FWD - Feature #2949

Alternative approach to handling case insensitivity

01/08/2016 05:24 AM - Paul E

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

Description

Related issues:

Related to Database - Bug #6837: prevent unnecessary upper/rtrim injection in...	**WIP**

History

#1 - 01/08/2016 05:25 AM - Paul E

We understand how your P2J implementations of this work for PostgreSQL and SQL Server, and we understand why these choices were made. There is no functional concern about this for most use-cases: the P2J version of our application will behave as it should; our main approach to reporting will be fine (we push all of the data into a SQL-server database and reports run from that DB). However, there are some use cases involving customers reading directly from the DB. There are also some administration use cases to do things like data fixes. In these cases, the extra effort to ensure you're using functional indexes or including calculated fields in your queries would be hard for us to justify and enforce.

As such, we'd like an alternative implementation to support use of DB-native case insensitive features rather than the current approach (perhaps as a toggle-able option if you prefer). This alternative implementation would use case insensitive text fields for PostgreSQL (citext) and a case insensitive collation for MS SQL Server. We would do right trimming pre-import (so your import routines wouldn't need to change).

We see this as something we'd prioritise along with the rest of the enhancements to be considered once the main functional milestones have been achieved.

#2 - 01/08/2016 05:29 AM - Paul E

Some benefits of this approach:

- easier to write ad-hoc queries
- no difference between queries written for PostgreSQL or SQL Server
- more similarity in PostgreSQL and SQL Server schemas
- performance benefits of not having to execute these functions in where clauses

Drawbacks:

- Implementation and support effort - to Quote Eric from an email:

  ... and I am concerned about support especially. Consider that if problems arise from the change (i.e., an incompatibility with Progress behavior), we won't know whether it's due to this (and thus, presumably acceptable as a divergence from the original behavior, or a customer's responsibility to change/fix), or a "real" bug, until we have taken the hit to diagnose any such problems. This is a primary driver as to why we have been so slavish in maintaining compatibility as closely as possible.

#3 - 11/16/2016 01:16 PM - Greg Shah

- Target version deleted (24)

#4 - 11/01/2022 09:19 AM - Greg Shah

- Related to Bug #6837: prevent unnecessary upper/rtrim injection in queries if the data type doesn't require it added

#5 - 11/01/2022 09:27 AM - Greg Shah

- Start date deleted (01/08/2016)

We are considering changes to the PostgreSQL schema. One possible is to use a case-insensitive text type (e.g. something like CITEXT) instead of TEXT. As Eric notes above, this will require intrusive changes in the persistence layer and at conversion time. We are already doing this for use in MariaDB. Note that at least inMariaDB, there are issues with fixed length fields (the type used there truncates data). It doesn't look like CITEXT is fixed length. On the other hand, we will have some other issues:

- It uses lower instead of upper for case-insensitivity. This is known to break certain use cases (upper matches 4GL behavior better).
- We still have an issue with RTRIM() for ignoring trailing whitespace in comparisons.
- CITEXT is now a trusted modules which means it can be loaded by non-superusers and it should exist everywhere (e.g. Amazon RDS or Aurora). But it may still require an extra setup step.
- Based on the documentation, the performance of CITEXT may be better in some cases and worse in others.

#6 - 11/01/2022 11:59 AM - Eric Faulhaber

Greg Shah wrote:

> It uses lower instead of upper for case-insensitivity. This is known to break certain use cases (upper matches 4GL behavior better).

Do you recall where these cases are documented?

#7 - 11/01/2022 12:37 PM - Greg Shah

> > It uses lower instead of upper for case-insensitivity. This is known to break certain use cases (upper matches 4GL behavior better).
>
> Do you recall where these cases are documented?

See #1587-235. And we more recently found issues with uppercasing. See #5085. We need testcases to explore the full extent of the problem.