

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
实验结果:	2
Task2	2
实验结果	2
Task3	2
实验结果:	3
Task4	3
实验结果:	4
Task5	4
实验结果:	4
Task6	4
Task7	5
实验结果:	5
Task8	5
实验结果:	6
Task9	6
实验结果:	6
Task10	7
实验结果:	7
Task11	7
实验结果:	7

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

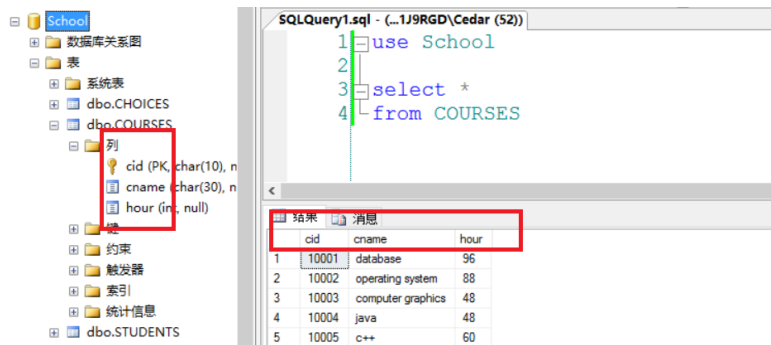
要求: 查询全部课程的详细记录。

```
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
```

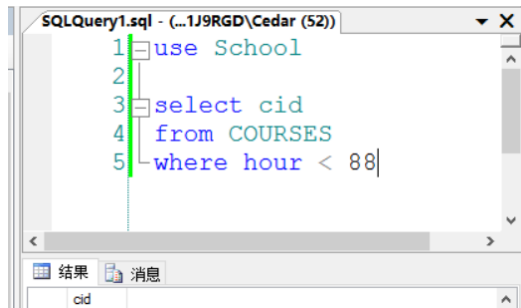
实验结果

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：

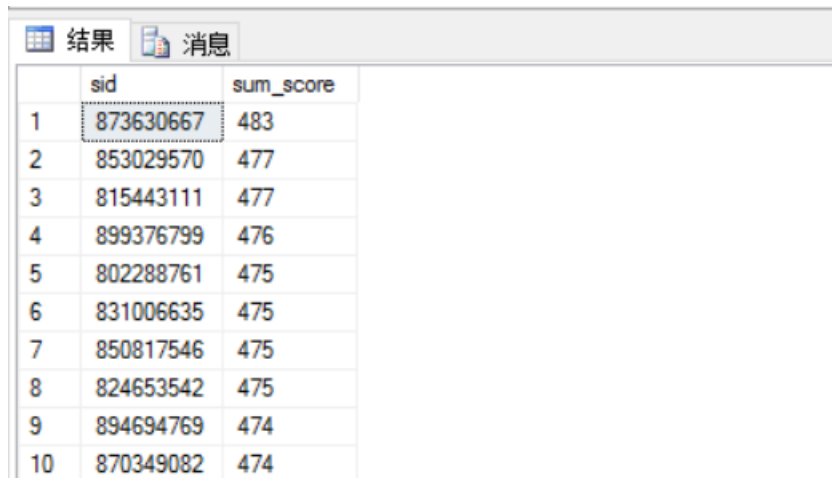
1. 该学生选的课的分：

在 choices 中，学生与课的分是有对应关系的，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；

2. 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：



	sid	sum_score
1	873630667	483
2	853029570	477
3	815443111	477
4	899376799	476
5	802288761	475
6	831006635	475
7	850817546	475
8	824653542	475
9	894694769	474
10	870349082	474

可以看到，数据总和都大于 400。

Task5

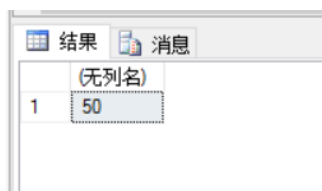
要求：查询课程的总数。

```
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：



(无列名)
1 50

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：

结果	消息	
sid	cnt	
43505	889006082	5
43506	894027747	5
43507	817864124	5
43508	883742928	5
43509	804182386	5
43510	857517124	5
43511	869782642	5

 查询已成功执行。

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_courses, AVG(score) as avg_score
from CHOICES
group by sid
```

实验结果：

	sid	num_courses	avg_score
1	881360462	5	74
2	825143812	5	78
3	852548704	3	84
4	827469796	3	71
5	832910376	5	82
6	828045342	5	80
7	821819409	1	98
8	891664888	1	79
9	863368900	1	83
10	886150400	3	89

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：

	sid	sname
1	801907073	kbcamw
2	801953725	slaekvid
3	801982664	nijpgyzk
4	801986946	ndqemymc
5	802248116	kkgept
6	802292079	kuhzj
7	802434720	qkcqz
8	802505624	nlowdh
9	802528467	taxgu
10	802603798	agoys

✓ 查询已成功执行。

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：

	cid	score
1	10037	84
2	10037	54
3	10019	79
4	10030	53
5	10004	76

查询已成功执行。

Task11

要求：查询其他课时比课程 C++ 多的课程名称。

```
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：

	cid	cname	hour
1	10001	database	96
2	10002	operating system	88
3	10015	tcp/ip protocol	68
4	10017	algorithm	72
5	10029	compiling principle	62

查询已成功执行。

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。