How to select the first/least/max row per group in SQL

Thu, Dec 7, 2006 in Databases

Here are some common SQL problems, all of which have related solutions: how do I find the most recent log entry for each program? How do I find the most popular item from each category? How do I find the top score for each player? In general, these types of "select the extreme from each group" queries can be solved with the same techniques. I'll explain how to do that in this article, including the harder problem of selecting the top N entries, not just the top 1.

This topic is related to numbering rows, which I just wrote about (see my articles about MySQL-specific and generic techniques to assign a number to each row in a group). Therefore I'll use nearly the same table and data as I used in those articles, with the addition of a `price` column:

```
+--------+-----------+-------+
| type   | variety   | price |
+--------+-----------+-------+
| apple  | gala      |  2.79 |
| apple  | fuji      |  0.24 |
| apple  | limbertwig|  2.87 |
| orange | valencia  |  3.59 |
| orange | navel     |  9.36 |
| pear   | bradford  |  6.05 |
| pear   | bartlett  |  2.14 |
| cherry | bing      |  2.55 |
| cherry | chelan    |  6.33 |
+--------+-----------+-------+
```

Selecting the one maximum row from each group

Let's say I want to select the most recent log entry for

each program, or the most recent changes in an audit table, or something of the sort. This question comes up over and over on IRC channels and mailing lists. I'll re-phrase the question in terms of fruits. I want to select the cheapest fruit from each type. Here's the desired result:

```
+--------+----------+-------+
| type   | variety  | price |
+--------+----------+-------+
| apple  | fuji     |  0.24 |
| orange | valencia |  3.59 |
| pear   | bartlett |  2.14 |
| cherry | bing     |  2.55 |
+--------+----------+-------+
```

There are a few common solutions to this problem. All involve two steps: finding the desired value of `price`, and then selecting the rest of the row based on that.

One common solution is a so-called self-join. Step one is to group the fruits by type (apple, cherry etc) and choose the minimum price:

```
select type, min(price) as minprice
from fruits
group by type;
+--------+----------+
| type   | minprice |
+--------+----------+
| apple  |     0.24 |
| cherry |     2.55 |
| orange |     3.59 |
| pear   |     2.14 |
+--------+----------+
```

Step two is to select the rest of the row by joining these results back to the same table. Since the first query is grouped, it needs to be put into a subquery so it can be joined against the non-grouped table:

```
select f.type, f.variety, f.price
from (
   select type, min(price) as minprice
   from fruits group by type
) as x inner join fruits as f on f.type = x.type
and f.price = x.minprice;
+--------+----------+-------+
```

```
| type   | variety  | price |
+--------+----------+-------+
| apple  | fuji     |  0.24 |
| cherry | bing     |  2.55 |
| orange | valencia |  3.59 |
| pear   | bartlett |  2.14 |
+--------+----------+-------+
```

Another common way to do this is with a correlated subquery. This can be much less efficient, depending on how good your system's query optimizer is. You might find it clearer, though.

```
select type, variety, price
from fruits
where price = (select min(price) from fruits as f
where f.type = fruits.type);
+--------+----------+-------+
| type   | variety  | price |
+--------+----------+-------+
| apple  | fuji     |  0.24 |
| orange | valencia |  3.59 |
| pear   | bartlett |  2.14 |
| cherry | bing     |  2.55 |
+--------+----------+-------+
```

Both queries are logically equivalent, though they may not perform the same.

Select the top N rows from each group

This is a slightly harder problem to solve. Finding a single row from each group is easy with SQL's aggregate functions (`MIN()` , `MAX()` , and so on). Finding the first several from each group is not possible with that method because aggregate functions only return a single value. Still, it's possible to do.

Let's say I want to select the two cheapest fruits from each type. Here's a first try:

```
select type, variety, price
from fruits
where price = (select min(price) from fruits as f
where f.type = fruits.type)
   or price = (select min(price) from fruits as f
where f.type = fruits.type
      and price > (select min(price) from fruits
```

```
as f2 where f2.type = fruits.type));
+--------+----------+-------+
| type   | variety  | price |
+--------+----------+-------+
| apple  | gala     |  2.79 |
| apple  | fuji     |  0.24 |
| orange | valencia |  3.59 |
| orange | navel    |  9.36 |
| pear   | bradford |  6.05 |
| pear   | bartlett |  2.14 |
| cherry | bing     |  2.55 |
| cherry | chelan   |  6.33 |
+--------+----------+-------+
```

Yuck! That can be written as a self-join, but it's just as bad (I leave it as an exercise for the reader). This gets worse as you go to higher numbers (top 3, top 4...). There are other ways to phrase the statement, but they all boil down to the same thing, and they're all pretty unwieldy and inefficient.

There's a better way: select the variety from each type where the variety is no more than the second-cheapest of that type.

```
select type, variety, price
from fruits
where (
   select count(*) from fruits as f
   where f.type = fruits.type and f.price <= frui
ts.price
) <= 2;
```

This is elegant, and lets you vary N without rewriting your query (a very good thing!), but it's functionally the same as the previous query. Both are essentially a quadratic algorithm relative to the number of varieties in each type. And again, some query optimizers may not do well with this and make it quadratic with respect to the number of rows in the table overall (especially if no useful index is defined), and the server might get clobbered. Are there better ways? Can it be done with one pass through the data, instead of the many passes required by a correlated subquery? You know it can, or I wouldn't be writing this, now would I?

Use `UNION`

If there's an index on (type, price), and there are many more records to eliminate than to include in each group, a more efficient single-pass method (especially for MySQL, but also for some other RDBMSs) is to break the queries out separately and put a limit on each, then `UNION` them all back together. Here's the syntax you need for MySQL:

```
(select * from fruits where type = 'apple' order
by price limit 2)
union all
(select * from fruits where type = 'orange' order
by price limit 2)
union all
(select * from fruits where type = 'pear' order b
y price limit 2)
union all
(select * from fruits where type = 'cherry' order
by price limit 2)
```

Peter Zaitsev has written in detail about this technique, so I won't go into it too much more here. If it suits your purposes, it can be a very good solution.

One note: use `UNION ALL` , not just `UNION` . It prevents the server sorting the results to eliminate duplicates before returning them. In this case there will be no duplicates, so I'm telling the server to skip that (useless, expensive) step.

Do it with user variables on MySQL

The `UNION` trick is an especially good idea when the results are a small fraction of the rows in the table and there is an index that can be used for sorting the rows. Another linear-time technique, which might be a good option in cases where you are selecting most of the rows from the table anyway, is user variables. This is MySQL-specific. Please refer to my previous post on how to number rows in MySQL for the gory details of why this works:

```
set @num := 0, @type := '';
```

```
select type, variety, price
from (
   select type, variety, price,
      @num := if(@type = type, @num + 1, 1) as ro
w_number,
      @type := type as dummy
  from fruits
  order by type, price
) as x where x.row_number <= 2;
```

This isn't one pass through the table, by the way. The subquery is implemented as a temporary table behind the scenes, so filling it with data is one pass; then selecting every row from it and applying the `WHERE` clause is another. However, twice through is still O(n) with respect to the table size. That's a lot better than correlated subqueries, which are $O(n^2)$ with respect to the group size – even moderate group sizes cause bad performance (say there are five varieties of each fruit. That's on the order of 25 passes through the table, all told).

One-pass technique on MySQL... maybe?

If you want to leave your queries up the the query optimizer's whims, you can try this technique, which builds no temporary tables and makes just one pass through:

```
set @num := 0, @type := '';

select type, variety, price,
      @num := if(@type = type, @num + 1, 1) as ro
w_number,
      @type := type as dummy
from fruits
group by type, price, variety
having row_number <= 2;
```

This theoretically ought to work if MySQL orders by the `GROUP BY` criteria, which it sometimes does for efficiency and to produce the expected results. Does it work? Here's what it returns on MySQL 5.0.27 on Windows:

```
+--------+----------+-------+------------+-------
-+
| type   | variety  | price | row_number | dummy
```

```
|
+--------+----------+-------+------------+-------
-+
| apple  | gala     |  2.79 |          1 | apple
|
| apple  | fuji     |  0.24 |          3 | apple
|
| orange | valencia |  3.59 |          1 | orange
|
| orange | navel    |  9.36 |          3 | orange
|
| pear   | bradford |  6.05 |          1 | pear
|
| pear   | bartlett |  2.14 |          3 | pear
|
| cherry | bing     |  2.55 |          1 | cherry
|
| cherry | chelan   |  6.33 |          3 | cherry
|
+--------+----------+-------+------------+-------
-+
```

Look closely… it's returning rows one and three from each group, and they're not numbered in order of increasing price? Huh? But the `HAVING` clause says the row_number should be no greater than 2! Here's what it returns on version 5.0.24a on Ubuntu:

```
+--------+-----------+-------+------------+-----
---+
| type   | variety   | price | row_number | dumm
y  |
+--------+-----------+-------+------------+-----
---+
| apple  | fuji      |  0.24 |          1 | appl
e  |
| apple  | gala      |  2.79 |          1 | appl
e  |
| apple  | limbertwig |  2.87 |          1 | appl
e  |
| cherry | bing      |  2.55 |          1 | cher
ry |
| cherry | chelan    |  6.33 |          1 | cher
ry |
| orange | valencia  |  3.59 |          1 | oran
ge |
| orange | navel     |  9.36 |          1 | oran
ge |
| pear   | bartlett  |  2.14 |          1 | pear
|
| pear   | bradford  |  6.05 |          1 | pear
|
+--------+-----------+-------+------------+-----
```

```
---+
```

Look, this time everything is numbered 1 and every row is returned. Wonky. This is exactly what the MySQL manual page on user variables warns about.

This technique is pretty much non-deterministic, because it relies on things that you and I don't get to control directly, such as which indexes MySQL decides to use for grouping. However, if you need to use it – and I know there are some folks out there who do, because I've consulted for them – you can still tweak it. We're getting into the realm of really bastardizing SQL, but the results above came from a table without indexes other than the primary key on (`type, variety`). What happens if I add an index MySQL can use for grouping?

```
alter table fruits add key(type, price);
```

Nothing changes, and `EXPLAIN` shows why: the query doesn't use the index I just added. Why? Because the grouping is on three columns, and the index is only on two. In fact, the query is using a temp table and filesort anyway, so this is still not achieving the once-through goal. I can force it to use the index:

```
set @num := 0, @type := '';

select type, variety, price,
      @num := if(@type = type, @num + 1, 1) as ro
w_number,
      @type := type as dummy
from fruits force index(type)
group by type, price, variety
having row_number <= 2;
```

Let's see if that works:

```
+--------+----------+-------+------------+-------
-+
| type   | variety  | price | row_number | dummy
|
+--------+----------+-------+------------+-------
-+
```

```
| apple  | fuji     |  0.24 |          1 | apple
|
| apple  | gala     |  2.79 |          2 | apple
|
| cherry | bing     |  2.55 |          1 | cherry
|
| cherry | chelan   |  6.33 |          2 | cherry
|
| orange | valencia |  3.59 |          1 | orange
|
| orange | navel    |  9.36 |          2 | orange
|
| pear   | bartlett |  2.14 |          1 | pear
|
| pear   | bradford |  6.05 |          2 | pear
|
+--------+----------+-------+------------+-------
-+
```

Ah, now we're cooking! It did what I wanted, without a filesort or temporary table. Another way to do this, by the way, is to take `variety` out of the `GROUP BY` so it uses the index on its own. Because this selects a non-grouped column from a grouped query, this only works if you are running with `ONLY_FULL_GROUP_BY` mode turned off, which I hope you are not doing without good reason.

Other methods

Be sure to check the comments for user-contributed methods. There are some really novel approaches. I always learn so much from your comments… thank you!

Conclusion

Well, that's it. I've shown you several ways of solving the common "get the extreme row from each group" query, and then moved on to how you can get the top N rows from each group in various ways. Then I dove into MySQL-specific techniques which some (including myself, depending on my mood) would regard as mildly foolish to utterly stupid. But if you need the last bit of speed out of your server, you sometimes have to know when to break the rules. And for those who think this is just MySQL foolishness, it's not; I've seen people desperately do these types of things on other platforms too, such as SQL Server. There are hacks and tweaks on every platform, and

people who need to use them.

I hope you've enjoyed and profited from this discussion. If you did, maybe you'd like to subscribe for convenient notification of future articles. Happy coding!

I'm Baron Schwartz, the founder and CEO of VividCortex. I am the author of High Performance MySQL and many open-source tools for performance analysis, monitoring, and system administration. I contribute to various database communities such as Oracle, PostgreSQL, Redis and MongoDB.

Newer | Older

Comments

Perrin Harkins · 9 years ago

I believe you can also solve the "cheapest 2 from each group" problem with a LEFT JOIN:

```
SELECT f.type, f.variety, f.price
FROM fruits f
LEFT JOIN fruits f2 ON (f.variety = f2.variety
  AND f.price <= f2.price)
GROUP BY f.type, f.variety, f.price
HAVING COUNT(*) <= 2
```

This is untested SQL, but hopefully you get the idea.

sow3110 ➦ Perrin Harkins · 5 months ago

Nice post!!

@Perrin: Cheapest should have been :

"f2.price <= f.price" and not "f.price <= f2.price"

Reply · Share ›

Xaprb · 9 years ago

The concept is right, but you got your criterion slightly mixed.

I knew about this one but somehow didn't think to include it. Thanks for the reminder. It is equivalent to the correlated subquery method, but without subqueries of course. (it is still O(n^2) though).

Reply · Share ›

Marc · 9 years ago

Excellent! This taught me how to solve [redacted] was working on.

search this websi

Thanks!

1 Reply · Share ›

Cristiano · 9 years ago

what about:

```
select
type,substr(min(concat(lpad(cast((price*100) as
char),10,'0'),variety)),11),min(price) from
fruits;
```

one pass, zero subqueries, the only drawback is that it can't be used with float type. It can also use a index for the min if you materialize the expression in the min with a trigger and index it.

1 Reply · Share ›

Paul ↗ Cristiano · 2 years ago

Can you please unpack how this works, specifically focusing on the "substr..." part? I am curious to know the motivation and reasoning.

Reply · Share ›

Jacor ↗ Paul · a year ago

I believe it can still be helpful for some

people. The technique above is to concatenate the zero-padded price and variety so it would yield the same result as min(price), then strip the price with substr.

Reply · Share ›

Anatoly ➚ Cristiano · 8 months ago

Super!
And you can use char(128+N) instead of lpad(...) if N is small (between -128..128)

Reply · Share ›

Xaprb · 9 years ago

Cristiano, that's a novel approach, and a great tip, though you forgot the `GROUP BY type` that's necessary to prevent non-deterministic results. For a sparsely grouped table (e.g. thousands of rows per type), adding an extra column and indexing it would probably have diminishing returns, but I can see some situations where it would be a very good idea.

Thanks for writing in; this is an approach I'd never thought about. Now I have another possible way to solve problems!

Reply · Share ›

Isaac · 9 years ago

This was unbelievably helpful

1 · Reply · Share ›

Enrique · 9 years ago

Great job. Thanks a lot for your article, very helpful.

1 · Reply · Share ›

Raj · 9 years ago

Thanks for this superb article.

1 · Reply · Share ›

Haitao · 9 years ago

RE: Select the top N rows from each group

To get better performance for table with large number of rows, using identity() function and temporary table (if your database has this two feature). The following scripts have been tested in SQL Server 2000

SQL Server 2005:

```
create table fruits
(
  type varchar(10),
  variety varchar(10),
  price decimal(18,2)
)

insert into fruits values( 'apple'  , 'gala'
insert into fruits values( 'apple'  , 'fuji'
insert into fruits values( 'apple'  , 'limbertw
```

see more

1 · Reply · Share ›

Xaprb · 9 years ago

Thanks Haitao. For those who don't know, prefixing a table name with # in SQL Server makes it a temporary table.

2 · Reply · Share ›

Haitao · 9 years ago

RE: Median calculation made easy by reusing the 'Select the top N rows from each group' query

Median calculation is one of the nightmares for some database developers (including myself), and it is disappointing that SQL Server 2005 still does not have the function available, **still not**. The calculation I googled so far are all correlation queries which are far too complex and not practical for tables with millions of records, because of the poor performance. But I just realize that the code I posted above just one step away from the median function, haha gotya...

```
create table fruits (type varchar(10), variety

insert into fruits values( 'apple'  , 'gala'
insert into fruits values( 'apple'  , 'fuji'
insert into fruits values( 'apple'  , 'limbertw
```

see more

1 · Reply · Share ›

Pierre Caron · 9 years ago

Hello,

I have a simple SQL question.

Here is the output I obtain when I issue the following command :

```
SELECT code_age_01, nuitees, COUNT(*) FROM stat

+-------------+---------+----------+
| code_age_01 | nuitees | COUNT(*) |
+-------------+---------+----------+
| A           |       1 |        3 |
| A           |       2 |       11 |
| A           |       3 |        3 |
| A           |       5 |        5 |
| A           |       8 |        1 |
| B           |       1 |        5 |
   snipped
```

see more

· Reply · Share ›

Xaprb · 9 years ago

Bonjour Pierre,

What you need is called a "cross tabulation" or "pivot table." Giuseppe Maxia wrote a great article about cross tabulations some time ago.

· Reply · Share ›

RN · 9 years ago

On the topic "Selecting the one maximum row from each group", how would this look if you got data from two tables, e.g. you split this table up with a column productid to link them together. In one table you have productid, type and variety. In the second table you have productid and price. Is it still possible to get it into one SQL statement?

· Reply · Share ›

Xaprb · 9 years ago

Certainly. You can do a subquery in the FROM clause, joining the two tables together in the suquery. Now they appear as one table to the rest of the query (hence the reason this is sometimes called a "derived table," which isn't technically incorrect, but it's a misuse of the term).

That's just a one-size-fits-all answer. In any given

query there are likely to be many other ways to do it.

Talahaski · 9 years ago

I have a similar problem I am trying to solve, but in my case I need to get only the top 1 per group, but within the group I need the greatest date and ID.

I was using the following

```
select
        a.CampaignID,
        a.Email,
        a.TS
into #tempLog
from dbo.xLog a
inner join (select email,max(TS) as TS from xLo
on a.email=b.email and
   a.TS = b.TS
where campaignid = @campaignid
```

This did not work because it turned out I had duplicate records with the same date (TS) per email

Byron V · 9 years ago

I found the "Selecting the one maximum row from each group" very helpful as well. It helped me write a query that I was finding difficult - joining a specific record from a grouped subquery back to the original table that I was grouping on in the subquery. I could not have done it without the clear and useful examples posted here.

Carl · 9 years ago

Genius and a life saver

trebskie · 9 years ago

This helped me do my task in an elegant way! Thanks for posting this! :)

David · 9 years ago

I was googling for a half hour and couldn't find what I

needed until I got here. Very clear explanation and exactly the code that I needed. Thanks very much!

matt · 9 years ago

Hello
great article :)
it would be interesting to edit the article and add Cristiano technique :-)

Ken · 9 years ago

Those are clear examples when using aggregate functions. I need to get the first record from a group, and the field is a char field. I am using advantage sql and do not have a first() func.

Specifically, I am linking two tables by a common field, Jobno. I need all
records from table a and only the first from b with the same jobno.

```
table a
jobno,quantity
111,22
112,23

table b
jobno,billno
111,aaa
111,aaa
111.bbb
112,ccc

The result I need is
jobno,billno
111,aaa
112,ccc
```

Thanks - Ken

Xaprb · 9 years ago

Ken, I don't know anything about SQL Advantage, but you should be able to apply the techniques I mention in this article. Don't make the common mistake of joining and then trying to extract. That is aggregation at the wrong place. Instead, aggregate

aggregation at the wrong place. Instead, aggregate and get the first record from table b. Now join that result to table a.

I have never heard of a first() function.

Rez · 9 years ago

Edit: Sorry, I didn't include formatting the first time I posted.

I've been searching high and low across the Internet kingdom for a solution to a problem I have. Your post has probably given the most insight so far but I'm still not there yet.

I have two tables - category references and posts and I would like to obtain the latest 10 posts for each of the categories with all results ordered in descending date order.

Category Reference Table

Stores cat_id to reference a category in a separate table and post_id referencing the post from the table

Matthieu Haller · 9 years ago

Simply great page.
It has taken me years to learn the hard way what's in this page, and I still learned a lot by reading it.
Thanks.

Warning, the different methods give the same results because each price is DIFFERENT in each group.
You have to be very carefull with queries like that if it's not the case.

If you want EXACTLY one line, Cristiano's method is the one you want. It has also the advantage to be deterministic : at equal price, you get the first variety alphabetically. Sometimes what you get is the last alphabetically (if you had searched for the max instead).
If that is a problem, you can play with binary

inversion of part of the string to get the correct order

see more

• Reply • Share ›

Michael Caldow · 9 years ago

Fantastic, so simple! As I needed exactly one row I used Cristianoâ€™s method (thanks for the tip Matthieu) and converted it to SQL server 2000 syntax, and in case this helps anyone else:

select type,
substring((
min(
right(replicate('0',10)
convert(varchar(10),price*100),10)
convert(varchar(10),variety)
))
,10,10 --add maximum possible length for variety here
)

,min(price)
from fruit
group by type

I tested this on my own data structure, then replaced the column and table name for those in the example.

• Reply • Share ›

Artur Filipiak · 9 years ago

If you dont expect too many values of a single type then it is possible to create a list of values of single type with a GROUP_CONCAT(cID) GROUP BY cType
1,2,3,...
4,5,6,...
7,8,9,...

select only first N values SUBSTRING_INDEX(cList, ',', 2)
1,2
4,5
7,8

join with the table again
1 - 1,2
1 - 4,5

1 - 7,8

see more

Reply · Share ›

Syanneth · 8 years ago

Hi! I have the following problem what i have this query i made from the article of finding duplicate rows but the downside of it is that it wonâ€™t put all the duplicates but only one because of the group by, i supposeâ€¦ what i want is to show all IPs alike and their accounts

Could you help me in this?

Here is the query iâ€™m using:
select username, last_ip from account group by last_ip HAVING count(*) >1

â€”â€”â€“ â€”â€”â€”â€“
|username|last_ip |
â€”â€”â€“ â€”â€”â€”â€“
abc	212.10.10.1
cde	212.10.10.1
ghi	198.12.50.4
ijk	198.12.20.2
â€”â€”â€“ â€”â€”â€”â€“

Reply · Share ›

angel · 8 years ago

if use GROUP statement,it will be a big problem to ORDER BY some fields which are not unique.

Reply · Share ›

martin · 8 years ago

thank you very much.

Reply · Share ›

mike · 8 years ago

Thanks! you helped me solve a problem. On Oracle databases only, a good way of limiting the number of rows returned is to use the system var 'rownum':
eg: select *
from (select ixa.alert_id , ixa.alert_title,
to_char(ixa.alert_date,'mm/dd/yyyy')
from iss_xf_alerts ixa
where ixa.oid = '5443')

order by ixa.alert_date desc) where rownum

Dan · 8 years ago

Great - very helpful!

I'd be interested in the performance implications of these various methods - it's easy testing on a simple db with a few rows, but how does (e.g. from your 'Selecting the one maximum row from each group'):

select f.type, f.variety, f.price
from (
select type, min(price) as minprice
from fruits group by type
) as x inner join fruits as f on f.type = x.type and f.price = x.minprice;

Work with a few million rows?

Cheers,
-Dan

WAWAN · 8 years ago

Genius ! this article just solve my half year headache :p

Oh and dan I just use this methods on hundred thousands of data, and no performance issues so far

CRIzz · 8 years ago

Hello Xaprb.
Iâ€™m novice in SQL statements.
I have one question if is possible
There is anyone how know how to make a query (using SQL selection) in access from one table.
(The table is made in MS Access 2003).
My table is like this:
- Name of table: database
- Columns: User_Name, Telephone, Time_call, Call_Duration_minutes.
I want to see the first 3 longest values for Call_Duration_minutes
Grouped by User_Name
I want the results to be something like:
User_Name Telephone Time_Call

Call_Duration_minutes
Aida 434443343 12:23 15
Aida 34342234 14:32 14
Aida 75656544 9:34 11
Cobal 44554554 19:34 57
Cobal 56544535 23:34 45
Cobal 89767554 01:23 5
Bob 34555432 03:45 125
Bob 78678676 12:23 21
Bob 67565444 17:54 11
- - - - - - - -ETC ETC ETC ETC - - - - - -
I made some queries but this one is to complicate for me,
Thank you very much!
Cheers,
CRizz

Reply · Share ›

Jordi Baylina · 8 years ago

Great Article!!!

I just want to point a couple of SQL extensions that I think it could be very helpfull and powerfull and not very difficult to implement.

Imagine a function called:
LPOSITION_IN_INDEX(, , ,...)

This function would return the position that a key would be inserted the key in a index. If the key alredy exists (not unique index) it would return the position just before (Left position).

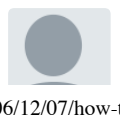
Suposing tht the table has the indes:

ALTER TABLE fruits ADD INDEX
XTypePrice(type,price);

the select would be:

SELECT type,variety,price
FROM fruits
WHERE LPOSITION_IN_INDEX(XTypePrice , type, price)-LPOSITION_IN_INDEX(XTypePrice, type,0)

Reply · Share ›



Jordi Baylina · 8 years ago

Great Article!!!

I just want to point a couple of SQL extensions that I think it could be very helpfull and powerfull and not very difficult to implement.

Imagine a function called:
LPOSITION_IN_INDEX(&ltINDEX_NAME&gt ,&ltKEY_1stPART&gt ,[KEY_2ndPART] ,â€¦)

This function would return the position that a key would be inserted the key in a index. If the key alredy exists (not unique index) it would return the position just before (Left position).

Suposing tht the table has the indes:

ALTER TABLE fruits ADD INDEX XTypePrice(type,price);

the select would be:

SELECT type,variety,price
FROM fruits
WHERE LPOSITION_IN_INDEX(XTypePrice , type, price)-LPOSITION_IN_INDEX(XTypePrice, type,0) &lt2

Jordi Baylina · 8 years ago

Another extension that I beleve that would be good is to define a function similar to MIN or MAX but with an extra argument. Lets call them MIN2 and MAX2

the select would be:

SELECT type,MIN2(price,variety),MIN(price)
FROM fruits
GROUP BY type

The idea is the same that CASANO's one but more elegant.

MIN2 and MAX2 could be defined in a mysql UDF very easyly!!

jalil · 8 years ago

very helpful
thanks

Michal Talaga · 8 years ago

Isn't the second method:

select type, variety, price
from fruits
where price = (select min(price) from fruits as f
where f.type = fruits.type

just wrong?

It more or less assumes that price will be unique. If the same price was repeated for 2 types of fruits and it happend to be the minumum price for one of them, the other fruit would also be returned.
Am I right?

2 · Reply · Share ›

sswong · 8 years ago

hi, refer to the second-cheapest of that type, the corelated way, i don't quite understand the count(*) for,
and another question is, what if i'm seeking for random 2 rows instead of the cheapest 2, i tried removing the price comparison, it doesn't work.
anyone care to enlighten me? thanks