**数据库实验**

**第一次实验——SQL语言中的单表查询与连接查询**

曹广杰

数据科学与计算机学院

2017/10/13

任课教师：刘玉葆。

关键词：数据库；SQL语言；连接查询；

Contents

[Task1插入单个元组 1](#_Toc495678715)

[Task2插入单个元组 1](#_Toc495678716)

[Task3：update & set 2](#_Toc495678717)

[Task4 update & set 2](#_Toc495678718)

[实验结果： 2](#_Toc495678719)

[Task5 update & 子查询 3](#_Toc495678720)

[实验结果： 3](#_Toc495678721)

[Task6 update & set 3](#_Toc495678722)

[Task7 delete & 子查询 3](#_Toc495678723)

[Task8 delete & reference 4](#_Toc495678724)

[实验结果： 4](#_Toc495678725)

[Task9 delete & 子查询 4](#_Toc495678726)

[Task10 delete 5](#_Toc495678727)

本次实验基于school数据，联系SQL查询语言进行实验。该数据文件由4种关系构成，分别为courses、students、choices 和 teachers；

## Task1插入单个元组

要求：向 STUDENTS表插入编号是“800022222”且姓名是“ WangLan”的元组。

|  |
| --- |
| **insert**  **into** STUDENTS  **values(**'800022222'**,** 'WangLan'**,** **null,** **null)** |

句法：

1. Insert表示在一个table中添加一行，即一个元组。

## Task2插入单个元组

要求：向 TEACHERS表插入元组(“200001000”,“LXL”,“s4zrck@pew,net”,

“3024”)。

|  |
| --- |
| **insert**  **into TEACHERS**  **values('200001000','LXL','s4zrck@pew,net','3024')** |

如果存在信息不知道，但是该信息由并非主键（key）， 那么我们应该非常诚实地表示我们不知道，即使用null补位，以构成一个完整的元组。

## Task3：update & set

要求：将 TEACHERS表中编号为“200010493”的老师工资改为4000。

|  |
| --- |
| **update TEACHERS**  **set salary = 4000**  **where tid = '200010493'** |

句法：update指示将要更改的table，而set标志具体更改的属性。

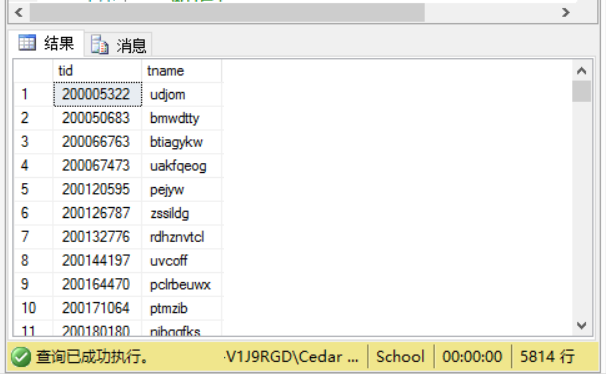
## Task4 update & set

要求：将 TEACHERS表中所有工资小于2500的老师工资改为2500。

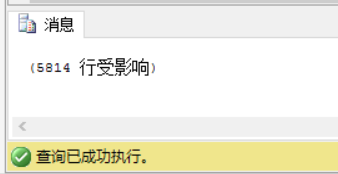
|  |
| --- |
| **update TEACHERS**  **set salary = 2500**  **where salary < 2500** |

### 实验结果：

首先我们查询一下所有工资少于2500元的低收入群体：



可以看到一共有5814lines，此时我们对工资信息进行修改：



这时候再次查询，将不会得到任何结果。

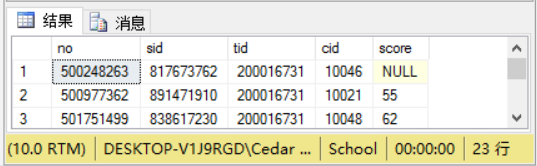
## Task5 update & 子查询

要求：将编号为“200016731”的老师讲授的课程全部改成由姓名为“rnupx”的老师讲授。

|  |
| --- |
| **update CHOICES**  **set tid =**  **(select tid**  **from TEACHERS**  **where tname = 'rnupx')**  **where tid = '200016731'** |

### 实验结果：

首先查询所有200016731老师的授课：



共有23行。而我们要将编号为“200016731”的老师讲授的课程全部改成由姓名为“rnupx”的老师讲授，其实意为：

1. 我们要在“课程”信息里【from/update】；
2. 修改200016731老师的信息【set】；
3. 修改为rnupx的信息【=右值】；

此时查询这位老师的课程信息，查询结果为空。而这也侧面说明了，在子查询语句中的局部变量可以覆盖母查询语句中的同名变量，就像子查询语句中的查询可以继承母查询中的实例化信息。

## Task6 update & set

要求：更新编号“800071780”的学生年级为“2001”。

|  |
| --- |
| **update STUDENTS**  **set grade = '2001'**  **where sid = '800071780'** |

## Task7 delete & 子查询

要求：删除没有学生选修的课程。

|  |
| --- |
| **delete**  **from COURSES**  **where COURSES.cid not in**  **(select cid**  **from CHOICES)** |

Choices中的课程都是有学生选的，因为sid是choices的主键之一。

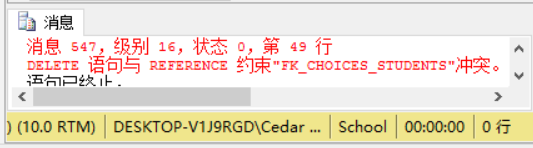
## Task8 delete & reference

要求：删除年级高于1998的学生信息。。

|  |
| --- |
| **delete**  **from CHOICES**  **where CHOICES.sid =**  **any(select sid**  **from STUDENTS**  **where grade < 1998)**  **delete**  **from STUDENTS**  **where grade < 1998** |

### 实验结果：

正常情况下，我们只使用第二个delete语句是不能执行该操作的：



因为我们删除students信息的时候，势必会修改一些sid信息，可是sid是一些choices的主键，不可以成为null——那么类似的，如果它不是主键，在我们删除一些信息之后，可以被设置为null。

## Task9 delete & 子查询

要求：删除没有选修课程的学生信息。

|  |
| --- |
| **delete**  **from STUDENTS**  **where students.sid not in**  **(select sid**  **from CHOICES)** |

其实，本来就不存在没有选课的学生——所有的sid都在choice中。

## Task10 delete

要求：删除成绩不及格的选课记录。

|  |
| --- |
| **delete**  **from CHOICES**  **where score < 60** |

我们知道，choices表示一个选课记录，即便是同一门课程，有同样的授课老师，以及同样的课程编号，对于不同的选课学生，也会有一一对应的选课记录，并包含该同学在修习课程时候的得分。

所以删除的时候，大胆地使用score，对信息进行筛选和限制即可。