Using GROUP BY in SQL

Posted At : April 1, 2005 11:17 AM | Posted By : Mark Kruger
Related Categories: SQL tips, Coldfusion & Databases

(reprinted from a previous blog)

Here's the dilemma. Let's say you have a log table where each row is a record of some action taken by a user. Perhaps the user logs in, updates his profile, searches for products and makes purchase. The table has the following fields:

- log_id (int) - a primary key for the table.
- user_id (int) - a foreign key reference to the "users" table.
- action_type (char) - a string indicating "login", "update", "search" or "purchase".
- UpdateTime (smalldatetime) - indicates when the event occurred.

Your boss (we'll call him Ralph) comes to you and says "I'd like to know the last action that each user took on the site before they went away. Can you build me a report like that? Well, you know a couple of ways to do this. One is that you can pull in the whole record set ordered by user and updatetime - then track the last record for each user. Or you might pull in all the users ids and do query looking for the max log_id for that user. Both of these approaches will work, but none of them will pull in the data required in a single query. Never Fear, there is another way - it's tricky, but it works!

irst, there is the problem of how to get that data both grouped and extracted. That's where the good old convert comes in. You can actually get at this data by converting the items in question to a string and concatenating the ids and date values together. For example, using the table described above: `The result of this query is a string built from the record that represents the oldest date in the record set for each user. Something like this:`

```
65_20040317
66_20040324
67_20040324
68_20040324
69_20040325
70_20040325
71_20040329
72_20040329
```

Ok, that's nifty (says ralph), but how do I get the "action" they took? Well, if you try to add that to your *Group by* clause you will end up with the oldest record per user *for each action*. Obviously that's not what you want either. Plus (and this is probably obvious to you by now) a string like 72_20040329 isn't terribly friendly to work with. I suppose you can unpack both the date and the user ID from that string and then query the db again for the action in question - but that is no better than the loop-d-loop issue we are trying to avoid.

You might not have thought of it, but this is a spot where a sub-query can really be useful. Using a subquery and a fancy where clause you can actually extract the whole row (primary key and all) in question from the log table. Here's how it's done:

```
SELECT log_id, User_id, Action_Type, UpdateTime
FROM    myLog
WHERE    CONVERT(varchar,user_id) + '_' +
```

```
      CONVERT(varchar,UpdateTime,112)
IN
( SELECT    CONVERT(varchar,user_id) + '_' +
         CONVERT(varchar,MAX(UpdateTime),112)
FROM    myLog
GROUP BY user_id
)
```

The trick is the MAX( ) function. When the root query concatenates the columns in the where clause it does it without regard to all rows in the table, but when the sub query runs it creates a string from only the maximum small date time for each user. The result is a subset of the records matching the "last action" that each user took before leaving the site. Ralph is impressed with you (of course) and immediately raises your salary - once again proving the old adage that if you build a better mouse trap people will beat a path to your door - usually trying to help you extricate your foot from your excellent device - but that is topic for another blog.