Using UNION in your queries

Posted At : July 25, 2005 11:24 AM | Posted By : Mark Kruger
Related Categories: SQL tips, MS SQL Server

Are you using Union queries yet? If not, you should get up to speed. A union query is an extremely useful method for returning records from different tables in the same recordset. You just have to remember that datatypes of the columns must match in the same order they are referenced. Here's an example:

```
<cfquery name="myQry" datasource="#dsn#">
   SELECT    ord_no, product,
         cost, shipping, status,
         OrdDt
   FROM   Orders
   WHERE   OrdDt Between '2005-07-18' AND '2005-07-19'
   UNION
   SELECT    ord_no, product,
         cost, shipping, status,
         OrdDt
   FROM   Orders_history
   WHERE   OrdDt dateBetween '2005-07-18' AND '2005-07-19'
</cfquery>
```

In this example we have a system that holds unposted orders in the 'orders' table. When they are posted they go from orders to the orders_history table. The union query above gives me information about the orders placed between specific dates - whether they were posted or not. There are a couple of gotchas. The order of the columns must be of the same data type. That means the columns can be *named differently*. They are only required to match for type. If the "shipping" column is "shipCost" in the Orders_history table we could easily do the following.

```
SELECT ord_no, shipping
   UNION
   SELECT ord_no, shipCost
```

The query would return results from both table and the first column specified (shipping) would be the name of the new column - although for readability an alias would be a better choice. Matching the type only and not the column name can make debugging tricky. What if you accidentally switched around shipping and cost and they were both decimal data types? You would end up with values reversed and your numbers would be off. So take great care in constructing your union query. As a side note, datatypes with "implicit" conversion (as in INT to decimal) will actually work - further complicating the issue.

Ordering in a union query

A union query has to have it's "order by" clause exist in the last query specified. It may not be in any other query in the group of queries. Sometimes you an do some clever things with Union queries ordering to give you "groups" of records. For example, let's say, in the query above, that I wanted to group by whether or not the record had been posted or not. In other words, I wanted to *order by* whether or not the record came from the "orders" table or the "orders_history" table. Obviously, the "status" field would probably provide some help there, but let's say there were multiple possibilities and ordering by "status" would give me more than the 2 groups I desire. Here's what I can

do.

```
<cfquery name="myQry" datasource="#dsn#">
   SELECT    ord_no, product,
         cost, shipping, status,
         OrdDt, 1 AS sortOrder
   FROM   Orders
   WHERE   OrdDt Between '2005-07-18' AND '2005-07-19'
   UNION
   SELECT    ord_no, product,
         cost, shipping, status,
         OrdDt, 2 AS sortOrder
   FROM   Orders_history
   WHERE   OrdDt dateBetween '2005-07-18' AND '2005-07-19'
   ORDER BY sortOrder
</cfquery>
```

Simple right? Add an int with the sort order you want and alias it as a column, then sort by that field.

Aggregate Functions using union

Recently a question came up on CF Talk about using UNION to aggregate data from more than 1 table. In other words, can I use my query to return a simple count of all the records from both tables. You do a select COUNT(*) from a subquery to make this happen.

```
<Cfquery name="myQuery" datasource="#dsn#">
SELECT count(*) FROM
    (
       SELECT    ord_no
      FROM   Orders
      WHERE   OrdDt Between '2005-07-18' AND '2005-07-19'
    UNION ALL
      SELECT    ord_no
      FROM   Orders_history
      WHERE   OrdDt dateBetween '2005-07-18' AND '2005-07-19'
    ) AS totalOrders
</CFQUERY>
```

A couple of things to note. First, it's easy to forget, but you need the "AS" clause (the alias) at the bottom of the query. Without it there is no "column" for CF to return to you. Secondly the "UNION" operator now says "UNION ALL". Without UNION ALL the UNION operator will automatically removed duplicates from the query - making it unreliable as a count if the same order number exists in both tables. In the case of orders vs. orders_history that's probably not possible, but in other cases it could be very important. Thanks to SQL GURU Joe Rinehart and several others who fiddled with this to get it to work. It could be very useful in certain cases.