Migrating Between MySQL to MSSQL

Posted At : July 13, 2009 6:02 PM | Posted By : Mark Kruger
Related Categories: MS SQL Server, Coldfusion & Databases

I recently did an emergency stint of troubleshooting for a site owner (a designer who owned a complex ColdFusion site) who was hit with the HTML injection issue on his site. He had done a good deal of work on his own and cleaned up the HTML as best he could. He was busy moving the sites to a more secure environment (a better hosting company, no more FTP, intrusion detection and solid VPN support). He had managed to travel a long way down the migration path before he ran into trouble. His new environment used MSSQL and his old environment used MySQL.

Now I love MSSQL and I think it is a wonderful choice (price notwithstanding), but had he contacted me before he decided to go this route I would have suggested that he stick with MySQL for the sake of compatibility. Unfortunately he had already "flipped the switch" before I got there and so there was a lot of "on the fly" changes to make just to get his site working correctly again. One of the biggest issues had to do with his choice for migrating the actual data. He had chosen to use an export tool to move the MySQL data into an Microsoft Access file. He then used Microsoft Access "upsize" wizard to send the data to the MSSQL server. The biggest flaw with this approach is that it resulted in missing dates which were not translated correctly from MySQL to Access to MSSQL. So we had to re-export the data in to SQL dumps, modify them and then run them against MSSQL.

The date problem is not a typical incompatibility with MSSQL, but there *are* several we ran into that we had to account for. Here they are in random order:

String Concatenation

MySQL uses a function called "concat()" for stringing together data types. It takes a list of column names or values and strings them together like so:

```
<cfquery ...>
    SELECT CONCAT('Welcome ', fname) as welcomemsg
    from users
</cfquery>
```

MySQL is very loose about what can be strung together as well. It uses predictable conversions so that things like a TEXT column can be strung together with a varchar or even an Int. MS SQL has more stringent rules about what can be concatenated *and* it uses the add operand (+) instead of a function. So the same code above in MSSQL would be:

```
<cfquery ...>
    SELECT 'Welcome ' + fname as welcomemsg
    from users
</cfquery>
```

In MS SQL, if I want to add 2 columns together that are, say, varchar and nText I cannot do i directly. If the column fname is varchar and notes is text then this would not work in MSSQL:

```
<cfquery ...>
    SELECT username + ' Notes: ' + notes
    from users
```

```
</cfquery>
</code>
I would get a datatype mismatch error. To make it work I would have to recast notes as
varchar:
<cfquery ...>
    SELECT username + ' Notes: '
            + CAST(notes as varchar(max))
    from users
</cfquery>
```

One thing you might notice is the "max" identifier in the varchar( ) declaration. This is new as of MSSQL 2005. In previous versions you would have to specify a number for length as in varchar(4000), but as of 05 you can tell MSSQL to use the max value allowed.

Q of a Q and Ordering.

This one caught me by surprise and it's difficult to explain. I had a complex query that used the "distinct" keyword to insure that all the values were distinct. The code included items in the "order by" clause that were not listed in the 'distinct' column list. This worked in MySQL, but MSSQL requires that ordering columns be part of the set (the select list) when distinct is used. Here's the tricky part. The code in question had a largish switch statement that created the Order by clause. It was obvious to me that a good bit of work would be necessary to make this compatible with MSSQL. So, I opted for a work around. I removed distinct allowing the code to run "as is", then added a query of a query to weed out any duplicates. Like so:

```
<!--- big complicated query --->
<cfquery name="bunchOfUsers" ...>
    SELECT col1, col2, col3, col4
    from users
    where ....
    order by
    <cfswitch>
        ... big switch statement...
    </cfswitch>
</cfquery>
<!--- q of a q for dups --->
<cfquery ...>
    SELECT distinct *
    FROM    bunchOfUsers
</cfquery>
```

Now this *seemed* to work - at least I found no more duplicates in the dataset. But as I looked further I found that the order had changed quite significantly between the 2 queries. The first query ordered by the ORDER BY clause and the second query ordered by some internal mechanism that I could not see - presumably Java array ordering under the hood. I had expected (foolishly) for the return values of the second query to be in the same order as the first query except the duplicates would be removed.

Yet, the more I think about it the more obvious it is. Java and Q of a Q have to do a number of ordering and comparison things to weed out the duplicates. Since I did not *tell* q of a q how to order it just assumes that whatever order it has at the time of return is fine. The fix was to re-implement the order by for Q of a Q - ignoring the columns that did not exist in the set of course.

LIMIT for Getting a Subset

In MySQL if you want to get just a subset of possible data you can limit the number of rows returned using the "LIMIT" keyword. It looks like this:

```
<cfquery ...>
    SELECT     *
    FROM    users
    ORDER BY createdDate
    LIMIT 10
</cfquery>
```

The equivilent code in MSSQL use *TOP N* at the beginning of the column list:

```
<cfquery ...>
    SELECT     TOP 10 *
    FROM    users
    ORDER BY createdDate
</cfquery>
```

Date Add

In MySQL, one of the ways you can add or subtract values from dates is using the operand and some pre defined keywords. For example, if you want to subtract a day from a date column in MySQL you might use something like this:

```
<cfquery ...>
    UPDATE Users
    SET lastLogin = NOW() - INTERVAL 1 DAY
</cfquery>
```

In MSSQL you use the dateAdd() function for this. DateAdd functions in MSSQL very much like it functions in ColdFusion.

```
<cfquery ...>
    UPDATE Users
    SET lastLogin = dateAdd(day,-1,getdate())
</cfquery>
```

The main difference between MSSQL dateadd() and ColdFusion dateAdd is that the interval identifier in MSSQL is typically a full word (day, month etc) and does not require single quotes around it.

Line Breaks in Text Fields

Character or fields in MySQL sometimes get exported with line breaks as the escape characters "*\r\n*". These characters show up in text outputted to the page and can cause formatting and readability issues. To fix it use the replace function in MSSQL:

```
<!--- Update a varchar --->
<cfquery ...>
    UPDATE Users
    SET notes = replace(notes,'\r\n','')
</cfquery>
<!--- Update a text field - must recast. --->
<cfquery ...>
    UPDATE Users
    SET long_desc = replace(CAST(long_desc AS varchar(max),'\r\n','')
</cfquery>
```

Random rows

MySQL has a clever little "rand()" function that allows you to sort a dataset randomly - meaning the order will be unpredictable each time. This can be useful if you have something like a "featured product" section where you have more product that can be displayed at a given moment and you want different products to appear on the page with each refresh. It's dead simple and works like this:

```
<cfquery ...>
    UPDATE top 100 *
    FROM users
    ORDER BY RAND()
</cfquery>
```

MSSQL 2005 finally has an easy equivilant to this called "newID()" that works exactly the same.

```
<cfquery ...>
    UPDATE top 100 *
    FROM users
    ORDER BY newID()
</cfquery>
```

Lessons Learned

Migrating between these two platforms is doable and fairly easy - but it should never be undertaken without testing. Now before you jump on my about ANSI standards etc. This is not a post that invites you to issue a diatribe on the evils of Microsoft or wax eloquent on why MySQL is missing A or B. Please confine your comments to helpful clues on migration between them. Otherwise we have to sacrifice a virgin and I'm afraid I left my fancy hat and loin cloth at home today.