

## 09 | 子查询：子查询的种类都有哪些，如何提高子查询的性能？

2019-07-01 陈旻



上节课我讲到了聚集函数，以及如何对数据进行分组统计，可以说我们之前讲的内容都是围绕单个表的**SELECT**查询展开的，实际上**SQL**还允许我们进行子查询，也就是嵌套在查询中的查询。这样做的好处是可以让我们进行更复杂的查询，同时更加容易理解查询的过程。因为很多时候，我们无法直接从数据表中得到查询结果，需要从查询结果集中再次进行查询，才能得到想要的结果。这个“查询结果集”就是今天我们要讲的子查询。

通过今天的文章，我希望你可以掌握以下内容：

1. 子查询可以分为关联子查询和非关联子查询。我会举一个**NBA**数据库查询的例子，告诉你什么是关联子查询，什么是非关联子查询；
2. 子查询中有一些关键词，可以方便我们对子查询的结果进行比较。比如存在性检测子查询，也就是**EXIST**子查询，以及集合比较子查询，其中集合比较子查询关键词有**IN**、**SOME**、**ANY**和**ALL**，这些关键词在子查询中的作用是什么；
3. 子查询也可以作为主查询的列，我们如何使用子查询作为计算字段出现在**SELECT**查询中呢？

### 什么是关联子查询，什么是非关联子查询

子查询虽然是一种嵌套查询的形式，不过我们依然可以依据子查询是否执行多次，从而将子查询划分为关联子查询和非关联子查询。

子查询从数据表中查询了数据结果，如果这个数据结果只执行一次，然后这个数据结果作为主查询的条件进行执行，那么这样的子查询叫做非关联子查询。

同样，如果子查询需要执行多次，即采用循环的方式，先从外部查询开始，每次都传入子查询进行查询，然后再将结果反馈给外部，这种嵌套的执行方式就称为关联子查询。

单说概念有点抽象，我们用数据表举例说明一下。这里我创建了NBA球员数据库，SQL文件你可以从[GitHub](#)上下载。

文件中一共包括了5张表，player表为球员表，team为球队表，team\_score为球队比赛表，player\_score为球员比赛成绩表，height\_grades为球员身高对应的等级表。

其中player表，也就是球员表，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

team\_score表为球队比赛成绩表，一共记录了两场比赛的成绩，如下所示：

game_id	h_team_id	v_team_id	h_team_score	v_team_score	game_date
10001	1001	1002	102	111	2019-04-01
10002	1002	1003	135	134	2019-04-10

player\_score表为球员比赛成绩表，记录了一场比赛中球员的表现。这张表一共包括19个字段，代表的含义如下：

game_id 比赛ID	player_id 球员ID	is_first 是否首发
playing_time 出场时间	rebound 篮板球	rebound_o 前场篮板
rebound_d 后场篮板	assist 助攻	score 比分
steal 抢断	blockshot 盖帽	fault 失误
foul 犯规	shoot_attempts 总出手	shoot_hits 命中
shoot_3_attempts 3分出手	shoot_3_hits 3分命中	shoot_p_attempts 罚球出手
shoot_p_hits 罚球命中		

其中shoot\_attempts代表总出手的次数，它等于二分之一出手和三分球出手次数的总和。比如2019年4月1日，韦恩·艾灵顿在底特律活塞和印第安纳步行者的比赛中，总出手次数为19，总命中10，三分球13投4中，罚球4罚2中，因此总分 $score=(10-4)\times2+4\times3+2=26$ ，也就是二分之一得分12+三分球得分12+罚球得分2=26。

需要说明的是，通常在工作中，数据表的字段比较多，一开始创建的时候会知道每个字段的定义，过了一段时间再回过头来看，对当初的定义就不那么确定了，容易混淆字段，解决这一问题最好的方式就是做个说明文档，用实例举例。

比如shoot\_attempts是总出手次数（这里的总出手次数=二分之一出手次数+三分球出手次数，不包括罚球的次数），用上面提到的韦恩·艾灵顿的例子做补充说明，再回过头来看这张表的时候，就可以很容易理解每个字段的定义了。

我们以NBA球员数据表为例，假设我们想要知道哪个球员的身高最高，最高身高是多少，就可以采用子查询的方式：

```
SQL: SELECT player_name, height FROM player WHERE height = (SELECT max(height) FROM player)
```

运行结果：（1条记录）

player_name	height
索恩·马克	2.16

你能看到，通过SELECT max(height) FROM player可以得到最高身高这个数值，结果为2.16，然后我们再通过player这个表，看谁具有这个身高，再进行输出，这样的子查询就是非关联子查询。

如果子查询的执行依赖于外部查询，通常情况下都是因为子查询中的表用到了外部的表，并进行了条件关联，因此每执行一次外部查询，子查询都要重新计算一次，这样的子查询就称之为关联子查询。比如我们想要查找每个球队中大于平均身高的球员有哪些，并显示他们的球员姓名、身高以及所在球队ID。

首先我们需要统计球队的平均身高，即SELECT avg(height) FROM player AS b WHERE a.team\_id = b.team\_id，然后筛选身高大于这个数值的球员姓名、身高和球队ID，即：

```
SELECT player_name, height, team_id FROM player AS a WHERE height > (SELECT avg(height) FROM player AS b WHERE a.team_id = b.team_id)
```

运行结果：（18条记录）

player_name	height	team_id
安德烈-德拉蒙德	2.11	1001
索恩-马克	2.16	1001
扎扎-帕楚里亚	2.11	1001
.....	.....	.....
伊凯·阿尼博古	2.08	1002

EXIST子查询

关联子查询通常也会和EXIST一起来使用，EXIST子查询用来判断条件是否满足，满足的话为True，不满足为False。

比如我们想要看出场过的球员都有哪些，并且显示他们的姓名、球员ID和球队ID。在这个统计中，是否出场是通过player\_score这张表中的球员出场表现来统计的，如果某个球员在

player\_score中有出场记录则代表他出场过，这里就使用到了EXIST子查询，即EXISTS (SELECT player\_id FROM player\_score WHERE player.player\_id = player\_score.player\_id), 然后将它作为筛选的条件，实际上也是关联子查询，即：

```
SQL: SELECT player_id, team_id, player_name FROM player WHERE EXISTS (SELECT player_id FROM player_score WHERE player.player_id = player_score.player_id)
```

运行结果：（19条记录）

player_id	team_id	player_name
10001	1001	韦恩·艾灵顿
10002	1001	雷吉·杰克逊
10003	1001	安德烈·德拉蒙德
.....	.....	.....
10032	1002	TJ·利夫

同样，NOT EXIST就是不存在的意思想，我们也可以通过NOT EXIST查询不存在于player\_score表中的球员信息，比如主表中的player\_id不在子表player\_score中，判断语句为NOT EXISTS (SELECT player\_id FROM player\_score WHERE player.player\_id = player\_score.player\_id)。整体的SQL语句为：

```
SQL: SELECT player_id, team_id, player_name FROM player WHERE NOT EXISTS (SELECT player_id FROM player_score WHERE player.player_id = player_score.player_id)
```

运行结果：（18条记录）

player_id	team_id	player_name
10011	1001	布雷克·格里芬
10012	1001	雷吉·巴洛克
10013	1001	卢克·肯纳德
.....	.....	.....
10037	1002	伊凯·阿尼博古

### 集合比较子查询

集合比较子查询的作用是与另一个查询结果集进行比较，我们可以在子查询中使用IN、ANY、ALL和SOME操作符，它们的含义和英文意义一样：

IN	判断是否在集合中
ANY	需要与比较操作符一起使用，与子查询返回的任何值做比较
ALL	需要与比较操作符一起使用，与子查询返回的所有值做比较
SOME	实际上是ANY的别名，作用相同，一般常使用ANY

还是通过上面那个例子，假设我们想要看出场过的球员都有哪些，可以采用IN子查询来进行操作：

```
SELECT player_id, team_id, player_name FROM player WHERE player_id in (SELECT player_id FROM player_s
```

你会发现运行结果和上面的是一样的，那么问题来了，既然IN和EXIST都可以得到相同的结果，那么我们该使用IN还是EXIST呢？

我们可以把这个模式抽象为：

```
SELECT * FROM A WHERE cc IN (SELECT cc FROM B)
```

```
SELECT * FROM A WHERE EXIST (SELECT cc FROM B WHERE B.cc=A.cc)
```

实际上在查询过程中，在我们对cc列建立索引的情况下，我们还需要判断表A和表B的大小。在这里例子当中，表A指的是player表，表B指的是player\_score表。如果表A比表B大，那么IN子查

询的效率要比**EXIST**子查询效率高，因为这时**B**表中如果对**cc**列进行了索引，那么**IN**子查询的效率就会比较高。

同样，如果表**A**比表**B**小，那么使用**EXIST**子查询效率会更高，因为我们可以使用到**A**表中对**cc**列的索引，而不用从**B**中进行**cc**列的查询。

了解了**IN**查询后，我们来看下**ANY**和**ALL**子查询。刚才讲到了**ANY**和**ALL**都需要使用比较符，比较符包括了（>）（=）（<）（>=）（<=）和（<>）等。

如果我们想要查询球员表中，比印第安纳步行者（对应的**team\_id**为**1002**）中任何一个球员身高高的球员的信息，并且输出他们的球员**ID**、球员姓名和球员身高，该怎么写呢？首先我们需要找出所有印第安纳步行者队中的球员身高，即**SELECT height FROM player WHERE team\_id = 1002**，然后使用**ANY**子查询即：

```
SQL: SELECT player_id, player_name, height FROM player WHERE height > ANY (SELECT height FROM player WHERE team_id = 1002)
```

运行结果：（35条记录）

player_id	player_name	height
10001	韦恩·艾灵顿	1.93
10002	雷吉·杰克逊	1.91
10003	安德烈·德拉蒙德	2.11
.....	.....	.....
10037	伊凯·阿尼博古	2.08

运行结果为**35**条，你发现有**2**个人的身高是不如印第安纳步行者的所有球员的。

同样，如果我们想要知道比印第安纳步行者（对应的**team\_id**为**1002**）中所有球员身高都高的球员的信息，并且输出球员**ID**、球员姓名和球员身高，该怎么写呢？

```
SQL: SELECT player_id, player_name, height FROM player WHERE height > ALL (SELECT height FROM player WHERE team_id = 1002)
```

运行结果：（1条记录）



player_id	player_name	height
10004	索恩·马克	2.16

我们能看到比印第安纳步行者所有球员都高的球员，在`player`这张表（一共37个球员）中只有索恩·马克。

需要强调的是**ANY**、**ALL**关键字必须与一个比较操作符一起使用。因为如果你不使用比较操作符，就起不到集合比较的作用，那么使用**ANY**和**ALL**就没有任何意义。

## 将子查询作为计算字段

我刚才讲了子查询的几种用法，实际上子查询也可以作为主查询的计算字段。比如我想查询每个球队的球员数，也就是对应`team`这张表，我需要查询相同的`team_id`在`player`这张表中所有的球员数量是多少。

```
SQL: SELECT team_name, (SELECT count(*) FROM player WHERE player.team_id = team.team_id) AS player_
< >
```

运行结果：（3条记录）

team_name	player_num
底特律活塞	20
印第安纳步行者	17
亚特兰大老鹰	0

你能看到，在`player`表中只有底特律活塞和印第安纳步行者的球员数据，所以它们的`player_num`不为0，而亚特兰大老鹰的`player_num`等于0。在查询的时候，我将子查询**SELECT count(\*) FROM player WHERE player.team\_id = team.team\_id**作为了计算字段，通常我们需要给这个计算字段起一个别名，这里我用的是`player_num`，因为子查询的语句比较长，使用别名更容易理解。

## 总结

今天我讲解了子查询的使用，按照子查询执行的次数，我们可以将子查询分成关联子查询和非关联子查询，其中非关联子查询与主查询的执行无关，只需要执行一次即可，而关联子查询，则需要将主查询的字段值传入子查询中进行关联查询。



同时，在子查询中你可能会使用到**EXIST**、**IN**、**ANY**、**ALL**和**SOME**等关键字。在某些情况下使用**EXIST**和**IN**可以得到相同的效果，具体使用哪个执行效率更高，则需要看字段的索引情况以及表**A**和表**B**哪个表更大。同样，**IN**、**ANY**、**ALL**、**SOME**这些关键字是用于集合比较的，**SOME**是**ANY**的别名，当我们使用**ANY**或**ALL**的时候，一定要使用比较操作符。

最后，我讲解了如何使用子查询作为计算字段，把子查询的结果作为主查询的列。

**SQL**中，子查询的使用大大增强了**SELECT**查询的能力，因为很多时候查询需要从结果集中获取数据，或者需要从同一个表中先计算得出一个数据结果，然后与这个数据结果（可能是某个标量，也可能是某个集合）进行比较。



我今天讲解了子查询的使用，其中讲到了**EXIST**和**IN**子查询效率的比较，当查询字段进行了索引时，主表**A**大于从表**B**，使用**IN**子查询效率更高，相反主表**A**小于从表**B**时，使用**EXIST**子查询效率更高，同样，如果使用**NOT IN**子查询和**NOT EXIST**子查询，在什么情况下，哪个效率更高呢？

最后请你使用子查询，编写**SQL**语句，得到场均得分大于**20**的球员。场均得分从**player\_score**表中获取，同时你需要输出球员的**ID**、球员姓名以及所在球队的**ID**信息。

欢迎在评论区写下你的思考，也欢迎点击[请朋友读把这篇文章分享给你的朋友或者同事](#)。

# SQL 必知必会

## 从入门到数据实战

陈旻

清华大学计算机博士



新版升级：点击「👤请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

### 精选留言



看，有只猪

👍 4

IN表是外边和内表进行hash连接，是先执行子查询。

EXISTS是对外表进行循环，然后在内表进行查询。

因此如果外表数据量大，则用IN，如果外表数据量小，也用EXISTS。

IN有一个缺陷是不能判断NULL，因此如果字段存在NULL值，则会出现返回，因为最好使用NOT EXISTS。

2019-07-01



Liam

👍 3

老师您好，关于exist和in的差别，总结和举例的不符吧？主表大于从表不应该是使用in更好吗

2019-07-01



Mr.H

👍 3

最后的总结in和exist写反了吧

2019-07-01



圆子蛋

👍 1

SELECT player\_id, player\_name, team\_id FROM player WHERE player\_id IN (SELECT player\_id FROM player\_score WHERE score > AVG (score) )

2019-07-01



Cary

👍 1

in 和 exist 上面和最后总结的好像不一致

2019-07-01



华夏

👍 0

这道题用AVG(score) 和score 结果一样，因为score只有一个。

2019-07-01



华夏

👍 0

SELECT player\_id, team\_id, player\_name FROM player WHERE player\_id in (SELECT player\_id FROM player\_score WHERE player.player\_id = player\_score.player\_id);

老师，文稿中这句的WHERE player.player\_id = player\_score.player\_id可以不要了哈。

2019-07-01



果子狗

👍 0

查询比印第安纳步行者（对应的 team\_id 为 1002）中所有球员身高都高的球员的信息是否能理解是比步行者队中最高的球员高就可以了？

SELECT player\_id, player\_name, height FROM player WHERE height > (SELECT MAX(height) FROM player WHERE team\_id = 1002)

如果这样理解是可以的话，怎么分析和使用ALL的子查询的效率高低呢

2019-07-01



华夏

👍 0

SELECT player\_id, player\_name, team\_id FROM player AS a WHERE (SELECT score FROM player\_score AS b WHERE a.player\_id = b.player\_id) > 20;

```
+-----+-----+-----+
| player_id | player_name | team_id |
+-----+-----+-----+
| 10001 | 韦恩-艾灵顿 | 1001 |
| 10002 | 雷吉-杰克逊 | 1001 |
+-----+-----+-----+
```

2 rows in set (0.01 sec)

2019-07-01



Andre

👍 0

终于等到了子查询，小本本记下来

2019-07-01



Sam

👍 0

not in是先执行子查询，得到一个结果集，将结果集代入外层谓词条件执行主查询，子查询只需要执行一次；

not exists是先从主查询中取得一条数据，再代入到子查询中，执行一次子查询，判断子查询是否能返回结果，主查询有多少条数据，子查询就要执行多少次。

2019-07-01



Hero

👍 0

```
1. SELECT
player_id,
team_id,
player_name
FROM
player a
WHERE
EXISTS ( SELECT b.player_id FROM player_score b GROUP BY b.player_id HAVING AVG( b.
score ) > 20 and a.player_id = b.player_id);
```

```
2.SELECT
player_id,
team_id,
player_name
FROM
player a
WHERE
EXISTS ( SELECT b.player_id FROM player_score b WHERE a.player_id = b.player_id GROU
P BY b.player_id HAVING AVG( b.score ) > 20);
```

```
3.SELECT
player_id,
team_id,
player_name
FROM
player
WHERE
player_id IN ( SELECT player_id FROM player_score GROUP BY player_id HAVING AVG( sco
re ) > 20 );
```

推荐3，因为子查询只会执行一次。2比1好，因为where会先过滤数据行，然后分组，然后对分组过滤。

2019-07-01



飞机

👍 0

```
SELECT p.player_id, p.player_name, p.team_id FROM nba.player p where (SELECT AVG(s.sc
ore) FROM nba.player_score s where p.player_id = s.player_id) > 20
```

nba是数据库名字

2019-07-01



太乙鯤

👍 0

```
SELECT player_id, player_name, team_id FROM player WHERE player_id IN (SELECT player_
```

```
id FROM player_score WHERE score > 20);
```

```
+-----+-----+-----+
| player_id | player_name | team_id |
+-----+-----+-----+
| 10001 | 韦恩-艾灵顿 | 1001 |
| 10002 | 雷吉-杰克逊 | 1001 |
+-----+-----+-----+
```

2019-07-01



白了少年头

👍 0

NOT EXIST 应该是主表A小于从表B的情况，效率更高吧  
NOT IN不太清楚，感觉还是主表A小于从表B的情况效率高

```
select player_id, player_name, team_id from player where (select avg(score) from player_score
where player.player_id = player_score.player_id) > 20;
```

```
+-----+-----+-----+
| player_id | player_name | team_id |
+-----+-----+-----+
| 10001 | 韦恩-艾灵顿 | 1001 |
| 10002 | 雷吉-杰克逊 | 1001 |
+-----+-----+-----+
```

2 rows in set (0.01 sec)

2019-07-01



孟崔广

👍 0

作业：

```
SELECT
p.player_name,
p.player_id,
p.team_id
FROM
player p
WHERE
p.player_id IN (
SELECT
s.player_id
FROM
player_score s
GROUP BY
s.player_id
HAVING
AVG(s.score) > 20
);
```

2019-07-01



大斌

👍 0

```
select player_id,player_name,team_id,(select score from player_score where player.player_id=
player_score.player_id and score>20) as score from player;
```

为什么设置了条件，输出里面还有很多个NULL。

2019-07-01



mickey

👍 0

```
SELECT player_id,player_name,team_id FROM player WHERE player_id IN (SELECT player_i
d FROM player_score GROUP BY player_id HAVING AVG(score)>20);
```

2019-07-01



cricket1981

👍 0

老师最后的总结IN/Exist用法和正文描述的不一致。IN子查询一般都有长度大小限制的。

```
mysql> SELECT player_id, player_name, team_id FROM player WHERE player_id in (SELECT
player_id FROM player_score GROUP BY player_id HAVING avg(score) > 20);
```

```
+-----+-----+-----+
| player_id | player_name | team_id |
+-----+-----+-----+
| 10001 | 韦恩-艾灵顿 | 1001 |
| 10002 | 雷吉-杰克逊 | 1001 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

2019-07-01



zjajxzg

👍 0

```
SELECT
player_id,
player_name,
team_id
FROM
player
WHERE
player_id IN ( SELECT player_id FROM player_score GROUP BY player_id HAVING AVG( sco
re ) > 20 )
```

2019-07-01