Article
Alexander Koblov · May 20, 2016 12m read

Collations in Caché

> Order is a necessity for everyone, but not everyone understands it in the same way
> (Fausto Cercignani)

Disclaimer: This article uses Russian language and Cyrillic alphabet as examples, but is relevant for anyone who uses Caché in a non-English locale.
Please note that this article refers mostly to NLS collations, which are different than SQL collations. SQL collations (such as SQLUPPER, SQLSTRING, EXACT which means no collation, TRUNCATE, etc.) are actual functions that are explicitly applied to some values, and whose results are sometimes explicitly stored in the global subscripts. When stored in subscripts, these values would naturally follow the NLS collation in effect ("SQL and NLS Collations").

Everything in Caché is stored in globals: data, metadata, classes, routines. Globals are persistent. Global nodes are ordered by subscript values, and stored on storage devices, not in insertion order, but in sorted order for better search and disk fetch performance:

```
USER>set ^a(10)=""
USER>set ^a("??")=""
USER>set ^a("??")=""
USER>set ^a(2)=""
USER>zwrite ^a
^a(2)=""
^a(10)=""
^a("??")=""
^a("??")=""
```

During sorting, Caché distinguishes numbers and strings — 2 is treated as number and sorts before 10. Command ZWrite and functions $Order and $Query return global subscripts in the same order these subscripts are stored: first empty string (you cannot use it as subscript), then negative numbers, zero, positive numbers, then strings in the order defined by collation (collation).

Standard collation in Caché is called (unsurprisingly) — Caché standard. It sorts each string accordingly to their Unicode character codes.

Collation for local arrays in current process is defined by locale (Management Portal > System administration > Configuration > System Configuration > National Language Settings > Locale Definitions). Russian locale for Caché Unicode installations is rusw, and default collation for rusw is Cyrillic3. Other possible collations in the rusw locale are Caché standard, Cyrillic1, Cyrillic3, Cyrillic4, Ukrainian1.

ClassMethod ##class(%Collate).SetLocalName() sets collation for local arrays in current process:

```
USER>write ##class(%Collate).GetLocalName()
Cyrillic3
USER>write ##class(%Collate).SetLocalName("Cache standard")
1
USER>write ##class(%Collate).GetLocalName()
```

```
Cache standard
USER>write ##class(%Collate).SetLocalName("Cyrillic3")
1
USER>write ##class(%Collate).GetLocalName()
Cyrillic3
```

For every collation, there is a fellow collation that sorts numbers as strings. Name of that collation contains " string " in the end:

```
USER>write ##class(%Collate).SetLocalName("Cache standard string")
1
USER>kill test

USER>set test(10) = "", test(2) = "", test("??") = "", test("??") = ""

USER>zwrite test
test(10)=""
test(2)=""
test("??")=""
test("??")=""

USER>write ##class(%Collate).SetLocalName("Cache standard")
1
USER>kill test

USER>set test(10) = "", test(2) = "", test("??") = "", test("??") = ""

USER>zwrite test
test(2)=""
test(10)=""
test("??")=""
test("??")=""
```

Caché standard and Cyrillic3

Caché standard sorts characters accordingly to their codes:

```
 write ##class(%Library.Collate).SetLocalName("Cache standard"),!
 write ##class(%Library.Collate).GetLocalName(),!
 set letters = "????????????????????????????????"
 set letters = letters _ $zconvert(letters,"U")
 kill test

 //fill local array “test” with data
 for i=1:1:$Length(letters) {
     set test($Extract(letters,i)) = ""
 }

 //print test subscripts in sorted order
 set l = "", cnt = 0
 for  {
     set l = $Order(test(l))
     quit:l=""
     write l, " ", $Ascii(l),","
```

```
     set cnt = cnt + 1
     write:cnt#8=0 !
 }
```

```
USER>do ^testcol
1
Cache standard
? 1025,? 1040,? 1041,? 1042,? 1043,? 1044,? 1045,? 1046,
? 1047,? 1048,? 1049,? 1050,? 1051,? 1052,? 1053,? 1054,
? 1055,? 1056,? 1057,? 1058,? 1059,? 1060,? 1061,? 1062,
? 1063,? 1065,? 1066,? 1067,? 1068,? 1069,? 1070,? 1071,
? 1072,? 1073,? 1074,? 1075,? 1076,? 1077,? 1078,? 1079,
? 1080,? 1081,? 1082,? 1083,? 1084,? 1085,? 1086,? 1087,
? 1088,? 1089,? 1090,? 1091,? 1092,? 1093,? 1094,? 1095,
? 1097,? 1098,? 1099,? 1100,? 1101,? 1102,? 1103,? 1105,
```

Cyrillic letters are printed in the same order they go in Russian alphabet, except ‘ ё ’ and ‘ Ё ’. Their Unicode character codes are out of order. ‘ Ё ’ should be collated between ‘ Е ’ and ‘ Д ’ and ‘ ё ’ between ‘ е ’ and ‘ д ‘. That ’ s wh locale needs its own collation — Cyrillic3, which has letters in the same order as in Russian alphabet:

```
USER>do ^testcol
1
Cyrillic3
? 1040,? 1041,? 1042,? 1043,? 1044,? 1045,? 1025,? 1046,
? 1047,? 1048,? 1049,? 1050,? 1051,? 1052,? 1053,? 1054,
? 1055,? 1056,? 1057,? 1058,? 1059,? 1060,? 1061,? 1062,
? 1063,? 1065,? 1066,? 1067,? 1068,? 1069,? 1070,? 1071,
? 1072,? 1073,? 1074,? 1075,? 1076,? 1077,? 1105,? 1078,
? 1079,? 1080,? 1081,? 1082,? 1083,? 1084,? 1085,? 1086,
? 1087,? 1088,? 1089,? 1090,? 1091,? 1092,? 1093,? 1094,
? 1095,? 1097,? 1098,? 1099,? 1100,? 1101,? 1102,? 1103,
```

Caché ObjectScript has a special binary operator]] — «sorts after». It returns 1, if subscript with first operand sorts after subscript with second operand, and 0 otherwise:

```
USER>write ##class(%Library.Collate).SetLocalName("Cache standard"),!
1
USER>write "?" ]] "?"
1
USER>write ##class(%Library.Collate).SetLocalName("Cyrillic3"),!
1
USER>write "?" ]] "?"
0
```

Globals and collations

Different globals in the same database may have different collation. Each database has a configuration option — default collation for new globals. Right after the installation all databases except USER use the default collation of Caché standard. Default collation for USER database is determined by installation locale. For rusw it is Cyrillic3.

To create a global with collation that is non-default for its database use ##class(%GlobalEdit).Create method:

```
USER>kill ^a
USER>write ##class(%GlobalEdit).Create(,"a",##class(%Collate).DisplayToLogical("Cache
 standard"))
```

There is a collation column for each global In list of globals in Management Portal (System Explorer > Globals).

You cannot change collation of existing global. You should create global with new collation and copy data with Merge command. To do mass conversion of globals use ##class(SYS.Database).Copy()

Cyrillic4, Cyrillic3, and umlauts

It turns out that conversion of string subscript to internal format takes noticeably more time with Cyrillic3 collation than with Caché standard collation, therefore insert and lookup for global (or local) array with Cyrliic3 collation is slower. Caché 2014.1 contains new collation — Cyrillic4 that has the same correct order of letters as Cyrillic3 and better performance.

```
for collation="Cache standard","Cyrillic3","Cyrillic4" {
     write ##class(%Library.Collate).SetLocalName(collation),!
     write ##class(%Library.Collate).GetLocalName(),!
     do test(100000)
 }
 quit
test(C)
 set letters = "????????????????????????????????????"
 set letters = letters _ $zconvert(letters,"U")

 kill test
 write "test insert: "
 //fill local array “test” with data
 set z1=$zh
 for c=1:1:C {
     for i=1:1:$Length(letters) {
         set test($Extract(letters,i)_"???? ??????? ??????? ?????" _ $Extract(letters
,i)) = ""
     }
 }
 write $zh-z1,!

 //looping through test subscripts
 write "test $Order: "
 set z1=$zh
 for c=1:1:C {
     set l = ""
     for  {
         set l = $Order(test(l))
         quit:l=""
     }
 }
 write $zh-z1,!
```

```
USER>do ^testcol
1
Cache standard
test insert: 1.520673
```

```
test $Order: 2.062228
1
Cyrillic3
test insert: 3.541697
test $Order: 5.938042
1
Cyrillic4
test insert: 1.925205
test $Order: 2.834399
```

Cyrillic4 is not the default collation for rusw locale yet, but you can define your own locale based on rusw and specify Cyrillic4 as default collation for local arrays. Or you can set Cyrillic4 as the default new collation for globals in database settings.

Cyrillic3 is slower than Caché standard and Cyrillic4 because it is based on algorithm that is more general than sorting two strings based on individual character codes.

In German language letter ß should be collated as ss during sorting. Caché respects that rule:

```
USER>write ##class(%Collate).GetLocalName()
German3
USER>set test("Straßer")=1
USER>set test("Strasser")=1
USER>set test("Straster")=1
USER>zwrite test
test("Strasser")=1
test("Straßer")=1
test("Straster")=1
```

Please notice the sorting order of strings in subscripts. Particularly, that first four letters of first string are “ Stras ” , then “ Straß ” , and then again “ Stras ” . It is impossible to sort strings in that manner if collation is just a sorting based on the codes of separate characters.

Another example is the Finnish language where ’ v ’ and ‘ w ’ should be collated as the same letter. Russian language collation rules are simpler — giving each letter some particular code and then sorting by these codes is enough. That allowed to improve performance of collation Cyrillic4 over Cyrillic3.

Collation and SQL

Don ’ t confuse collation of array with SQL collation. The latter one is conversion applied to the string before comparison or using it as a subscript in index global. Default SQL Collation in Caché is SQLUPPER. This collation converts all characters to uppercase, removes space characters and adds one space at the beginning of the string. Other SQL Collations (EXACT, SQLSTRING, TRUNCATE) are described in the documentation.

It ’ s easy to mess things up when different globals in the same database have different collation, and local arrays have other collation. SQL uses CACHETEMP database for temporary data. Default collation for globals in CACHETEMP might be different from collation for Caché installation locale.

There is one main rule — for ORDER BY in SQL queries to return rows in expected order, collation of globals where data and indexes of relevant tables are stored should be the same as the default collation of CACHETEMP database and collation of local arrays. For more details — see the paragraph in documentation “ SQL and NLS Collations ” .

Let ’ s create test class:

```
Class Collation.test Extends %Persistent
{

Property Name As %String;

Property Surname As %String;

Index SurInd On Surname;

ClassMethod populate()
{
    do ..%KillExtent()

    set t = ..%New()
    set t.Name = "?????", t.Surname = "?????"
    write t.%Save()

    set t = ..%New()
    set t.Name = "????", t.Surname = "??????"
    write t.%Save()

    set t = ..%New()
    set t.Name = "??????", t.Surname = "???????????"
    write t.%Save()
}

}
```

Populate class with data (later you can try to use words from the previous example with the German language):

```
USER>do ##class(Collation.test).populate()
```

Run the query:

```
select name from collation.test order by name
```

Row count: **3** Performance: **0.002** seconds **60** global references Cached Query: %sqlcq.USER.cls1 Last update: 2016-05-1

Name
Пётр
Павел
Прохор

3 row(s) affected

That is the unexpected result. The main question is why names are not ordered alphabetically? (Павел, Пётр, Прохор)? Let ' s look at query plan:

Key words in this plan are " populates temp-file " . SQL engine decided to use temporary structure to run this query. Although called " file " , really this is process-private global and in some cases local array. Subscripts of this global are values to order by, in this particular case — person names. Process-private globals are stored in CACHETEMP database and default collation for new globals in CACHETEMP is Caché standard.

Another reasonable question is why ' ё ' is returned at the top and not at the bottom (remember, in Caché Standard ' ё ' is sorted after all Russian letters and ' Ё ' — before). Subscripts of temporary global are not exact value of Name field, but uppercased values of Name (SQLUPPER is default SQL collation for strings), and therefore ' Ё ' is returned before other characters.

Modifying default collation using %Exact function, we would receive still incorrect, but at least expected result with ' ё ' sorted after other letters.

For now, let ' s not change default collation of CACHETEMP — let ' s check queries with Surname column. Index on this column is stored in ^Collation.testI global. Collation of that global is Cyrillic3, so we should see correct row order:

Wrong again — ' Ё ' should go between ' А ' and ' И ' . Look at the query plan:

Index data is not enough to output original values of Surname field because SQLUPPER is applied to values in SurInd index. SQL Engine decided to use values from the table itself and sort values in temporary file, just like it did before with Name column.

We can state in the query that we are OK with surnames in uppercase. The order will be correct because rows will be taken directly from index global ^Collation.testI:

Query plan is as expected:

Now let ' s do what we should have been done long ago — change default collation of CACHETEMP database to Cyrillic3 (or Cyrillic4).

Queries that use temporary files will output rows in correct order:

Summary

- If you don ' t care for nuances of local alphabets — use collation Caché standard.
- Some collations (Cyrillic4) have better performance than others (Cyrillic3).
- Check that CACHETEMP has the same collation as your main database and local arrays.

#Best Practices #Caché #Globals #ObjectScript #SQL