Get top 1 row of each group

Ask Question

440

146

I have a table which I want to get the latest entry for each group. Here's the table:

`DocumentStatusLogs` Table

```
|ID| DocumentID | Status | DateCreated |
| 2| 1          | S1     | 7/29/2011   |
| 3| 1          | S2     | 7/30/2011   |
| 6| 1          | S1     | 8/02/2011   |
| 1| 2          | S1     | 7/28/2011   |
| 4| 2          | S2     | 7/30/2011   |
| 5| 2          | S3     | 8/01/2011   |
| 6| 3          | S1     | 8/02/2011   |
```

The table will be grouped by `DocumentID` and sorted by `DateCreated` in descending order. For each `DocumentID`, I want to get the latest status.

My preferred output:

```
| DocumentID | Status | DateCreated |
| 1          | S1     | 8/02/2011   |
| 2          | S3     | 8/01/2011   |
| 3          | S1     | 8/02/2011   |
```

- Is there any aggregate function to get only the top from each group? See pseudo-code `GetOnlyTheTop` below:

```
    GROUP BY DocumentID
    ORDER BY DateCreated DESC
```

- If such function doesn't exist, is there any way I can achieve the output I want?
- Or at the first place, could this be caused by unnormalized database? I'm thinking, since what I'm looking for is just one row, should that `status` also be located in the parent table?

Please see the parent table for more information:

Current `Documents` Table

```
| DocumentID | Title  | Content  | DateCreated |
| 1          | TitleA | ...      | ...         |
| 2          | TitleB | ...      | ...         |
| 3          | TitleC | ...      | ...         |
```

Should the parent table be like this so that I can easily access its status?

```
| DocumentID | Title  | Content  | DateCreated | CurrentStatus |
| 1          | TitleA | ...      | ...         | s1            |
| 2          | TitleB | ...      | ...         | s3            |
| 3          | TitleC | ...      | ...         | s1            |
```

UPDATE I just learned how to use "apply" which makes it easier to address such problems.

sql tsql sql-server-2005 group-by greatest-n-per-group

edited Dec 22 '17 at 7:25

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

12.8k 26 86 147

1 For a more detailed discussion and comparison of possible solutions I recommend to read the similar question on dba.se: Retrieving n rows per group. – Vladimir Baranov Nov 6 '16 at 10:33

I looked at the post and tried it. Using *group by StoreID* generated an error. – UltraJ Sep 6 '18 at 21:32

16 Answers

635

```
;WITH cte AS
(
   SELECT *,
         ROW_NUMBER() OVER (PARTITION BY DocumentID ORDER
   FROM DocumentStatusLogs
)
SELECT *
FROM cte
WHERE rn = 1
```

If you expect 2 entries per day, then this will arbitrarily pick one. To get both entries for a day, use DENSE_RANK instead

As for normalised or not, it depends if you want to:

- maintain status in 2 places
- preserve status history
- ...

As it stands, you preserve status history. If you want latest

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams
Q&A for work

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

answered Jul 27 '11 at 8:44

gbn
351k 60 493 584

3 And... What is `Partition By` ? `With` is new to me also :(I'm using mssql 2005 anyway. – dpp Jul 27 '11 at 8:48

4 @domanokz: Partition By resets the count. So in this case, it says to count per DocumentID – gbn Jul 27 '11 at 8:50

1 Hm, I worry about the performance, I'll be querying millions of rows. Is SELECT * FROM (SELECT ...) affects the performance? Also, is `ROW_NUMBER` some kind of a subquery for each row? – dpp Jul 27 '11 at 9:21

1 @domanokz: no, it's not a subquery. If you have correct indexes then millions shouldn't be a problem. There are only 2 set based ways anyway: this and the aggregate (Ariel's solution). So try them both... – gbn Jul 27 '11 at 9:30

1 @domanokz: Just change ORDER BY DateCreated DESC to ORDER BY ID DESC – gbn Jul 27 '11 at 9:52

141

I just learned how to use `cross apply` . Here's how to use it in this scenario:

```
select d.DocumentID, ds.Status, ds.DateCreated
from Documents as d
cross apply
    (select top 1 Status, DateCreated
     from DocumentStatusLogs
```

answered Aug 30 '12 at 6:10

dpp
12.8k 26 86 147

1 That actually makes no difference since the issue is still addressed. – dpp Sep 5 '12 at 5:57

13 I just posted the results of my timing tests against all of the proposed solutions and yours came out on top. Giving you an up vote :-) – John Mar 7 '15 at 15:00

2 +1 for huge speed improvement. This is much faster than a windowing function such as ROW_NUMBER(). It would be nice if SQL recognized ROW_NUMBER() = 1 like queries and optimized them into Applies. Note: I used OUTER APPLY as I needed results, even if they didn't exist in the apply. – TamusJRoyce Oct 19 '15 at 14:17

7 @TamusJRoyce you can't extrapolate that just because it was faster once this is always the case. It depends. As described here sqlmag.com/database-development/optimizing-top-n-group-queries – Martin Smith Jun 3 '16 at 21:26

1 My comment is about having multiple rows, and only desiring one of those multiple rows per group. Joins are for when you want one to many. Applies are for when you have one to many, but want to filter out all except a one to one. Scenario: For 100 members, give me each their best phone number (where each could have several numbers). This is where Apply excels. Less reads = less disk access = better performance. Given my experience is with poorly designed non-normalized databases. – TamusJRoyce Jun 5 '16 at 17:36

2008-R2, using a table with 6,500 records, and another (identical schema) with 137 million records. The columns being queried are part of the primary key on the table, and the table width is very small (about 30 bytes). The times are reported by SQL Server from the actual execution plan.

```
Query                                     Time for 6500 (ms)

CROSS APPLY                                      17.9
SELECT WHERE col = (SELECT MAX(COL)…)             6.6
DENSE_RANK() OVER PARTITION                       6.6
```

I think the really amazing thing was how consistent the time was for the CROSS APPLY regardless of the number of rows involved.

answered Mar 7 '15 at 14:57
John
867 8 13

4 It all depends on the data distribution and available indexes. It was discussed at great lengths on dba.se. – Vladimir Baranov Nov 6 '16 at 10:27

26

```
SELECT * FROM
DocumentStatusLogs JOIN (
  SELECT DocumentID, MAX(DateCreated) DateCreated
  FROM DocumentStatusLogs
  GROUP BY DocumentID
  ) max_date USING (DocumentID, DateCreated)
```

Regarding the second half of your question, it seems reasonable to me to include the status as a column. You can leave `DocumentStatusLogs` as a log, but still store the latest info in the main table.

BTW, if you already have the `DateCreated` column in the Documents table you can just join `DocumentStatusLogs` using that (as long as `DateCreated` is unique in `DocumentStatusLogs`).

Edit: MsSQL does not support USING, so change it to:

```
ON DocumentStatusLogs.DocumentID = max_date.DocumentID AND
DocumentStatusLogs.DateCreated = max_date.DateCreated
```

edited Jul 27 '11 at 8:49

answered Jul 27 '11 at 8:44

Ariel
20.6k 3 47 66

mine is MSSQL 2005. – dpp Jul 27 '11 at 8:47

4 The clue was in the title: MSSQL. SQL Server does not have USING but the idea is OK. – gbn Jul 27 '11 at 8:50

5 @gbn The stupid moderators usually delete important keywords from titles, as they have done here. Making it very difficult to find the correct answers in search results or Google. – NickG Sep 24 '15 at 8:41

Jus to point out that this "solution" can still give you multiple records if you have a tie on the `max(DateCreated)` – MoonKnight Nov 16 '17 at 21:40

23

If you're worried about performance, you can also do this with MAX():

```
SELECT *
FROM DocumentStatusLogs D
WHERE DateCreated = (SELECT MAX(DateCreated) FROM Document!
```

ROW_NUMBER() requires a sort of all the rows in your SELECT statement, whereas MAX does not. Should drastically speed up your query.

answered Jan 15 '13 at 20:57
Daniel Cotter
773 2 10 25

1 Cannot performance issues with ROW_NUMBER() be addressed with proper indexing? (I feel that should be done anyhow) – Kristoffer L Oct 22 '13 at 7:17

3 With datetime, you cannot guarantee two entries won't be added on the same date and time. Precision isn't high enough. – TamusJRoyce Oct 19 '15 at 14:20

+1 for simplicity. @TamusJRoyce is right. What about? 'select * from DocumentStatusLog D where ID = (select ID from DocumentsStatusLog where D.DocumentID = DocumentID order by DateCreated DESC limit 1);' – cibercitizen1 Jun 10 '17 at 16:22

SELECT * FROM EventScheduleTbl D WHERE DatesPicked = (SELECT top 1 min(DatesPicked) FROM EventScheduleTbl WHERE EventIDf = D.EventIDf and DatesPicked>= convert(date,getdate())) – Arun Prasad E S Feb 1 '18 at 7:58

There are definitely cases where this will outperform `row_number()` even with proper indexing. I find it especially valuable in self-join scenarios. The thing to be cognizant of

21

I know this is an old thread but the `TOP 1 WITH TIES` solutions is quite nice and might be helpful to some reading through the solutions.

```
select top 1 with ties
   DocumentID
  ,Status
  ,DateCreated
from DocumentStatusLogs
order by row_number() over (partition by DocumentID order |
```

More about the TOP clause can be found here.

answered Jan 24 '18 at 0:14
Josh Gilfillan
1,483 11 22

1 This is the most elegant solution imo – George Menoutis Oct 16 '18 at 13:20

9

This is quite an old thread, but I thought I'd throw my two cents in just the same as the accepted answer didn't work particularly well for me. I tried gbn's solution on a large dataset and found it to be terribly slow (>45 seconds on 5 million plus records in SQL Server 2012). Looking at the execution plan it's obvious that the issue is that it requires a SORT operation which slows things down significantly

```
SELECT
[Limit1].[DocumentID] AS [DocumentID],
[Limit1].[Status] AS [Status],
[Limit1].[DateCreated] AS [DateCreated]
FROM   (SELECT DISTINCT [Extent1].[DocumentID] AS [Documen
[DocumentStatusLogs] AS [Extent1]) AS [Distinct1]
OUTER APPLY  (SELECT TOP (1) [Project2].[ID] AS [ID], [Pro
[DocumentID], [Project2].[Status] AS [Status], [Project2].
    FROM (SELECT
        [Extent2].[ID] AS [ID],
        [Extent2].[DocumentID] AS [DocumentID],
        [Extent2].[Status] AS [Status],
        [Extent2].[DateCreated] AS [DateCreated]
        FROM [dbo].[DocumentStatusLogs] AS [Extent2]
        WHERE ([Distinct1].[DocumentID] = [Extent2].[Docum
    )  AS [Project2]
    ORDER BY [Project2].[ID] DESC) AS [Limit1]
```

Now I'm assuming something that isn't entirely specified in the original question, but if your table design is such that your ID column is an auto-increment ID, and the DateCreated is set to the current date with each insert, then even without running with my query above you could actually get a sizable performance boost to gbn's solution (about half the execution time) just from **ordering on ID instead of ordering on DateCreated** as this will provide an identical sort order and it's a faster sort.

answered Jun 3 '14 at 8:34

Clint
954 1 8 18

My code to select top 1 from each group

```
order by datecreated desc
)
```

answered Sep 23 '12 at 11:22

AnuPrakash
51 1 4

5

This is one of the most easily found question on the topic, so I wanted to give a modern answer to the it (both for my reference and to help others out). By using over and first value you can make short work of the above query:

```
select distinct DocumentID
  , first_value(status) over (partition by DocumentID orde
Status
  , first_value(DateCreated) over (partition by DocumentID
DateCreated
From DocumentStatusLogs
```

This should work in sql server 2008 and up. First value can be thought of as a way to accomplish select top 1 when using an over clause. Over allows grouping in the select list so instead of writing nested subqueries (like many of the existing answers do), this does it in a more readable fashion. Hope this helps.

answered Jan 18 '18 at 0:55

Randall
1,049 8 18

2

```
SELECT o.*
FROM `DocumentStatusLogs` o
  LEFT JOIN `DocumentStatusLogs` b
  ON o.DocumentID = b.DocumentID AND o.DateCreated < b.Dat
 WHERE b.DocumentID is NULL ;
```

If you want to return only recent document order by DateCreated, it will return only top 1 document by DocumentID

answered Dec 19 '16 at 15:10
cho
51 5

2

Verifying Clint's awesome and correct answer from above:

The performance between the two queries below is interesting. 52% being the top one. And 48% being the second one. A 4% improvement in performance using DISTINCT instead of ORDER BY. But ORDER BY has the advantage to sort by multiple columns.

```
IF (OBJECT_ID('tempdb..#DocumentStatusLogs') IS NOT NULL)
#DocumentStatusLogs END

CREATE TABLE #DocumentStatusLogs (
    [ID] int NOT NULL,
    [DocumentID] int NOT NULL,
    [Status] varchar(20),
    [DateCreated] datetime
)
```

```
1, 'S1', '8/02/2011 3:00:00')
INSERT INTO #DocumentStatusLogs([ID], [DocumentID], [Statu
2, 'S1', '7/28/2011 4:00:00')
INSERT INTO #DocumentStatusLogs([ID], [DocumentID], [Statu
2, 'S2', '7/30/2011 5:00:00')
INSERT INTO #DocumentStatusLogs([ID], [DocumentID], [Statu
2, 'S3', '8/01/2011 6:00:00')
INSERT INTO #DocumentStatusLogs([ID], [DocumentID], [Statu
3, 'S1', '8/02/2011 7:00:00')
```

Option 1:

```
    SELECT
    [Extent1].[ID],
    [Extent1].[DocumentID],
    [Extent1].[Status],
    [Extent1].[DateCreated]
FROM #DocumentStatusLogs AS [Extent1]
    OUTER APPLY (
        SELECT TOP 1
            [Extent2].[ID],
            [Extent2].[DocumentID],
            [Extent2].[Status],
            [Extent2].[DateCreated]
        FROM #DocumentStatusLogs AS [Extent2]
        WHERE [Extent1].[DocumentID] = [Extent2].[Document
        ORDER BY [Extent2].[DateCreated] DESC, [Extent2].[
    ) AS [Project2]
WHERE ([Project2].[ID] IS NULL OR [Project2].[ID] = [Exten
```

Option 2:

```
SELECT
    [Limit1].[DocumentID] AS [ID],
    [Limit1].[DocumentID] AS [DocumentID],
    [Limit1].[Status] AS [Status],
    [Limit1].[DateCreated] AS [DateCreated]
FROM (
    SELECT DISTINCT [Extent1].[DocumentID] AS [DocumentID]
```

```
                    [Extent2].[ID] AS [ID],
                    [Extent2].[DocumentID] AS [DocumentID],
                    [Extent2].[Status] AS [Status],
                    [Extent2].[DateCreated] AS [DateCreated]
                FROM #DocumentStatusLogs AS [Extent2]
                WHERE [Distinct1].[DocumentID] = [Extent2].[Do
            )  AS [Project2]
            ORDER BY [Project2].[ID] DESC
        ) AS [Limit1]
```

M$'s Management Studio: After highlighting and running the first block, highlight both Option 1 and Option 2, Right click -> [Display Estimated Execution Plan]. Then run the entire thing to see the results.

Option 1 Results:

```
 ID  DocumentID  Status  DateCreated
 6   1   S1  8/2/11 3:00
 5   2   S3  8/1/11 6:00
 6   3   S1  8/2/11 7:00
```

Option 2 Results:

```
 ID  DocumentID  Status  DateCreated
 6   1   S1  8/2/11 3:00
 5   2   S3  8/1/11 6:00
 6   3   S1  8/2/11 7:00
```

Note:

> I tend to use APPLY when I want a join to be 1-to-(1 of many).
>
> I use a JOIN if I want the join to be 1-to-many, or many-to-many.

I also avoid EXISTS / IN subqueries in the WHERE or ON clause, as I have experienced this causing some terrible execution plans. But mileage varies. Review the execution plan and profile performance where and when needed!

edited Nov 27 '17 at 21:38

answered Oct 28 '15 at 22:10

TamusJRoyce
431 7 19

0

In scenarios where you want to avoid using row_count(), you can also use a left join:

```
select ds.DocumentID, ds.Status, ds.DateCreated
from DocumentStatusLogs ds
left join DocumentStatusLogs filter
    ON ds.DocumentID = filter.DocumentID
    -- Match any row that has another row that was created
    AND ds.DateCreated < filter.DateCreated
-- then filter out any rows that matched
where filter.DocumentID is null
```

For the example schema, you could also use a "not in subquery", which generally compiles to the same output as the left join:

```
select ds.DocumentID, ds.Status, ds.DateCreated
from DocumentStatusLogs ds
```

Note, the subquery pattern wouldn't work if the table didn't have at least one single-column unique key/constraint/index, in this case the primary key "Id".

Both of these queries tend to be more "expensive" than the row_count() query (as measured by Query Analyzer). However, you might encounter scenarios where they return results faster or enable other optimizations.

answered Sep 4 '12 at 20:47

BitwiseMan
1,179 7 22

0

Try this:

```
        SELECT [DocumentID],
        [tmpRez].value('/x[2]','varchar(20)') as [Status],
 [tmpRez].value('/x[3]','datetime') as [DateCreated]
FROM (
        SELECT [DocumentID],
    cast('<x>'+max(cast([ID] as varchar(10))+'</x><x>'+[St
    +cast([DateCreated] as varchar(20)))+'</x>' as XML) as
        FROM DocumentStatusLogs
        GROUP by DocumentID) as [tmpQry]
```

edited Nov 6 '16 at 9:10

answered Nov 5 '16 at 11:57

gng
1 2

0

```
SELECT doc_id,status,date_created FROM (
SELECT a.*,Row_Number() OVER(PARTITION BY doc_id ORDER BY 
FROM doc a)
WHERE rnk=1;
```

answered Oct 16 '18 at 8:40

praveen
19 2

-1

This is the most vanilla TSQL I can come up with

```
SELECT * FROM DocumentStatusLogs D1 JOIN
(
  SELECT
    DocumentID,MAX(DateCreated) AS MaxDate
  FROM
    DocumentStatusLogs
  GROUP BY
    DocumentID
) D2
ON
  D2.DocumentID=D1.DocumentID
AND
  D2.MaxDate=D1.DateCreated
```

answered Jul 30 '15 at 12:25

rich s
11 1

Unfortunately MaxDate is not unique. It is possible to have two dates entered at the same exact time. So this can result in duplicates per group. You can, however, use an identity means two items added at exactly the same time are equally 'the latest' – rich s Mar 31 '17 at 10:56

Latest record will be one record. So yes. You need to consider the auto-increment identity column. – TamusJRoyce Nov 27 '17 at 21:20

-2

It is checked in SQLite that you can use the following simple query with *GROUP BY*

```
SELECT MAX(DateCreated), *
FROM DocumentStatusLogs
GROUP BY DocumentID
```

Here *MAX* help to get the maximum *DateCreated* FROM each group.

But it seems that MYSQL doesn't associate *-columns with the value of max DateCreated :(

edited Jan 22 '14 at 18:35

answered Jan 22 '14 at 18:07

malex
8,698 3 45 67

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook