

Handling Unicode Data types in MS SQL

Posted At : July 6, 2006 10:28 PM | Posted By : Mark Kruger

Related Categories: MS SQL Server

We have a customer who wants to support Asian languages. They have a lightweight CMS tool that they use to update portions of their website. Our first task was to duplicate this functionality for each language supported. Our first hurdle was the Chinese character set. We could update the DB directly through cut and paste, but the CF code we were using resulted in inscrutable question marks. We were using CFQUERYPARAM and none of the character types we tried worked. It looked as if we were up against a great wall.

Reaching back into the distant past I remembered something I had discovered years ago. While using the script generator to build a "create" script for a database I noticed that the "DROP" commands all looked like this:

```
<cfquery>
if exists
    (select * from dbo.sysobjects
     where id =
object_id(N'[dbo].[FK_architectural_feature_doors_architectural_features]')
     and OBJECTPROPERTY(id, N'IsForeignKey') = 1
    )
...Drop constraint....
</cfquery>
```

I had wondered why all the table and constraint definitions (character variables in the code above) had a big N in front of them. I discovered that an N keeps the DB server from applying a code page. In other words, it allows the characters to be treated "as is" instead of trying to format them. In reality it stands for "Native Language" - and forces the server to *not* treat it as any particular language. Of course you still need to be using double byte data types (ntext, nvarchar nchar instead of text, varchar and char). Here's a blurb from MS books on line about it.

MS Books Online RE: Constants

Unicode strings

Unicode strings have a format similar to character strings but are preceded by an N identifier (N stands for National Language in the SQL-92 standard). The N prefix must be uppercase. For example, 'Michél' is a character constant while N'Michél' is a Unicode constant. Unicode constants are interpreted as Unicode data, and are not evaluated using a code page. Unicode constants do have a collation, which primarily controls comparisons and case sensitivity. Unicode constants are assigned the default collation of the current database, unless the COLLATE clause is used to specify a collation. Unicode data is stored using two bytes per character, as opposed to one byte per character for character data. For more information, see Using Unicode Data.