链滴

数据库多表联合查询——笛卡尔积、自然连接、左外连接、右外连接、全外连接等介绍

作者：aopstudio
原文链接：https://ld246.com/article/1583671301562
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

前言

面试的时候被面试官问到关于数据库多表查询的方式，其实以前都是学过的，但我当时只记得笛卡尔（也没记起来这个术语，只知道是逗号分隔表名）和自然连接，有点惭愧。现在再重新整理并记录下。

使用的表例子

来源于Abraham Silberschatz等著的《数据库系统概念（第6版）》中大学数据库的例子，可以参考个链接：https://blog.csdn.net/m0_37961948/article/details/88426110

笛卡尔积

如果不限定任何条件，则表1中的每个元组和表2中的每个元组都会进行组合。这样的组合通常是没有义的，一般会用where子句来限定查询的条件。比如我们想要查询每个教师的姓名和他教授的所有课程D，我们就可以这样写

```
select name,course_id
from instructor,teaches
where instructor.ID=teaches.ID
```

如果不加where子句进行限定，那么查询出来就会有老师对应到其他老师教授的课程的结果。

自然连接

自然连接只考虑在两个表中都出现的属性上取值相同的元组对。回到instructor和teaches关系的例子，instructor和teaches的自然连接计算中只考虑这样的元组对：来自instructor的元组和来自teache的元组在共同属性ID上的取值相同。SQL语句这样写：

```
select name,course_id
from instructor natural join teaches;
```

数据库会自动对两个表的共用属性进行匹配操作。这里要注意的是，如果限定好要查询的属性比如na e和course_id，那么这一种方法是等价于用笛卡尔积的方法的，但如果是查询所有属性，用*的话，者之间会存在区别。使用笛卡尔积会把两个表的共有属性都显示一次，也就是说会有两列重复的ID列而自然连接则会去重，只显示一列ID，如图：

```
mysql> select *
    -> from instructor natural join teaches;
```

ID	name	dept_name	salary	course_id	sec_id	semester	year
76766	Crick	Biology	72000.00	BIO-101	1	Summer	2009
76766	Crick	Biology	72000.00	BIO-301	1	Summer	2010
10101	Srinvasan	Comp. Sci.	65000.00	CS-101	1	Fall	2009
45565	Katz	Comp. Sci.	75000.00	CS-101	1	Spring	2010
83821	Brandt	Comp. Sci.	92000.00	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000.00	CS-190	2	Spring	2009
10101	Srinvasan	Comp. Sci.	65000.00	CS-315	1	Spring	2010
45565	Katz	Comp. Sci.	75000.00	CS-319	1	Spring	2010
83821	Brandt	Comp. Sci.	92000.00	CS-319	2	Spring	2010
10101	Srinvasan	Comp. Sci.	65000.00	CS-347	1	Fall	2009
98345	Kim	Elec. Eng.	80000.00	EE-181	1	Spring	2009
12121	Wu	Finance	90000.00	FIN-201	1	Spring	2010
32343	EI Said	History	60000.00	HIS-351	1	Spring	2010
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010
22222	Einstein	Physics	95000.00	PHY-101	1	Fall	2009

```
15 rows in set (0.00 sec)

mysql> select *
    -> from instructor,teaches
    -> where instructor.ID=teaches.ID;
```

ID	name	dept_name	salary	ID	course_id	sec_id	semester	year
76766	Crick	Biology	72000.00	76766	BIO-101	1	Summer	2009
76766	Crick	Biology	72000.00	76766	BIO-301	1	Summer	2010
10101	Srinvasan	Comp. Sci.	65000.00	10101	CS-101	1	Fall	2009
45565	Katz	Comp. Sci.	75000.00	45565	CS-101	1	Spring	2010
83821	Brandt	Comp. Sci.	92000.00	83821	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000.00	83821	CS-190	2	Spring	2009
10101	Srinvasan	Comp. Sci.	65000.00	10101	CS-315	1	Spring	2010
45565	Katz	Comp. Sci.	75000.00	45565	CS-319	1	Spring	2010
83821	Brandt	Comp. Sci.	92000.00	83821	CS-319	2	Spring	2010
10101	Srinvasan	Comp. Sci.	65000.00	10101	CS-347	1	Fall	2009
98345	Kim	Elec. Eng.	80000.00	98345	EE-181	1	Spring	2009
12121	Wu	Finance	90000.00	12121	FIN-201	1	Spring	2010
32343	EI Said	History	60000.00	32343	HIS-351	1	Spring	2010
15151	Mozart	Music	40000.00	15151	MU-199	1	Spring	2010
22222	Einstein	Physics	95000.00	22222	PHY-101	1	Fall	2009

自然连接运算的结果是关系。从概念上讲，from子句种的instructor natural join teaches表达式可替换成执行该自然连接后所得到的关系，相当于一张虚表，然后在这张虚表上执行select语句。

在一个SQL查询的from子句中，可以用自然连接将多个关系结合在一起，如下所示：

```
select A1,A2,...,An
from r1 natural join r2 natural join ... natural join rm
where P;
```

更为一般地说，from子句可以为如下形式：

```
from E1,E2,...,En
```

其中Ei可以是单个关系，也可以是一个包含自然连接的表达式。假设我们要查询老师的名字和他们教

课程的名称，用SQL语句可以这样写：

```
select name,title
from instructor natural join teaches, course
where teaches.course_id=course.course_id;
```

where子句中的teaches.course_id指的是自然连接后的course_id，之所以要写teaches是因为它最来自于teaches表。

但是下面的SQL语句不会计算出相同的结果：

```
select name,title
from instructor natural join teaches natural join course;
```

注意instructor和teaches的自然连接包含属性(ID,name,dept_name,salary,course_id,sec_id)，而course表包含的属性是(course_id,title,dept_name,credits)。作为者二者自然连接的结果，需要元组既在dept_name上相等，又要在course_id上取值相同。问题在于course表中的dept_name指的是课程为哪个系开设的，前面的dept_name指的是老师所在的系，而老师可以在其他系开设课程，就比如我大学里面上的政治课都是马克思主义学院的老师上的，但是这门课是为我们软件学院开设的。因此连两个自然连接会导致查询不到老师在其他系开设的课程。

using

为了避免不必要的相等属性带来的错误，SQL允许用户来指定需要列相等，关键词是using，使用usin时，**直接写join而不是natural join**。我们可以用这样的SQL语句来代替上面的：

```
select name,title
from (instructor natural join course) join course using (course_id);
```

r1 join r2 using(A1,A2)与r1,r2的自然连接类似，但只要r1.A1=r2.A1且r1.A2=r2.A2就能成立，即使r,r2都有名为A3的属性，也不需要t1.A3=t2.A3成立。

on

on的用法和using类似，不过需要自己出表名。使用on时同样是直接写join而不是natural join。区别on中指定的匹配属性会出现两次，就和我们一开始介绍的使用笛卡尔积配合where一样，而using中定的属性只会出现一次，和自然连接一样。on的SQL语句如下：

```
select *
from student join takes student.ID=takes.ID;
```

这个语句的查询结果是学生和各自所修课程的对应关系，它和以下查询是等价的：

```
select *
from student,takes
where student.ID = takes.ID;
```

on条件可以表示任何SQL为此，它可以表示比自然连接更为丰富的连接条件。但上面的例子中on可用where来替换，这样看来，on似乎是一个冗余的SQL特征。

但是引入on有两个优点。首先，对于下面马上要介绍的外连接来说，on条件的表现和where是不同；其次，如果在on子句中指定连接条件，在where子句中指定其他条件，这样的SQL语句更容易让人懂。

外连接

假设我们要查询一个包含所有学生的列表，显示他们的ID、name、dept_name和tot_cred，以及他选修的课程。下面的查询好像检索出了所需的信息：

```
select *
from student natural join takes;
```

但它存在问题。假设有一些学生，他们没有选修任何课程。那么这些学生在student表中对应的元组和akes表中的任何元组配对，都不会满足自然连接的条件。也就是说他们会在查询结果中丢失。

外连接可以防止这种丢失情况，它会保留没有被匹配上的元组。

实际上有三种形式的外连接：

- 左外连接(left outer join)：保留出现在左外连接运算之前（左边）的关系中的元组
- 右外连接(left outer join)：保留出现在右外连接运算之后（右边）的关系中的元组
- 全外连接(full outer join)：对两个关系中的元组都保留

左外连接

要改进刚才的查询可以这样写：

```
select *
from student natural left outer join takes;
```

这样没有选修课程的学生也会出现在查询结果中，对于课程相关属性它的值均为null。可以利用这个质找出所有没有选修课程的学生

```
select *
from student natural left outer join takes
where course_id is null;
```

右外连接

右外连接和左外连接是对称的，上面的查询可以写成

```
select *
from takes natural right outer join student;
```

不过属性的出现顺序不同，前一种是先出现student表属性，后一种是先出现takes表属性

全外连接

全外连接是左外连接与右外连接的组合。在自然连接结果计算出来之后，左侧关系中不匹配右侧关系何元组的元组被添上空值并加入结果中，右侧关系中不匹配左侧关系任何元组的元组也被添上空值并到结果中

我们举一个稍微复杂点的例子来说明全外连接，假设我们要执行如下查询：“显示Comp. Sci.系所有生以及他们在2009年春季选修的的所有课程的列表，一门课都没选的学生和没有人选的课程都要显示来。”我们可以这样写：

```
select *
from(select *
    from student
    where dept_name='Comp. Sci')
natural full outer join
(select *
from takes
where semester='Spring' and year=2009);
```

on子句可以和外连接一起使用。下述查询与student natural left outer join takes的查询是相同的，不过属性ID在结果中出现两次。

```
select *
from student left outer join takes on students.ID=takes.ID;
```

我们在前面提到on和where在外连接中的表现是不同的。原因是外连接只为那些对相应内连接结果没贡献的元组补上空值并加入结果。on条件是外连接声明的一部分，但where子句却不是。假设我们将面采用on的查询换成采用where，并使用on条件true：

```
select *
from student left outer join takes on true
where students.ID=takes.ID;
```

第一种查询中on属于左外连接运算的整体，左外连接的性质决定了它会将没有选修课程课程的学生保下来。第二种语句可以理解为先产生左外连接结果，之后再用where子句去对这个结果中的元组作匹，此时没有选修课程的学生对应的元组中takes.ID为null，不满足students.ID=takes.ID的要求，所会丢弃他们。

连接类型和条件

为了把常规连接和外连接区别开来，SQL中把常规连接也称作内连接。当连接子句使用inner join时明是内连接，不过inner是可选的，直接写join默认就是inner join。也就是说：

```
select *
from student join takes using(ID);
```

等价于

```
select *
from student inner join takes using(ID);
```

类似地，natural inner join等价于natural join