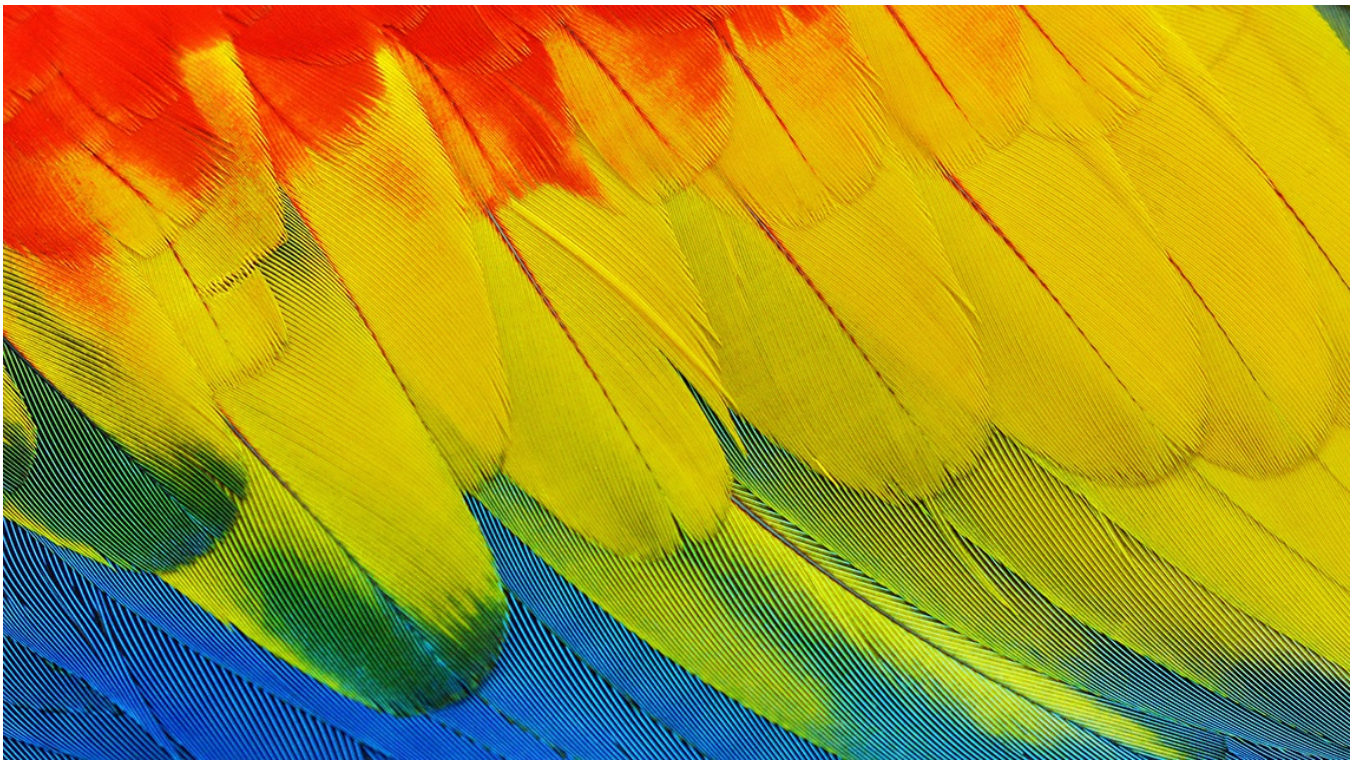
# 10丨常用的SQL标准有哪些，在SQL92中是如何使用连接的？

2019-07-03 陈旸



今天我主要讲解连接表的操作。在讲解之前，我想先给你介绍下连接（JOIN）在SQL中的重要性。

我们知道SQL的英文全称叫做Structured Query Language，它有一个很强大的功能，就是能在各个数据表之间进行连接查询（Query）。这是因为SQL是建立在关系型数据库基础上的一种语言。关系型数据库的典型数据结构就是数据表，这些数据表的组成都是结构化的（Structured）。你可以把关系模型理解成一个二维表格模型，这个二维表格是由行（row）和列（column）组成的。每一个行（row）就是一条数据，每一列（column）就是数据在某一维度的属性。

正是因为在数据库中，表的组成是基于关系模型的，所以一个表就是一个关系。一个数据库中可以包括多个表，也就是存在多种数据之间的关系。而我们之所以能使用SQL语言对各个数据表进行复杂查询，核心就在于连接，它可以用一条SELECT语句在多张表之间进行查询。你也可以理解为，关系型数据库的核心之一就是连接。

既然连接在SQL中这么重要，那么针对今天的内容，需要你从以下几个方面进行掌握：

1. SQL实际上存在不同的标准，不同标准下的连接定义也有不同。你首先需要了解常用的SQL标准有哪些；
2. 了解了SQL的标准之后，我们从SQL92标准入门，来看下连接表的种类有哪些；
3. 针对一个实际的数据库表，如果你想要做数据统计，需要学会使用跨表的连接进行操作。

## 常用的SQL标准有哪些

在正式开始讲连接表的种类时，我们首先需要知道SQL存在不同版本的标准规范，因为不同规范下的表连接操作是有区别的。

SQL有两个主要的标准，分别是SQL92和SQL99。92和99代表了标准提出的时间，SQL92就是92年提出的标准规范。当然除了SQL92和SQL99以外，还存在SQL-86、SQL-89、SQL:2003、SQL:2008、SQL:2011和SQL:2016等其他的标准。

这么多标准，到底该学习哪个呢？实际上最重要的SQL标准就是SQL92和SQL99。一般来说SQL92的形式更简单，但是写的SQL语句会比较长，可读性较差。而SQL99相比于SQL92来说，语法更加复杂，但可读性更强。我们从这两个标准发布的页数也能看出，SQL92的标准有500页，而SQL99标准超过了1000页。实际上你不用担心要学习这么多内容，基本上从SQL99之后，很少有人能掌握所有内容，因为确实太多了。就好比我们使用Windows、Linux和Office的时候，很少有人能掌握全部内容一样。我们只需要掌握一些核心的功能，满足日常工作的需求即可。

## 在SQL92中是如何使用连接的

相比于SQL99，SQL92规则更简单，更适合入门。在这篇文章中，我会先讲SQL92是如何对连接表进行操作的，下一篇文章再讲SQL99，到时候你可以对比下这两者之间有什么区别。

在进行连接之前，我们需要用数据表做举例。这里我创建了NBA球员和球队两张表，SQL文件你可以从GitHub上下载。

其中player表为球员表，一共有37个球员，如下所示：

| player_id | team_id | player_name | height |
|---|---|---|---|
| 10001 | 1001 | 韦恩·艾灵顿 | 1.93 |
| 10002 | 1001 | 雷吉·杰克逊 | 1.91 |
| 10003 | 1001 | 安德烈·德拉蒙德 | 2.11 |
| 10004 | 1001 | 索恩·马克 | 2.16 |
| ..... | ..... | ..... | ..... |
| 10037 | 1002 | 伊凯·阿尼博古 | 2.08 |

team表为球队表，一共有3支球队，如下所示：

| team_id | team_name |
| --- | --- |
| 1001 | 底特律活塞 |
| 1002 | 印第安纳步行者 |
| 1003 | 亚特兰大老鹰 |

有了这两个数据表之后，我们再来看下SQL92中的5种连接方式，它们分别是笛卡尔积、等值连接、非等值连接、外连接（左连接、右连接）和自连接。

**笛卡尔积**

笛卡尔乘积是一个数学运算。假设我有两个集合X和Y，那么X和Y的笛卡尔积就是X和Y的所有可能组合，也就是第一个对象来自于X，第二个对象来自于Y的所有可能。

我们假定player表的数据是集合X，先进行SQL查询：

```
SELECT * FROM player
```

再假定team表的数据为集合Y，同样需要进行SQL查询：

```
SELECT * FROM team
```

你会看到运行结果会显示出上面的两张表格。

接着我们再来看下两张表的笛卡尔积的结果，这是笛卡尔积的调用方式：

```
SQL: SELECT * FROM player, team
```

运行结果（一共37*3=111条记录）：

| player_id | team_id | player_name | height | team_id(1) | team_name |
| --- | --- | --- | --- | --- | --- |
| 10001 | 1001 | 韦恩·艾灵顿 | 1.93 | 1001 | 底特律活塞 |
| 10001 | 1001 | 韦恩·艾灵顿 | 1.93 | 1002 | 印第安纳步行者 |
| 10001 | 1001 | 韦恩·艾灵顿 | 1.93 | 1003 | 亚特兰大老鹰 |
| ..... | ..... | ..... | ..... | ..... | ..... |
| 10037 | 1002 | 伊凯·阿尼博古 | 2.08 | 1003 | 亚特兰大老鹰 |

笛卡尔积也称为交叉连接，英文是CROSS JOIN，它的作用就是可以把任意表进行连接，即使这两张表不相关。但我们通常进行连接还是需要筛选的，因此你需要在连接后面加上WHERE子句，也就是作为过滤条件对连接数据进行筛选。比如后面要讲到的等值连接。

### 等值连接

两张表的等值连接就是用两张表中都存在的列进行连接。我们也可以对多张表进行等值连接。

针对player表和team表都存在team_id这一列，我们可以用等值连接进行查询。

```
SQL: SELECT player_id, player.team_id, player_name, height, team_name FROM player, team WHERE player.te
```

运行结果（一共37条记录）：

| player_id | team_id | player_name | height | team_name |
| --- | --- | --- | --- | --- |
| 10001 | 1001 | 韦恩-艾灵顿 | 1.93 | 底特律活塞 |
| 10002 | 1001 | 雷吉-杰克逊 | 1.91 | 底特律活塞 |
| 10003 | 1001 | 安德烈-德拉蒙德 | 2.11 | 底特律活塞 |
| ..... | ..... | ..... | ..... | ..... |
| 10037 | 1002 | Ike Anigbogu | 2.08 | 印第安纳步行者 |

我们在进行等值连接的时候，可以使用表的别名，这样会让SQL语句更简洁：

```
SELECT player_id, a.team_id, player_name, height, team_name FROM player AS a, team AS b WHERE a.team_
```

需要注意的是，如果我们使用了表的别名，在查询字段中就只能使用别名进行代替，不能使用原有的表名，比如下面的SQL查询就会报错：

```
SELECT player_id, player.team_id, player_name, height, team_name FROM player AS a, team AS b WHERE a.te
```

### 非等值连接

当我们进行多表查询的时候，如果连接多个表的条件是等号时，就是等值连接，其他的运算符连接就是非等值查询。

这里我创建一个身高级别表height_grades，如下所示：

| height_level | height_lowest | height_highest |
|---|---|---|
| A | 2.00 | 2.50 |
| B | 1.90 | 1.99 |
| C | 1.80 | 1.89 |
| D | 1.60 | 1.79 |

我们知道player表中有身高height字段，如果想要知道每个球员的身高的级别，可以采用非等值连接查询。

```
SQL：SELECT p.player_name, p.height, h.height_level
FROM player AS p, height_grades AS h
WHERE p.height BETWEEN h.height_lowest AND h.height_highest
```

运行结果（37条记录）：

| player_name | height | height_level |
|---|---|---|
| 韦恩·艾灵顿 | 1.93 | B |
| 雷吉·杰克逊 | 1.91 | B |
| 安德烈·德拉蒙德 | 2.11 | A |
| …… | …… | …… |
| Ike Anigbogu | 2.08 | A |

**外连接**

除了查询满足条件的记录以外，外连接还可以查询某一方不满足条件的记录。两张表的外连接，会有一张是主表，另一张是从表。如果是多张表的外连接，那么第一张表是主表，即显示全部的行，而第剩下的表则显示对应连接的信息。在SQL92中采用（+）代表从表所在的位置，而且在SQL92中，只有左外连接和右外连接，没有全外连接。

什么是左外连接，什么是右外连接呢？

左外连接，就是指左边的表是主表，需要显示左边表的全部行，而右侧的表是从表，（+）表示

哪个是从表。

```
SQL：SELECT * FROM player, team where player.team_id = team.team_id(+)
```

相当于SQL99中的：

```
SQL：SELECT * FROM player LEFT JOIN team on player.team_id = team.team_id
```

右外连接，指的就是右边的表是主表，需要显示右边表的全部行，而左侧的表是从表。

```
SQL：SELECT * FROM player, team where player.team_id(+) = team.team_id
```

相当于SQL99中的：

```
SQL：SELECT * FROM player RIGHT JOIN team on player.team_id = team.team_id
```

需要注意的是，LEFT JOIN和RIGHT JOIN只存在于SQL99及以后的标准中，在SQL92中不存在，只能用（+）表示。

**自连接**

自连接可以对多个表进行操作，也可以对同一个表进行操作。也就是说查询条件使用了当前表的字段。

比如我们想要查看比布雷克·格里芬高的球员都有谁，以及他们的对应身高：

```
SQL：SELECT b.player_name, b.height FROM player as a , player as b WHERE a.player_name = '布雷克-格里芬
```

运行结果（6条记录）：

| player_name | height |
| --- | --- |
| 安德烈-德拉蒙德 | 2.11 |
| 索恩-马克 | 2.16 |
| 扎扎-帕楚里亚 | 2.11 |
| 亨利-埃伦森 | 2.11 |
| 多曼塔斯-萨博尼斯 | 2.11 |
| 迈尔斯-特纳 | 2.11 |

如果不用自连接的话，需要采用两次SQL查询。首先需要查询布雷克·格里芬的身高。

```
SQL：SELECT height FROM player WHERE player_name = '布雷克-格里芬'
```

运行结果为2.08。

然后再查询比2.08高的球员都有谁，以及他们的对应身高：

```
SQL：SELECT player_name, height FROM player WHERE height > 2.08
```

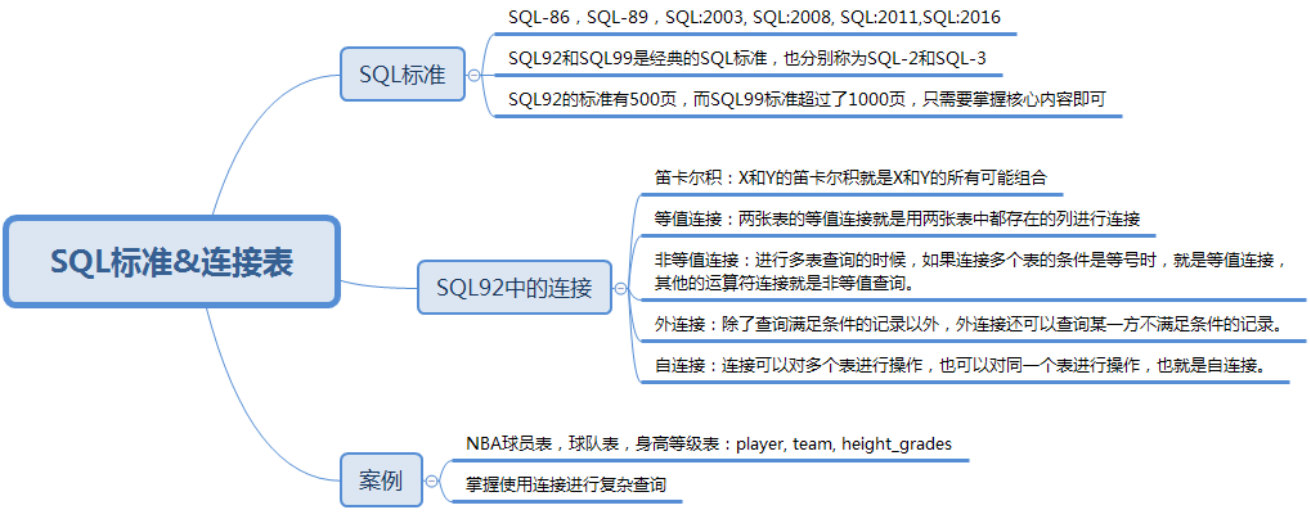
运行结果和采用自连接的运行结果是一致的。

## 总结

今天我讲解了常用的SQL标准以及SQL92中的连接操作。SQL92和SQL99是经典的SQL标准，也分别叫做SQL-2和SQL-3标准。也正是在这两个标准发布之后，SQL影响力越来越大，甚至超越了数据库领域。现如今SQL已经不仅仅是数据库领域的主流语言，还是信息领域中信息处理的主流语言。在图形检索、图像检索以及语音检索中都能看到SQL语言的使用。

除此以外，我们使用的主流RDBMS，比如MySQL、Oracle、SQL Sever、DB2、PostgreSQL等都支持SQL语言，也就是说它们的使用符合大部分SQL标准，但很难完全符合，因为这些数据库管理系统都在SQL语言的基础上，根据自身产品的特点进行了扩充。即使这样，SQL语言也是目前所有语言中半衰期最长的，在1992年，Windows3.1发布，SQL92标准也同时发布，如今我们早已不使用Windows3.1操作系统，而SQL92标准却一直持续至今。

当然我们也要注意到SQL标准的变化，以及不同数据库管理系统使用时的差别，比如Oracle对SQL92支持较好，而MySQL则不支持SQL92的外连接。




我今天讲解了SQL的连接操作，你能说说内连接、外连接和自连接指的是什么吗？另外，你不妨拿案例中的team表做一道动手题，表格中一共有3支球队，现在这3支球队需要进行比赛，请用一条SQL语句显示出所有可能的比赛组合。

欢迎你在评论区写下你的答案，也欢迎把这篇文章分享给你的朋友或者同事，与他们一起交流一下。



精选留言

/*

team 表做一道动手题，表格中一共有 3 支球队，现在这 3 支球队需要进行比赛，请用一条 SQL 语句显示出所有可能的比赛组合。

*/

#分主客队

SELECT CONCAT(kedui.team_name, ' VS ', zhudui.team_name) as '客队 VS 主队' FROM team as zhudui LEFT JOIN team as kedui on zhudui.team_id<>kedui.team_id;

客队 VS 主队

---

底特律活塞 VS 印第安纳步行者
底特律活塞 VS 亚特兰大老鹰
印第安纳步行者 VS 底特律活塞
印第安纳步行者 VS 亚特兰大老鹰
亚特兰大老鹰 VS 底特律活塞
亚特兰大老鹰 VS 印第安纳步行者

#不分主客队

SELECT a.team_name as '队伍1' ,'VS' , b.team_name as '队伍2' FROM team as a ,team as b where a.team_id<b.team_id;

队伍1 VS 队伍2

---

底特律活塞 VS 印第安纳步行者
底特律活塞 VS 亚特兰大老鹰
印第安纳步行者 VS 亚特兰大老鹰

2019-07-03

作者回复

可以看下这个留言，解释的很详细

2019-07-04

圆子蛋

2

三队对阵的可能组合：

SELECT * FROM team AS a,team AS b WHERE a.team_id < b.team_id

主客场对阵的可能（只列出名字的话是不是可以这样？）

SELECT a.team_name as 主场,b.team_name as 客场 FROM team AS a,team AS b WHERE a.team_id != b.team_id

2019-07-03

作者回复

这两个SQL都正确

2019-07-04

Samson

1

SQL：SELECT p.player_name, p.height, h.height_level
FROM player AS p, height_grades AS h
WHERE p.height BETWEEN h.height_lowest AND h.height_highest

老师，我还是不能够理解这条语句中WHERE之后的部分，可以麻烦详加解释一番吗？

另外，对于外连接的两个例子，可以把已经结果也贴一下吗？感觉这样子下效果会更好

2019-07-05

太乙鲲

1

select * from team as a, team as b where a.team_id != b.team_id;

```
+---------+--------------------+---------+--------------------+
| team_id | team_name | team_id | team_name |
+---------+--------------------+---------+--------------------+
| 1002 | 印第安纳步行者 | 1001 | 底特律活塞 |
| 1003 | 亚特兰大老鹰 | 1001 | 底特律活塞 |
| 1001 | 底特律活塞 | 1002 | 印第安纳步行者 |
| 1003 | 亚特兰大老鹰 | 1002 | 印第安纳步行者 |
| 1001 | 底特律活塞 | 1003 | 亚特兰大老鹰 |
| 1002 | 印第安纳步行者 | 1003 | 亚特兰大老鹰 |
+---------+--------------------+---------+--------------------+
```

2019-07-04

作者回复

主队，客队的话 是这样的

2019-07-04

长安落雪

1

SELECT t1.team_name,t2.team_name FROM team as t1 LEFT JOIN team as t2 ON t1.team_id != t2.team_id

SELECT t1.team_name , t2.team_name FROM team as t1 ,team as t2 where t1.team_id<t2.team_id;

2019-07-04

作者回复

正确，大家可以参考下这个。如果是分主客场的话，是第一个SQL。如果是不重复的两个球队的比赛的话，是第二个SQL。

2019-07-04

野马

1

那一个RDBMS支持多个SQL标准吗？

2019-07-04

作者回复

这是一个好问题，有的时候是同时支持的，比如：

SELECT * FROM a, b 和 SELECT * FROM a JOIN b

但有的时候又不支持，比如在MySQL中不支持SQL92标准下的 +号方式的外连接，但是支持SQL99的外连接方式。

具体还需要参考RDBMS相关的文档，即使是同一个标准，每个RDBMS都有自己的“方言”，使用的语法也有差异。

2019-07-04

墨禾 1

```
/*等值连接:两张表存在相同的列属性*/
SELECT player_id, player.team_id, player_name, height, team_name FROM player, team WHERE player.team_id = team.team_id;

/*非等值连接*/
SELECT p.player_name, p.height...
```



```
/*外连接：包括左连接、右连接、全连接*/
-- 左外连接：左边的表为主表
select count(*) from team t left outer join player p on t.team_id = p.team_id;
/*
1001 底特律活塞 10001 1001 韦恩-艾灵顿 1.93
1001 底特律活塞 10002 1001 雷吉-杰克逊 1.91
1002 印第安纳步行者 10037 1002 Ike Anigbogu 2.08
1003 亚特兰大老鹰
*/
-- 右外连接：右边的表为主表
select count(*) from team t RIGHT outer join player p on t.team_id = p.team_id;
/*
1001 底特律活塞 10001 1001 韦恩-艾灵顿 1.93
1001 底特律活塞 10002 1001 雷吉-杰克逊 1.91
1002 印第安纳步行者 10037 1002 Ike Anigbogu 2.08
1003 亚特兰大老鹰
*/

-- 全连接：两张表做笛卡尔积
select count(*) from team t outer join player p on t.team_id = p.team_id;
/*
1001 底特律活塞 10001 1001 韦恩-艾灵顿 1.93
1001 底特律活塞 10002 1001 雷吉-杰克逊 1.91
1002 印第安纳步行者 10037 1002 Ike Anigbogu 2.08
```

1003 亚特兰大老鹰
*/

select count(*) from team t inner join player p on t.team_id = p.team_id;

-- 自连接：可对单表或多表进行操作
SELECT b.player_name, b.height FROM player as a , player as b WHERE a.player_name = '布雷克 - 格里芬' and a.height < b.height

/*请用一条 SQL 语句显示出所有可能的比赛组合*/
SELECT * FROM team t1 , team t2 WHERE t1.team_id<> t2.team_id
-- 或
SELECT * FROM team t1 , team t2 WHERE t1.team_id !=t2.team_id

2019-07-03

ack 1

练习：
1.①内连接也叫连接，是最早的一种连接。还可以被称为普通连接或者自然连接。自然连接是一种特殊的等值连接，他要求两个关系表中进行比较的必须是相同的属性列，无须添加连接条件，并且在结果中消除重复的属性列。要求是两个这两个关系中参与比较的属性列必须是同名、同属性。
②外连接有三种方式：左连接，右连接和全连接。
③自连接，连接的两个表都是同一个表
2.SELECT * FROM team a,team b WHERE a.team_id < b.team_id;

2019-07-03

Geek_669849 0

SELECT * FROM team t1, team t2 WHERE t1.team_id <> t2.team_id ORDER BY t1.team_id ;

2019-07-07

作者回复

针对主队VS客队的形式 是正确的

2019-07-08

Elliot 0

我觉得这一篇应该放在前边，对于入门者比较友好

2019-07-07

一叶知秋 0

只想到了分主客场的。。
SELECT t1.team_name as '主场', t2.team_name as '客场' FROM team as t1, team as t2 WHERE t1.team_id != t2.team_id;

执行结果：

```
+--------------------+--------------------+
| 主场 | 客场 |
+--------------------+--------------------+
| 印第安纳步行者 | 底特律活塞 |
| 亚特兰大老鹰 | 底特律活塞 |
| 底特律活塞 | 印第安纳步行者 |
| 亚特兰大老鹰 | 印第安纳步行者 |
| 底特律活塞 | 亚特兰大老鹰 |
| 印第安纳步行者 | 亚特兰大老鹰 |
+--------------------+--------------------+
6 rows in set (0.00 sec)
```

不分主客场的脑抽了下 没想出来一开始想的distinct...尴尬 看过评论区答案 就不贴了

2019-07-05

一步 0

当我们进行多表查询的时候，如果连接多个表的条件是等号时，就是等值连接，其他的运算符连接就是非等值查询。

---

对于这句话，如果我使用 != 算等值连接吗？ <> 这个应该算非等值查询？

2019-07-05

作者回复

!=和<>作用是一样的（具体看RDBMS是否支持，比如 Access 不支持!=，而会使用<>），都属于非等值查询。

等值查询就是=，也就是帮我们做了等值的关联。比如自然连接就是在相同列名上进行了等值连接，同时保留了所有不重复的列。

2019-07-05

一步 0

有两个问题：

1: 在进行连接查询的时候，查询的顺序是什么呢？ 是先进行笛卡尔积在进行条件条件筛选吗？

2: 在进行连接查询的时候 on 中的条件和 where 中的条件有什么区别呢？ 这两个的筛选顺序一样吗？

2019-07-05

作者回复

1、查询顺序是：FROM > WHERE > GROUP BY > HAVING > SELECT 的字段 > DISTINCT > ORDER BY > LIMIT

可以看下05篇后面的内容。

你说的正确，是先进行 CROSS JOIN 求笛卡尔积，然后进行条件筛选。

2、执行的顺序会先进行ON连接，然后进行WHERE筛选。ON连接是一般连接表的方式，当我们得到数据之后，再会对数据行进行条件筛选

2019-07-05

Yt

0

SELECT CONCAT(a.team_name,'-',b.team_name) AS against
-> FROM team AS a,team AS b
-> WHERE a.team_id !=b.team_id;
+---------------------------+
| against |
+---------------------------+
| 印第安纳步行者-底特律活塞 |
| 亚特兰大老鹰-底特律活塞 |
| 底特律活塞-印第安纳步行者 |
| 亚特兰大老鹰-印第安纳步行者 |
| 底特律活塞-亚特兰大老鹰 |
| 印第安纳步行者-亚特兰大老鹰 |
+---------------------------+
6 rows in set (0.00 sec)

2019-07-04

不是是不是

0

极客时间不可以对文章内容进行复制, 如果使用复制代码功能由会产生SQL:导致无法执行, SQL : SELECT player_id, player.team_id, player_name, height, team_name FROM player, team WHERE player.team_id = team.team_id

2019-07-04

星星

0

select * from team as a, team as b
where a.team_id<>b.team_id;

2019-07-04

华夏

0

SELECT a.team_name AS '主队', b.team_name AS '客队' FROM team AS a, te AS b WHERE a.team_name != b.team_name;
+--------------------+--------------------+
| 主队 | 客队 |
+--------------------+--------------------+
| 印第安纳步行者 | 底特律活塞 |
| 亚特兰大老鹰 | 底特律活塞 |
| 底特律活塞 | 印第安纳步行者 |
| 亚特兰大老鹰 | 印第安纳步行者 |
| 底特律活塞 | 亚特兰大老鹰 |
| 印第安纳步行者 | 亚特兰大老鹰 |
+--------------------+--------------------+
6 rows in set (0.00 sec)

2019-07-03

作者回复

按照主队，客队的方式是这个SQL语句

2019-07-04



Fred 0

SELECT a.team_name AS 主场, b.team_name AS 客场 FROM team AS a, team AS b WHERE a.team_id！=b.team_id

SELECT * FROM team AS a, team AS b WHERE a.team_id<b.team_id

2019-07-03

空知 0

等值连接 的样例SQL写错了

SELECT player_id, player.team_id, player_name, height, team_name FROM player AS a, team AS b a, team as b WHERE a.team_id = b.team_id

起别名地方

2019-07-03

Destroy、 0

select a.team_name, b.team_name from team as a, team as b where a.team_name != b.team_name;

2019-07-03

作者回复

主队和客队的话 可以这么写

2019-07-04