**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

2017/9/30

任课教师：刘玉葆。

关键词：数据库；SQL语言；连接查询；

Contents

[Task1 1](#_Toc494575700)

[实验结果： 2](#_Toc494575701)

[Task2 2](#_Toc494575702)

[实验结果 2](#_Toc494575703)

[Task3 2](#_Toc494575704)

[实验结果： 3](#_Toc494575705)

[Task4 3](#_Toc494575706)

[实验结果： 4](#_Toc494575707)

[Task5 4](#_Toc494575708)

[实验结果： 4](#_Toc494575709)

[Task6 4](#_Toc494575710)

[Task7 5](#_Toc494575711)

[实验结果： 5](#_Toc494575712)

[Task8 5](#_Toc494575713)

[实验结果： 6](#_Toc494575714)

[Task9 6](#_Toc494575715)

[实验结果： 6](#_Toc494575716)

[Task10 7](#_Toc494575717)

[实验结果： 7](#_Toc494575718)

[Task11 7](#_Toc494575719)

[实验结果： 7](#_Toc494575720)

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

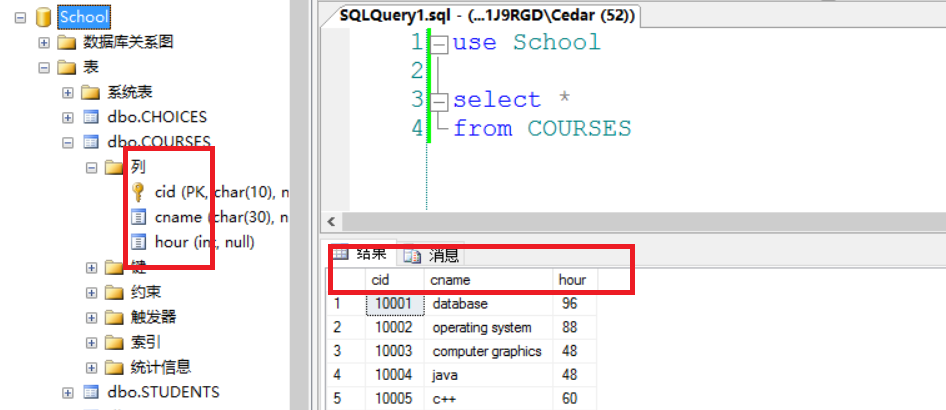
要求：查询全部课程的详细记录。

|  |
| --- |
| **select \***  **from COURSES** |

句法：

1. 结构：select –from –where结构；
2. 语法： \*， 表示详细信息；

### 实验结果：



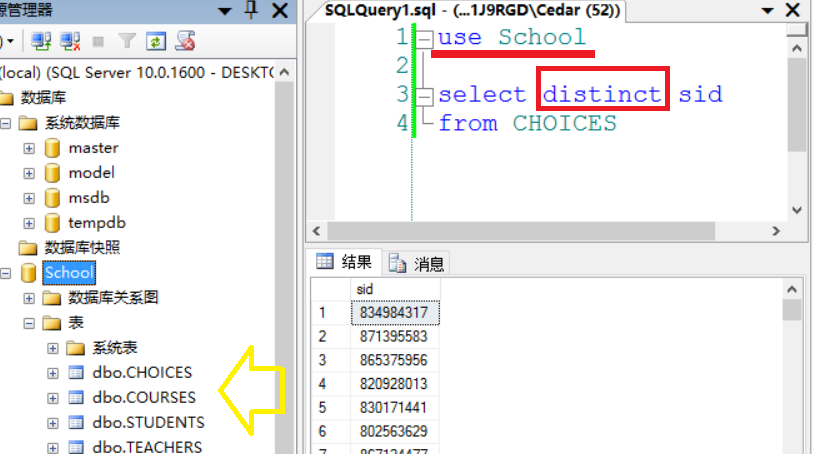
实验运行之后，可以看到左侧出现添加的数据库表，下方同时会出现消息。而查询结果中出现的属性恰好是courses关系中的所有属性，这说明\*的意义正是查询所有属性的详细信息。

## Task2

要求：查询所有有选修课的学生的编号。

|  |
| --- |
| **--设置使用的库，限定查询范围**  **use School**  **--distinct表ID各不相同**  **--choices表示选修课**  **select distinct sid**  **from CHOICES** |

### 实验结果



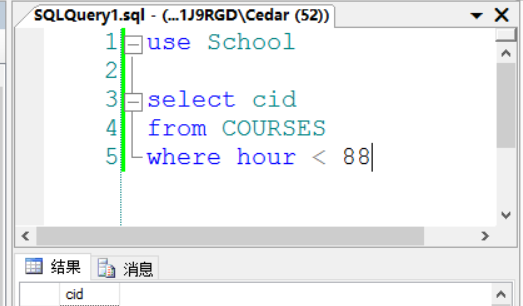
这里主要是测试创建数据库表的语句，左侧相对应地出现了数据库表说明运行地很成功。如果没有出现，请点击红色标注刷新标识刷新界面。

## Task3

要求：查询课时<88(小时)的课程的编号。属于单表查询，直接在choices中查询即可。

|  |
| --- |
| **select cid**  **from COURSES**  **where hour < 88** |

### 实验结果：



此处测试查询coursesID的语句，相对应地出现了添加的属性，说明运行无误。

## Task4

要求：请找出总分超过400分的学生。

|  |
| --- |
| **select sid, SUM(CHOICES.score)**  **from CHOICES**  **group by CHOICES.sid**  **having SUM(CHOICES.score) > 400**  **order by SUM(CHOICES.score) desc** |

思路：

找出总分400分的学生，则说明找出的是学生。问题在于表示总分400的限定条件。

每一个学生可能会选多个课，这是choices数据结构中的信息。所以接下来为了表示课分数的和，我们需要表示：

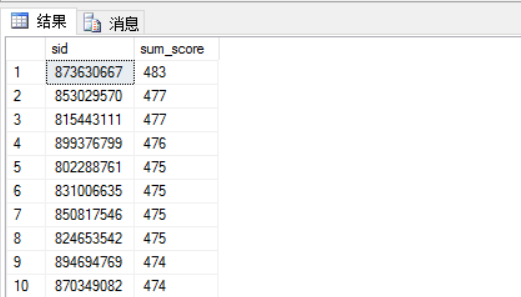
1. 该学生选的课的分数：

在choices中，学生与课的分数是有对应关系的，而我们需要查询对于每一个学生而言的分数总和，因此将所有出现过的分数以学生的ID为标准分为多个类，这时候我们就找到了每一位学生的分数的集合；

1. 所有分数之和：

笔者使用了SUM函数，直接对同一类型的数据做和。

### 实验结果：



可以看到，数据总和都大于400。

## Task5

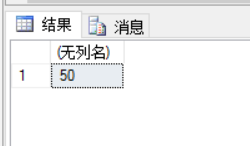
要求：查询课程的总数。

|  |
| --- |
| **select COUNT(\*)**  **from COURSES** |

句法：

Count，表计算。\*，表所有元组。该语句即为计算有多少元组，又因为该table中有主码，因而每一个元组都是可以互相区分的，查询到的即为课程的总数。

### 实验结果：



## Task6

要求：查询所有课程和选修该课程的学生总数。

|  |
| --- |
| **select cid, COUNT(sid)**  **from CHOICES**  **group by cid** |

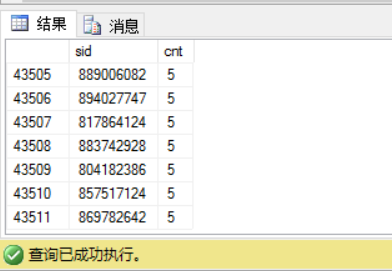
此处测试查询coursesID的语句，既然无新的语法，此处笔者便不予赘述。

## Task7

要求：查询选修成绩超过60的课程超过两门的学生编号。

|  |
| --- |
| **select sid, COUNT(cid) as cnt**  **from CHOICES**  **where score > 60**  **group by sid**  **having 2 < COUNT(cid)**  **order by COUNT(cid)** |

### 实验结果：



查询学生的考核信息：

1. 选修了至少3门课。
2. 这3门课分数都大于60。

表示“至少3门课”时，我们需要对cid进行统计，根据学生的ID分类。使用group by可以实现，这个情况在其他的例子中出现过，不予赘言。

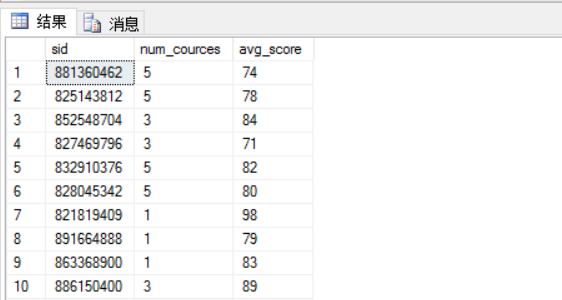
接下来表示“3门课的分数大于60”，我们已经根据学生的ID对课进行了分类查询，其中“只要有3门课分数大于60”等同于“只要大于60的课至少3门即可”，这时候我们就可以不再对所有的课进行统计，而是对分数大于60的课进行统计即可。于是我们添加了where语句进行限定。

## Task8

要求：统计各个学生的选修课程数目和平均成绩。

|  |
| --- |
| **select sid, COUNT(cid) as num\_cources, AVG(score) as avg\_score**  **from CHOICES**  **group by sid** |

### 实验结果：



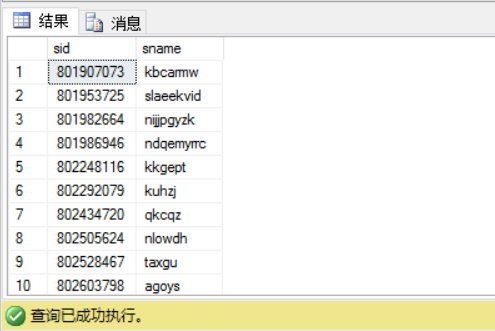
依然是统计属于学生的相关信息，每一个学生的ID可以对应着很多属性，而每一个属性之中又可以对应着很多数据，比如一个人可以选很多种课，这种情况下，对于该ID进行分析时，其他的相关属性需要根据该信息进行分类，此时使用group by。

## Task9

要求：查询选修Java的所有学生的编号及姓名。

|  |
| --- |
| **select STUDENTS.sid, sname**  **from CHOICES, COURSES, STUDENTS**  **where CHOICES.cid = COURSES.cid**  **and COURSES.cname = 'Java'**  **and CHOICES.sid = STUDENTS.sid** |

### 实验结果：



因为课程名字的信息只存在与courses中，而学生姓名又只存在于students中，需要联系二者时，一定要使用choices作为中间连接量，使用外键将三个table关联起来，此时数据对应结构清晰，查询结果也就很简单了。

## Task10

要求：查询姓名为sssht的学生所选的课程的编号和成绩。

|  |
| --- |
| **select CHOICES.cid, CHOICES.score**  **from STUDENTS, CHOICES**  **where sname = 'sssht'**  **and STUDENTS.sid = CHOICES.sid** |

### 实验结果：

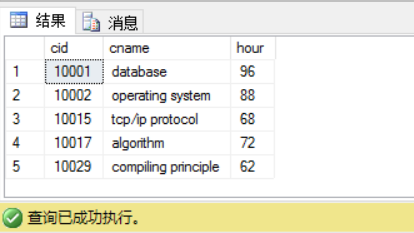


## Task11

要求：查询其他课时比课程C++多的课程的名称。

|  |
| --- |
| **select q.cid, q.cname, q.hour**  **from COURSES as q, COURSES as c**  **where q.hour > c.hour**  **and c.cname = 'c++'** |

### 实验结果：



此处是对于同类事物的查询，因而在实现的过程中，需要对关系实体化，继而进行比较。