HP e3000, CFMX and datetime

Posted At : April 5, 2005 10:39 AM | Posted By : Mark Kruger
Related Categories: HP e30000

We have a customer using the ODBC minisoft driver and the ODBC Socket driver (from CFMX) to connect to an HP e3000 legacy system. We are currently trying to get the JDBC driver to work, but that's a different story. The purpose of this entry is to detail a particularly tricky variance that seems to only affect this particular e3000 to ODBC to JDBC Socket setup. The reason it's worth a blog entry is because we spent 4 or 5 hours beating our heads against the wall - and we can save you a couple of concussions if you ever run into the same issue. So, if you have any projects using the e3000 and Coldfusion you might want to read on.

ur e3000 schema has a number of fields that it treats like datetime stamps. In reality they are simply 14 character strings - all numeric. Through design or error these fields (probably for the sake of ordering) are treated like numbers in the schema. A 14 character number with no decimals to be exact. To the e3000 this is fine and dandy. It uses a bigInt type to make it work.

Our problem is that Java - or more appropriately the socket driver that interfaces with the ODBC driver - sees this very large integer field and formats it as a "bigint" with scientific notation. So instead of *20050331083001*, we see something like "2.00503313E2". Now CFMX can convert that to a proper string using number format (numberFormat(bigNum,'______________')), but that is a bit clunky. After all we now have to loop through the HP query and reset all the values to something we can use - or we have to go through ALL the display code and add this number format mask to the outputs. Neither solution is a time saver. What we wanted was a way to have the query itself return the properly formatted string. After all, we are going to tease out the year, month, date and time as characters anyway - we don't actually need a number. We discovered the following works pretty well inside your HP query:

```
SELECT
   {fn CONCAT( {fn LEFT(dttmField,8)},{fn RIGHT(dttmField,12)} }
AS newdt
```

Since the bigInt type exists on the HP as a fixed length 20 character field it is necessary to grab the right 12 characters rather than the right 6 as you might expect. This code returns the string as we expect and eliminates the process of CF converting a large number to scientific notation - and the necessity to format it back to a big number only to treat it like a fixed length character field. The reason this works is because there is an implicit conversion when using the Left, right and Concat functions - the e3000 types the data for us - as character data rather than numeric.