Subject: Re: Oracle8: UTF8 charset Patch
Posted by rylek on Wed, 21 Jul 2010 20:48:38 GMT

Hello Tojocky!

I can easily live without an ASSERT in DisableUtf8Mode (I originally proposed it because I believed you had asked for it in an earlier message). I, however, don't think the main reason is that it's not a "system error". The main purpose of ASSERT / NEVER, I believe, is primarily not to catch situations which would otherwise trigger a hard error and application abort, but to bang the programmer over the head that they're doing something wrong in the sense they're using interfaces as they shouldn't. Which, I believe, includes calling DisableUtf8Mode at a wrong time, i.e. after establishing a connection.

But that's only a minor philosophical contemplation. What's worse, I've encountered a new error in a commercial database application of mine (WinZPV, surface water data processing I wrote for the Czech Hydrological Institute), which I found out to be related to our UTF8 patches.

The cause of trouble is of course the damned CLOB. I might understand it but I still hate it, however OCI8 interface counts CLOB length in characters, not bytes. Namely, OCILobGetLength returns number of characters in a CLOB and OCILobRead reads given number of characters at a given character offset.

This, of course, wreaks complete havoc when using UTF8 encoding, because the relation between byte and character offsets and lengths is not linear. Currently the only way around this problem I see is dividing OracleBlob into two classes, OracleBlob and OracleClob, where OracleClob would map the CLOB to Unicode byte pairs. This allows directly mapping the character-based count/offset OCI functions to file offsets (* 2) and random access to the input stream.

OracleBlob should then ASSERT that the handle it processes is really a BLOB, not a CLOB. Or it should do so at least in the UTF8 mode; in a "native encoding" mode this is not necessary, because the character set width is fixed and the relation "character = byte" holds.

Some internal hackery would of course be necessary to read the UTF8 stream data into an intermediate buffer and convert them on-the-fly to Unicode which would then be fed into the stream read queue. A similar procedure would have to be applied on output.

To be honest, I currently have no energy to write this (although I accept readiness to do it sometimes next week), moreover I believe this to be a partially backwards-incompatible modification and therefore I would appreciate your opinion and perhaps others' as well, at least Mirek's. Perhaps you can devise a smarter way to get along with things as they are, I can't.

Regards

Tomas