Database - Feature #2312

match Progress collation in a SQL Server database

06/02/2014 03:44 PM - Eric Faulhaber

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Database - Feature #2313: match Progress collation on a PostgreSQL...	**Hold**
Related to Database - Feature #2273: address remaining limitations of SQL Ser...	**Closed**

History

#1 - 06/12/2014 05:02 PM - Eric Faulhaber

- Assignee set to Ovidiu Maxiniuc

When sorting on a text column, Progress sorts its rows differently in certain cases than PostgreSQL does by default, given the same locale/charset. This most likely is the case for SQL Server, too. If we do not match this sorting behavior, we will have cases of mismatched results (e.g., unexpected content in reports or browse controls), and unexpected behavior (e.g., FOR EACH, FIND FIRST/LAST/NEXT/PREV, etc.) -- which could change the flow of programs. The easiest way to detect this is to populate a character/text field in a table with all manner of sample data (including fields with both alphanumeric and non-alphanumeric leading characters) in both Progress and SQL Server, and analyze the results of queries which sort on those fields.

For PostgreSQL and H2, we provide custom locale implementations for the en_US charset which match, as closely as possible, the default sorting behavior in Progress. For PostgreSQL, this is achieved with a custom, Linux locale which is installed at the operating system level, and upon which a database cluster is initialized (see p2j/locale/). For H2, this is achieved using Java's service provider interface architecture, to add a locale to the JVM's available list (see com.goldencode.p2j.spi).

Please review the P2J documentation regarding custom collation within the PostgreSQL and H2 databases (P2J Conversion Handbook -> Internationalization; P2J Developer Guide -> Part 1 -> Development Environment Setup -> Database Server -> PostgreSQL Server). This documentation discusses how to install a custom locale to enable Progress-like collation on a PostgreSQL database cluster. Unfortunately, it does not discuss how this locale was created.

In p2j/locale/, you will find the text form of the en_US@p2j_basic locale I created in 2006, as well as a tool (P2JLocaleHelper) I used to help create it. In order to understand what the locale is and how it is composed, you may need some background material: http://pubs.opengroup.org/onlinepubs/009696699/nframe.html (specifically, chapters 6 & 7).

I don't know how relevant any of this is to Windows and SQL Server (probably not directly), but we will need something that serves the same purpose for SQL Server (and PostgreSQL on Windows, for that matter -- but that will be handled in a separate issue #2313), assuming we find differences compared to Progress' collation.

The input I used to the P2JLocaleHelper tool were collation tables I exported from an older version of Progress. I'm not sure these are provided by the newer versions. Assuming we will need to create custom locales for the en_GB character set, we may need to write test cases to determine the collation behavior, and base our collation order on their results.

#2 - 06/13/2014 02:19 PM - Ovidiu Maxiniuc

- File installed-collations-sql2012express.ods added

- Status changed from New to WIP

- File collation-test.p added

I am not sure we create a user-defined collation in SQL Server :(. From my research we have to be lucky enough to find one of the built-in collations. The first attached file is the list of all available collations from my SQL Server 2012 Express. There are a few thousands, but most of them are particular cases of i18n, including case-sensitivity (CS / CI), accent sensitive (AS / AI) and support for Chinese kana glyphs (KS / KI).

The order in which strings are sorted in p4gl is detected by the attached Progress procedure. It created a temp tables and populate it with 256 strings that have only the middle character changed. When the table is listed using the Progress sorting, the records are rearranged so the mapping is given by the permutation of ix field. To find the correct collation we need to find the one from sql server that matches the ix order in equivalent sql procedure. This is only a mandatory requirement because the collations (not sure about 4gl) can check more than one character when sorting strings.

So, I also wrote the corresponding procedure in t-sql language (also attached).

I have not yet tested all possible candidates. The viable ones are SQL_* and Latin1_General_*, the other like Albanian_*, Arabic_* are excluded.

#3 - 06/13/2014 02:20 PM - Ovidiu Maxiniuc

- File collation-test.sql added

Forgot to add collation test for Sql Server.

#4 - 06/13/2014 03:55 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> The order in which strings are sorted in p4gl is detected by the attached Progress procedure.

Nice. Please be sure to run this both on windev01 and lindev01, to make sure there are no surprises between them.

#5 - 06/16/2014 04:53 AM - Ovidiu Maxiniuc

- File collation-test.lindev01.4gl.txt added

As expected, the order is the same on both windows and linux implementations. The output file is attached.

I go on testing checking the available collations of SQL Server. If none matches P4GL, this might be a new MSSQL specific problem we need to overcome.

#6 - 06/16/2014 05:36 PM - Ovidiu Maxiniuc

- File Latin1-General-Collations comp.ods added

Apparently none of available collations are good enough.
Attached a spreadsheet with summary of the research.

- 1st sheet contain the Available Non-i18n Collations on SQL-Windows. Language specific were eliminated in order to reduce the number of tested collations from about 4K to a manageable amount. Collations with unicode-spacific kana and supplimentary sets were not tested.
- 2nd sheet contains the groups of chars that are EQ from P4GL POV. Ex: all lower and upper case, accentuated and non-accentuated A are equals even if they have different CHR code.
- 3rd sheet contains the summary of the compare operation. The 1st 2 columns contain the CS and CI order extracted using previously attached procedure. The 3rd is the collation taken from ISO_8859_1Collator / en_US@p2j_basic form P2J project. The space (CHR / #20) doesn't look correct as the 1st character to me. Next columns are the SQL Server Collations, grouped by CP. There is no a perfect match for ISO 8859-15 that is mentioned in the footer of the exported files, however, the latin1 (ISO 8859-1) is very close (only 8 chars are different). Overall, the character distribution is very different for SQL collations (I highlighted some characters to make changes visible: A/a with red, not the accentuated, B/b with blue, the d/D/ð/Ð with magenta- they were fewer and Z/z/Ž/ž in green - these last 2 are some new addition to 8859-15).

#7 - 06/20/2014 07:42 AM - Eric Faulhaber

Is it possible to create new collations/locales for Windows itself (as we do on Linux) and then use these with SQL Server, or does SQL Server only work with its pre-packaged list of available collations?

#8 - 06/20/2014 09:57 AM - Ovidiu Maxiniuc

Reading [[http://technet.microsoft.com/en-us/library/ms175194(v=sql.110).aspx]], I understand that SQL Server has a built-in set of collations that *match* those of windows. They are different resources, unlike PostgreSQL that uses the resources form Linux OS.

I googled a lot in search for 'custom', 'new' or 'user-defined' collations for SQL server and I could not find anything except questions of others.

Looking over the distribution and character groups treated by 4GL as equals, the correct collation should only have the _AI (accents ignored) sorting style because all accented letters are treated as equals by Progress sorting. On the other hand, _CS (case sensitivity) is irrelevant, as we handle this in application, using computed columns. _KS (Japanese kana characters, Hiragana and Katakana) sort style and _WS (multibyte, width-sensitive) are to be ignored as long as the imported data is (at least for the customer) encoded in ISO8859-15 code page (8-bit, single-byte coded character set).

To workaround this issue, I wonder if we could create some kind of mapping to be used just before persisting the character columns. Of course, they need to be mapped back in order to be correctly processed in application. But, in order that this idea o work, we need some MSSQL collation that has the same groups of equal characters (accented ones).

#9 - 06/27/2014 06:29 PM - Eric Faulhaber

Please choose from the available collations the ones you feel are the closest, practical matches for Progress' collation behavior. I understand they will not be perfect matches. Please document the key differences for those few. We will let the customer decide which is most appropriate for them, based on their specific requirements, knowing that there will be known limitations that may affect the processing of their application.

#10 - 07/01/2014 09:41 AM - Ovidiu Maxiniuc

I have run some scans over the database dump from lincon01, indexing the occurrences for each character in strings. I was hoping to discover that the customer is not using the whole set of Latin-9 (ISO-8859-1) but rather a subset of the range of 256. This would have allowed to pick on of the available collations to match the Progress collation for that subset.

As I understand that this is only a sample database (possible not very representative) this is only a case-study. Possibly, this is part of the solution, to choose the appropriate collation for each installation based on the character distribution.

From my discoveries, the Progress collation has some particularities that make it not fully compatible with any of SQL Server collations (in fact not even with PSQL, but it has the great advantage that can use a customized collation):

- First of all, the first 64 characters (including white-spaces, digits and punctuation) are sorted in their strict natural order, which make it compatible with BIN -ary collations of MSSQL for this subset.
- Then, the following 64 characters up to 127 are also sorted (when case-sensible) also in their natural order. This includes the both uppercase and lowercase chars and the punctuation in the region. What is different from the 1 st subset is that the order is not strict in the way that accented forms of the letters are inserted with their unaccented forms to which they are considered equals (as you know, in Progress, À, Á, Â, Ã and Ä are all equals to A - but not Å, nor Æ). In this regard, most SQL_Latin1_General collations do the correct sorting except for Å and Æ that are sorted next to A.
- The characters starting from 128 are of two kind:
  - accented letters. These were discussed above.
  - other characters (the rest 67 of them). These are really messed up and I could not find any logical order. They are sorted last in Progress (after 127 in ASC order). In MSSQL, SQL_Latin1_General collations sort them usually somewhere before A and BIN collations usually sort them last.
- A special mention should be added for the \0 character. While most of MSSQL collations will sort it as 1 st, Progress collation is very weired, from my testing, it occurs between lowercase y and z. This is debatable however. Depending on how the database engine sees it. Most likely, MSSQL treats it as a normal character and sort it fist for most collations. On the other hand, I believe for Progress, the \0 character is 'invisible' so the next character of the string was used for sorting.

As conclusion, there is no MSSQL collation to satisfy Progress way of sorting. If the strings from database would not contain accented and other character from the second half of the set some *BIN collation would just work. If the strings would would not contain digits and punctuation up to A char(65), there are some Latin_General collations that might work.

The last idea that I have in mind and think of as a possible solution is to use a dedicated sorting column (perhaps the same computed column we are using now). This column would contain the initial string morphed using the table from com.goldencode.p2j.spi.ISO_8859_1Collator. Each occurrence of À, Á, Â, Ã and Ä will be replaced with the equal A, and so on, the second half of the ASCII table would be permuted to fit the order from the points table.
The problems here is to maintain this structure with data updates (maybe a native database trigger), and to force SQL to always use this column for sorting when the initial char value is used in some fancy expressions.

#11 - 11/20/2014 02:47 PM - Ovidiu Maxiniuc

- File Latin1-General-Collations compared.xls added

This is a newer version of the Latin1-General-Collations comp.ods in the old legacy Microsoft Office 2003 xls format.
The most important is the 3rd sheet (Compare SQL Collations to 4GL).
It contain some additional research on the groups of characters that P4GL collations treat as equals. Some of them have been highlighted. For example A and a are in red, the capitals are in bold. Z / z are in green. To have a perfect match, each cell of the whole column should be equal to corresponding cell of 1st (Case Insensitive) or 2nd (Case Sensitive) column.

#12 - 03/30/2016 01:00 PM - Eric Faulhaber

- Target version deleted (Milestone 11)

I am removing this issue from M11, because it has been taken as far as the SQL Server limitations will allow us. It will be up to customers to decide which of the standard collations available meet their needs best. However, I have related this issue to #3042 for reference, as I suspect some of the differences in behavior will be related to collation.

#13 - 01/19/2017 05:53 PM - Greg Shah

These are two useful references for Windows Custom Locales and Working with Custom Locales

It is definitely possible to implement a custom locale in Windows. Microsoft does this so that customers that aren't otherwise supported can support themselves. Although there is some text saying that sorting uses only "Microsoft-supplied sorts", I believe this is referencing that the core sorting APIs in WIN32 are not replaceable. We will need to use the Locale Builder tool (see below) which is provided to customize locales (or create a new one) to see what is possible. I would be very surprised if it wasn't possible to control/customize the collation tables inside the custom locale. The APIs in WIN32 can't be replaced, but by changing the tables the ordering can be controlled and customized. It should be possible otherwise custom locales would be impossible.

This reference explains SQLServer Collation and Unicode Support information. Based on SQLServer 2016, it is possible to use Windows locales and also something they call Binary locales. These are both different from the SQLServer locales which are now only used for compatibility with older SQLServer versions. It seems we have some new options.

Locale Builder 2.0 can be used to customize locales.

#14 - 01/19/2017 06:10 PM - Greg Shah

- Related to Feature #2273: address remaining limitations of SQL Server needed to support legacy 4GL behavior added

Files

installed-collations-sql2012express.ods	56.5 KB	06/13/2014	Ovidiu Maxiniuc
collation-test.p	1.03 KB	06/13/2014	Ovidiu Maxiniuc
collation-test.sql	916 Bytes	06/13/2014	Ovidiu Maxiniuc
collation-test.lindev01.4gl.txt	4.15 KB	06/16/2014	Ovidiu Maxiniuc
Latin1-General-Collations comp.ods	76.3 KB	06/16/2014	Ovidiu Maxiniuc
Latin1-General-Collations compared.xls	526 KB	11/20/2014	Ovidiu Maxiniuc