生选修的课号为“C02”的记录。

```
DELETE
FROM    Reports
WHERE   Sno='S04' AND Cno='C02';
```

例 1.57 删除所有学生的选课记录。

```
DELETE
FROM    Reports;
```

这条 DELETE 语句将删除 Reports 的所有元组，使 Reports 成为空表。

例 1.58 删除数学系所有学生的选课记录。

```
DELETE
FROM    Reports
WHERE   '数学'=
        (SELECT    Sdept
        FROM    Students
WHERE   Students.Sno=Reports.Sno);
```

实验六 SQL 的视图(2 课时，验证)

一、实验目的

理解视图的定义、视图的优点与视图的工作原理。掌握在企业管理器和查询分析器中创建、修改及删除视图。能够熟练掌握利用视图向表中插入、删除和修改数据。

二、实验要求

1．掌握 sql 视图建立、修改和删除;

2．掌握 sql 视图查询。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

sql 视图建立、修改和删除。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．验证如下例题：

1．定义视图

（1）建立视图

例 1.59 建立数学系学生的视图，并要求进行修改和插入操作时仍需保证该视图只有数学系的学生，视图的属性名为 Sno，Sname，Sage，Sdept。

```
CREATE  VIEW  C_Student
        AS
        SELECT  Sno, Sname, Sage, Sdept
        FROM  Students
        WHERE  Sdept='数学'
        WITH CHECK OPTION
```

例 1.60 建立学生的学号(Sno)、姓名(Sname)、选修课程名(Cname)及成绩(Grade)的视图。

本视图由三个基本表的连接操作导出，其 SQL 语句如下：

```
CREATE VIEW  Student_CR
        AS
        SELECT  Students.Sno, Sname, Cname, Grade
        FROM  Students, Reports, Courses
        WHERE  Students.Sno= Reports.Sno AND Reports.Cno=Courses.Cno
```

例 1.61 定义一个反映学生出生年份的视图。

```
CREATE VIEW Student_birth(Sno, Sname, Sbirth)
      AS  SELECT  Sno, Sname, 1996-Sage
      FROM  Students
```

（2）删除视图

例 1.62 删除视图 Student_CR。

```
DROP VIEW Student_CR;
```

2．查询视图

例 1.63 在数学系的学生视图 C_Student 中找出年龄(Sage)小于 20 岁的学生姓名(Sname)和年龄(Sage)。

```
SELECT  Sname, Sage
FROM  C_Student
WHERE  Sage<20;
```

说明：本例转换后的查询语句为：

```
SELECT  Sname, Sage
FROM  Students
WHERE  Sdept='数学' AND Sage<20;
```

例 1.64 在 Student_CR 视图中查询成绩在 85 分以上的学生学号(Sno)、姓名(Sname)和课程名称(Cname)。

```
SELECT  Sno, Sname, Cname
    FROM Student_CR
WHERE  Grade>85;
```

3．更新视图

例 1.65 将数学系学生视图 C_Student 中学号为 S05 的学生姓名改为“黄海”。

```
UPDATE  C_Student
SET  Sname='黄海'
WHERE  Sno='S05';
```

说明：DBMS 自动转换为对基本表的更新语句如下：

```
UPDATE  Students
SET  Sname='黄海'
WHERE Sno='S05' AND Sdept='数学';
```

例 1.66 向数学系学生视图 C_Student 中插入一个新的学生记录，其中学号为“S09”，姓名为“王海”，年龄为 20 岁。

```
INSERT
INTO  C_Student
VALUES ('S09', '王海', 20, ’数学’);
```

例 1.67 删除数学系学生视图 C_Student 中学号为“S09”的记录。

```
DELETE
FROM  C_Student
WHERE  Sno=’S09
```

实验七 数据库安全和完整性(2 课时，设计)

一、实验目的

1．熟悉通过 SQL 对数据进行安全性和完整性控制的方法；

2．熟悉 SQL SERVER 实用数据库的基本操作。

二、实验要求

1．根据实验要求认真填写实验报告，记录所有的实验用例；

2．在实验报告中要给出具体的操作要求和过程，并针对各种情况做出具体的分析和讨论。

三、实验环境

1．windows 操作系统；

2．Sqlserver 数据库管理系统软件。

四、实验内容

1．使用企业管理器建立数据库用户，使用 SQL 对数据进行安全性控制，包括：授权和权力回收。操作完成后看看已授权的用户是否真正具有授予的数据操作的权力了；权力收回操作之后的用户是否确实丧失了收回的数据操作的权力。

2．使用 SQL 对数据进行完整性控制(三类完整性、CHECK 短语、CONSTRAIN 子句)。用实验证实，当操作违反了完整性约束条件时，系统是如何处理的。

五、实验步骤

1．启动 SQL 查询分析器；

2．选择 SQL　SERVER 后，按确认；

3．选择数据库；

4．参照如下实验用例进行设计练习。

实验用例：

1)假设有下面两个关系模式：

职工(职工号，姓名，年龄，职务，工资，部门号)，其中职工号为主码；

部门(部门号，名称，经理名，电话)，其中部门号为主码；

用 SQL 语言定义这两个关系模式，要求在模式中完成以下完整性约束条件的定义：

(1)定义每个模式的主码；(2)定义参照完整性；(3)定义职工年龄不得超过 60 岁。

2)用 SQL 的 GRANT 和 REVOKE 语句(加上视图机制)完成以下授权定义或存取控制功能：

(a)用户王明对两个表有 SELECT 权力；

(b)用户李勇对两个表有 INSERT 和 DELETE 权力；

(c)用户刘星对职工表有 SELECT 权力，对工资字段具有更新权力；

(d)用户周平具有对两个表所有权力(读，插，改，删数据)，并具有给其他用户授权的权力；

(e)用户杨兰具有从每个部门职工中 SELECT 最高工资，最低工资，平均工资的权力，他不能查看每个人的工资。

3)把(a)～(e)的每一种情况，撤销各用户所授予的权力。

附录：创建数据库用户

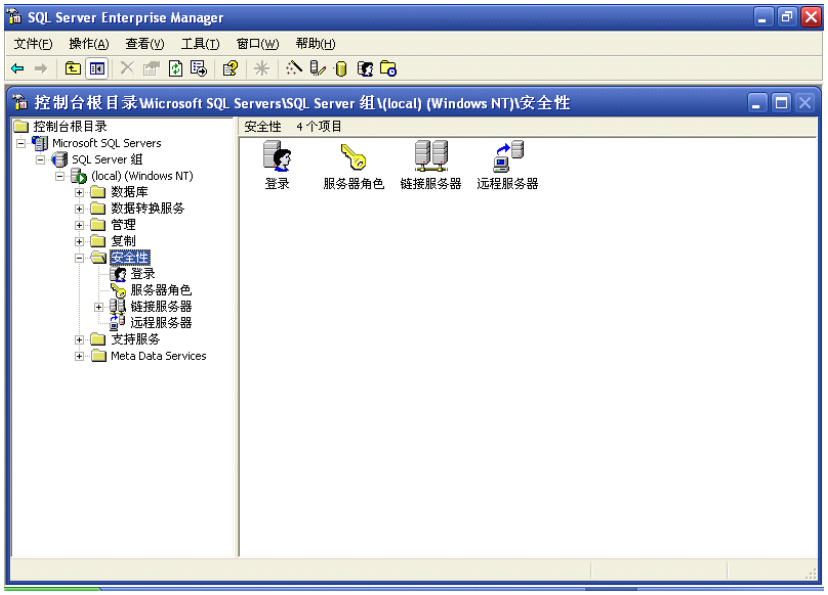
使用企业管理器创建用户步骤：

1.　打开企业管理器

开始|程序|Microsoft SQL Server|企业管理器。

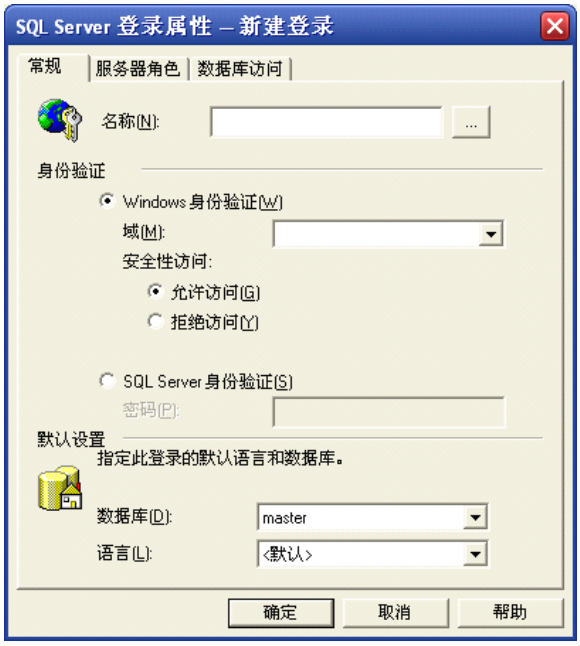
2.　打开安全性子菜单

在企业管理器窗口中，逐层展开实行目录结构，直至安全性目录。



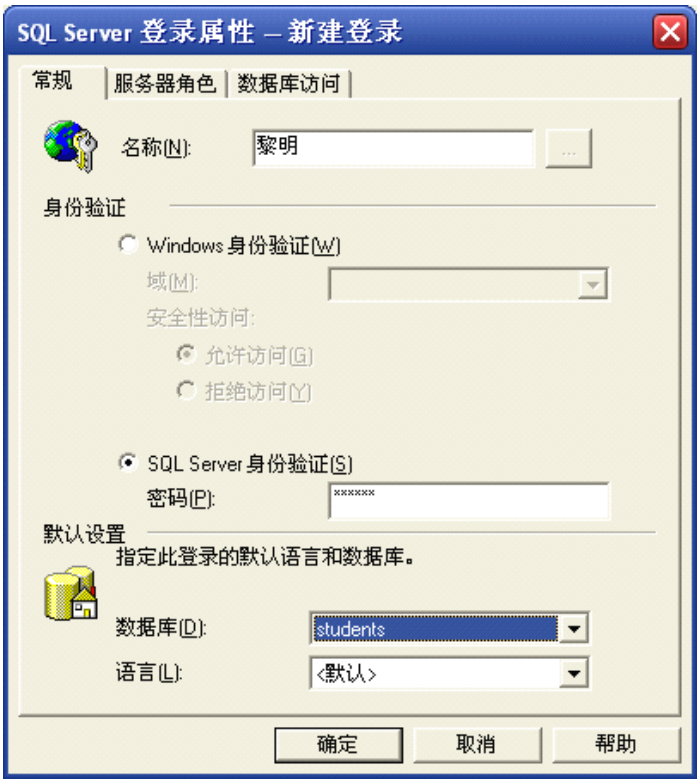
3. 进入新建登录窗口

在安全性目录下，右键单击登录选项，在出现的快捷菜单中，选择新建登录选项，进入新建登录窗口。



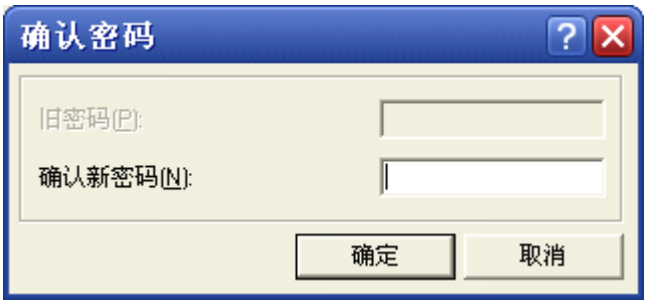
4. 建立新用户

在名称栏中，填入新建用户的名字，如“黎明”；在身份验证栏中，选择 SQL Server 身份验证，并在密码文本框中输入密码；在默认设置栏中，选择该用户登录时进入的数据库，一般选择该用户使用的数据库，如“students”。



在新建登录窗口中，选择数据库访问选项卡，选择登录可以访问的数据库，如“student”，然后选择该用户在数据库中的角色，默认为 public 角色，如下图所示。

选择完成后，单击确定按钮，在出现的密码确认对话框中，再次输入密码并按确定按钮，就建立了新的登录用户。



建立了新用户后，就可以在查询分析器中以该用户登录，进行安全性和完整性的实验。