602. Friend Requests II: Who Has Most Friend?

June 9, 2017 | 22.5K views

Average Rating: 3.36 (11 votes)

In social network like Facebook or Twitter, people send friend requests and accept others' requests as well.

Table `request_accepted`

```
+--------------+-------------+------------+
| requester_id | accepter_id | accept_date|
|--------------|-------------|------------|
| 1            | 2           | 2016_06-03 |
| 1            | 3           | 2016-06-08 |
| 2            | 3           | 2016-06-08 |
| 3            | 4           | 2016-06-09 |
+--------------+-------------+------------+
This table holds the data of friend acceptance, while requester_id and accepter_id bot
```

Write a query to find the the people who has most friends and the most friends number under the following rules:

- It is guaranteed there is only 1 people having the most friends.
- The friend request could only been accepted once, which mean there is no multiple records with the same **requester_id** and **accepter_id** value.

For the sample data above, the result is:

```
Result table:
+------+------+
| id   | num  |
|------|------|
| 3    | 3    |
+------+------+
The person with id '3' is a friend of people '1', '2' and '4', so he has 3 friends in
```

Follow-up:
In the real world, multiple people could have the same most number of friends, can you find all these people in this case?

Solution

Approach: Union *requester_id* and *accepter_id* [Accepted]

Algorithm

Being friends is bidirectional, so if one person accepts a request from another person, both of them will have one more friend.

Thus, we can union column *requester_id* and *accepter_id*, and then count the number of the occurrence of each person.

```
select requester_id as ids from request_accepted
union all
select accepter_id from request_accepted;
```

> Note: Here we should use `union all` instead of `union` because `union all` will keep all the records even the 'duplicated' one.

Taking the sample as an example, the output is:

ids
1
1
2
3
2
3
3
4

Then it will be fairly easy to get the 'ids' with most occurrence using the same technique as mentioned in problem 580. Customer Placing the Largest Number of Orders.

MySQL

```
select ids as id, cnt as num
from
(
select ids, count(*) as cnt
   from
   (
        select requester_id as ids from request_accepted
        union all
        select accepter_id from request_accepted
    ) as tbl1
   group by ids
   ) as tbl2
order by cnt desc
limit 1
;
```

Rate this article:

Previous Next

Comments: 28 Sort By

Type comment here... (Markdown is supported)

Preview Post

wangruoying ★ 63 August 3, 2018 11:12 AM
The answer is wrong. Instead of union just one id, we should union the distinct friend pairs (id1 and id2), then count the number of id1. The answer is double counting if I have person 1 sending request to person 2, and person 2 sending request to person 1. If this happens, we should count this as adding 1 friend to person 1. But the answer would count 2 more friends.
This following is my solution
Read More
63 Share Reply
SHOW 10 REPLIES

ziweifan177 ★ 20 January 11, 2019 6:24 AM
I think it is not necessary to write like the example with so many statements, just this:
SELECT ids, COUNT(ids)
FROM(
SELECT request_id AS ids
FROM leetcode.request_accepted
Read More
17 Share Reply

RogerFederer ★ 757 November 25, 2017 12:24 AM
```
SELECT ids as id , COUNT(*) as num
FROM
  (SELECT requester_id as ids FROM request_accepted
  UNION ALL
```
Read More
9 Share Reply

wjch7722 ★ 5 February 10, 2019 2:17 AM
After union all, when I tried to count distinct friend_id, the result turned out wrong. I think it is a problem here.
5 Share Reply

cailingjie ★ 5 January 23, 2019 4:59 AM
This answer neglects the case that A sent B request meanwhile B sent A request. Also by testing the code, I think the database contains this case.
5 Share Reply

mercuryrecord18 ★ 16 July 7, 2018 10:10 AM
select ids as id,sum(number) as num
from
(select requester_id as ids,count(requester_id) as number
from request_accepted
group by requester_id
Read More
3 Share Reply

shanehhh ★ 2 November 16, 2017 8:28 AM
I am a little confused about why we have to keep duplicate record? Consider the following event: (1) id 12 send a request to id 18, then id 18 accept; (2) id 18 send a request to id 12, then id 12 accept. If we use union all, id 12 will have two friend id 18. It looks weird.
2 Share Reply
SHOW 4 REPLIES

lolozo ★ 86 September 11, 2017 5:59 AM
The outer select is redundant, can just do
select id, count(1) as num
from (
select requester_id as id from request_accepted
union all
Read More
2 Share Reply

andalusia ★ 1 November 2, 2018 8:11 PM
select ids as id, count() as num from
(select requester_id as ids from request_accepted
union all
select accepter_id as ids from request_accepted)
as Results
Read More
1 Share Reply

ye42 ★ 1 July 24, 2018 4:39 AM
Need to remove duplicates.

SELECT id, COUNT(*)
FROM (
SELECT requester_id as id
Read More
1 Share Reply
SHOW 1 REPLY