L2/17-050R

Georgian: Comments on Database Stability

Author: Steven R. Loomis @srl295 srloomis@us.ibm.com

Updated: 9 მაისი, 2017 (May 9, 2017)

Introduction

Relevant to the response to point 2 in L2/17–045 (Razmadze), here are a few comments on how addition of codepoints can cause database instability when different implementations are used.

Background

> 1. If a new set of capital letters were added now, it would be very destabilizing and lead to serious representation and interoperability issues. Names in databases, for example, would be destabilized if capitalization was introduced. Searching and comparison operations would no longer give expected results, which would be significant when analyzing the large corpus of existing documentation.
>
> We have had consults with local specialists and based on their conclusions, names in databases will not be causing destabilization. There won't be any problems with searching, either, since Mtavruli and Mkhedruli letters will be linked through the case pairing function and Mtavruli letter results will also be represented in Mkhedruli letter searches, just like in Latin letter search for the word "GEORGIA" we will receive the lowercase result – "georgia" as well.

Overview

Changing the encoding of Georgian to disunify Mkhedruli/Mtavruli has effects beyond those within a single closed system. For example, if a single computer was upgraded to support this disunification, that computer would have an updated keyboard/keyboard layout, updated fonts, and updated software (perhaps a word processor) which would recognize "მადლობა"/"მᲐᲓᲚᲝᲑᲐ" as equivalent but with case differences. If the user of computer then were to interchange documents with other computers— whether interacting with web servers or sending a document file— any system with a lack of support for disunification would simply not support the Mtavruli content until that system were upgraded. A non-upgraded system might only show ??? or ××× or □□□ until such time as the system is upgraded. This is an inconvenience, but can be resolved in time.

However, what this present document will discuss is the more serious effect which a casing change has on operations within a database system or within multiple interoperating databases. The issues discussed would not simply result in display problems, but could result in loss or corruption of data, potentially making some database records inaccessible.

If a database represents something such as patient medical records, a missing record could result in not only inconvenience but serious injury.

Databases today

A real example from MySQL today illustrates the current situation:

```
UPPER('ჭიჭიკო ბენდელიანი')
ჭიჭიკო ბენდელიანი
UPPER('суп')
СУП
```

and SQLite:

```
sqlite> select UPPER('ჭიჭიკო ბენდელიანი');
ჭიჭიკო ბენდელიანი
```

So today `UPPER('ჭიჭიკო ბენდელიანი')` just produces `ჭიჭიკო ბენდელიანი`, not `ჭიჭიკო ბენდელიანი`

Effect of disunification

Suppose there were a patient records table with a constraint that patient names must be in uppercase (via the function `UPPER`). Currently, Georgian text would satisfy this requirement (as per above). I am going to display the example in Mkhedruli as it is "current Unicode".

id	name	medicine
2	ჭიჭიკო ბენდელიანი	TRUE
8	TEST PATIENT	FALSE

We can search for patients:

```
sqlite> SELECT * FROM PATIENTS WHERE NAME LIKE UPPER('TEST PATIENT');
```

id	name	medicine
8	TEST PATIENT	FALSE

```
sqlite> SELECT * FROM PATIENTS WHERE NAME LIKE UPPER('ჭიჭიკო ბენდელიანი');
```

id	name	medicine

2	ჭიჭიკო ბენდელიანი	TRUE

Because the uppercase version is the same:

```
sqlite> select UPPER('ჭიჭიკო ბენდეღიანი');
```

UPPER('ჭიჭიკო ბენდელიანი')
ჭიჭიკო ბენდელიანი

Unicode upgrade

However, if the database system is now upgraded to support disunified Mtavruli we now have the following: (I added another patient whose name is now Mtavruli for comparison).

```
sqlite> SELECT * FROM PATIENTS WHERE NAME LIKE UPPER('ჭიჭიკო ბენდეღიანი');
```

id	name	medicine
2	ჭიჭიკო ბენდელიანი	TRUE
8	TEST PATIENT	FALSE
16	მადლობა ბენდელიანი	FALSE

Now, we can **no longer** locate patient #2, even with the very same SQL query:

```
sqlite> SELECT * FROM PATIENTS WHERE NAME LIKE UPPER('ჭიჭიკო ბენდეღიანი');
(no results)
```

This is because the uppercase function has changed.

```
sqlite> select UPPER('ჭიჭიკო ბენდეღიანი');
```

UPPER('ჭიჭიკო ბენდელიანი')
ჭიჭიკო ბენდელიანი

For simplicity, a `SELECT` statement was shown. But the above scenario could also occur if there is a foreign key, stored procedure, or constraint. In other words, other data or code could need to interact with the patient record table.

Remediation

While these difficulties are definitely problematic, they are not too difficult to be solved with careful work. Upgrading the Unicode version of such databases must be done on a very careful basis. In some cases, delaying an upgrade may be the right solution for a system.

Maintaining a list or some venue for discussion of these issues and discussing them among database administrators would be important to ensure a smooth transition.

Colophon