Artículo
Ricardo Paiva · 13 mayo, 2021 Lectura de 13 min

Compilaciones en InterSystems IRIS

El orden es una necesidad para todos, pero no todos lo entienden de la misma manera (Fausto Cercignani)

Nota de traducción: este articulo se publicó originalmente basado en Caché. Para esta traducción se revisó todo el contenido usando InterSystems IRIS.

*Descargo de responsabilidad: En este artículo se utilizan el ruso y el alfabeto cirílico como ejemplos, pero esto solamente es importante para las personas que utilizan InterSystems IRIS en lugares en los que no se habla inglés. Ten en cuenta que este artículo se refiere principalmente a las compilaciones de NLS, que son diferentes a las compilaciones de SQL. Las compilaciones en SQL (por ejemplo, SQLUPPER, SQLSTRING, EXACT que significa que no hay compilación, TRUNCATE, etc.) son funciones reales que se aplican de manera explícita a algunos valores, y cuyos resultados a veces se almacenan explícitamente en los subíndices de los *globals*.* De hecho, cuando se almacenan en subíndices, de forma natural estos valores seguirían la compilación de NLS ("Compilaciones de SQL y NLS ").*

En InterSystems IRIS, todo se almacena en *globals*: datos, metadatos, clases, rutinas. Los *globals* son persistentes. Los nodos de los *globals* se ordenan por los valores de los subíndices, y son almacenados en dispositivos de almacenamiento, sin un orden establecido de inserción, pero de forma ordenada para un mejor resultado en las búsquedas y 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("??")=""
```

Durante el proceso de clasificación, InterSystems IRIS distingue números y cadenas - 2 se maneja como un número y se clasifica antes que 10. El comando ZWrite y las funciones $Order y $Query devuelven subíndices en el mismo orden en que se almacenan estos subíndices: primero se almacena una cadena vacía (la cual no se puede utilizar como subíndice), después los números negativos, cero, los números positivos, y luego las cadenas en el orden que se define en la compilación (compilación).

La compilación estándar en InterSystems IRIS se llama (como era de esperar) IRIS standard, y clasifica cada cadena de acuerdo a sus códigos de caracteres Unicode.

La compilación de las matrices locales en el proceso actual se define en local (Management Portal > System administration > Configuration > System Configuration > National Language Settings > Locale Definitions). La configuración local rusa para las instalaciones Unicode en InterSystems IRIS es rusw y la compilación predeterminada para rusw es Cyrillic3. Otras posibles compilaciones en la rusw local son IRIS standard, Cyrillic1, Cyrillic3, Cyrillic4, Ukrainian1.

ClassMethod ##class(%Collate).SetLocalName() configura la compilación de las matrices locales en el proceso actual:

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

Por cada compilación, hay una compilación alterna que clasifica los números como cadenas. El nombre de esa compilación contiene la variable " string " al final:

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

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

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

IRIS standar y Cyrillic3

IRIS standar clasifica los caracteres según sus códigos:

```
 write ##class(%Library.Collate).SetLocalName("IRIS 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
IRIS 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,
```

Las letras cirílicas se imprimen en el mismo orden que aparecen en el alfabeto ruso, excepto la “ ё ” y la “ Ё ” . Sus códigos de caracteres Unicode están desordenados. “ Ё ” debe compilarse entre “ Е ” y “ Д ” , y “ ё ” entre “ е ” y “ д eso la configuración local rusa necesita su propia compilación - Cyrillic3, que tiene las letras en el mismo orden que el alfabeto ruso:

```
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 utiliza un operador binario especial]] — «ordenar después». Devuelve el número 1, si el subíndice con el primer operador se ubica después del segundo operador, en caso contrario devuelve el número 0:

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

Globals y compilaciones

Globals diferentes en la misma base de datos pueden tener una compilación diferente. Cada base de datos tiene una opción de configuración - compilación predeterminada para los nuevos *globals*. Justo después de que se realiza la instalación, todas las bases de datos, excepto USER, utilizan la compilación predeterminada de IRIS standard. La compilación predeterminada para la base de datos USER se determina por la configuración local. Para rusw es Cyrillic3.

Para crear un *global* con una compilación que no sea la predeterminada para su base de datos, utiliza ##class(%GlobalEdit)##class(%GlobalEdit).Create method:

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

Hay una columna de compilación para cada *global* en la lista de *globals* en el Management Portal (System Explorer > Globals).

No es posible cambiar la compilación de los *globals* actuales. Deberías crear un *global* con una nueva compilación y copiar los datos con el comando Merge. Para realizar la conversión masiva de *globals* utiliza ##class(SYS.Database).Copy()

Cyrillic4, Cyrillic3 y umlauts

Parece ser que convertir el subíndice de una cadena a un formato interno lleva, evidentemente, más tiempo con la compilación Cyrillic3 que con la compilación IRIS estándar. Por lo tanto, insertar y buscar la matriz global (o local) con la compilación Cyrillic3 es más lento. Caché 2014.1 incluye una nueva compilación - Cyrillic4, que tiene el mismo orden correcto de letras que Cyrillic3, y un mejor rendimiento.

```
for collation="IRIS 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
IRIS 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 aún no es la compilación predeterminada para la configuración local rusw, pero puedes definir tu propia configuración local basada en rusw y especificar Cyrillic4 como compilación predeterminada para las matrices locales. O puedes configurar Cyrillic4 como la nueva compilación predeterminada para los *globals* en la configuración de la base de datos.

Cyrillic3 es más lento que IRIS estándar y Cyrillic4, porque se basa en un algoritmo más general que clasificar dos cadenas basándose en códigos de caracteres individuales.

En alemán, la letra ß debería compilarse como ss durante el proceso de clasificación. InterSystems IRIS respeta esa regla:

```
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
```

Ten en cuenta que la clasificación ordena las cadenas en subíndices. Específicamente, las primeras cuatro letras de la primera cadena son “ Stras ” , después “ Straß ” , y otra vez “ Stras ” . Es imposible clasificar las cadenas de esa manera, si la compilación solo es una clasificación basada en los códigos de caracteres separados.

Otro ejemplo es el finlandés, donde “ v ” y “ w ” deben compilarse como la misma letra. Las reglas para compilar el idioma ruso son más simples: a cada letra se le asigna un código específico y la clasificación mediante estos códigos es suficiente. Eso permitió mejorar el rendimiento de la compilación Cyrillic4 sobre Cyrillic3.

Compilación y SQL

No hay que confundir la compilación de una matriz con la compilación de SQL. Esta última es la conversión que se implementa en la cadena antes de que realice una comparación, o que se utilice como subíndice en el índice global. SQLUPPER es la compilación predeterminada de SQL en InterSystems IRIS. En esta compilación todos los caracteres se convierten en mayúsculas, se eliminan los caracteres de espacio y se añade un espacio al inicio de la cadena. Otras compilaciones SQL (EXACT, SQLSTRING, TRUNCATE) se describen en la documentación.

Es fácil complicar las cosas cuando diferentes *globals* en la misma base de datos tienen diferente compilación, y las matrices locales tienen otra compilación. SQL utiliza la base de datos IRISTEMP para los datos temporales. La compilación predeterminada para los *globals* en IRISTEMP puede ser diferente de la compilación para la configuración local de InterSystems IRIS.

Hay una regla principal - para que las consultas ORDER BY en SQL devuelvan las filas en el orden esperado, la compilación de los *globals* donde se almacenan los datos y los índices de las tablas relevantes deben ser la misma que la compilación predeterminada de la base de datos IRISTEMP y la compilación de las matrices locales. Para obtener más información, consulta el párrafo en la documentación "Compilaciones de SQL y NLS" .

Vamos a crear la clase de prueba:

```
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()
}

}
```

Añade datos a la clase (más tarde puedes intentar utilizar las palabras del ejemplo anterior con el alemán):

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

Realiza la consulta:

```
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

Ese es el resultado inesperado. La pregunta principal es: ¿por qué los nombres no se ordenan alfabéticamente? (Павел, Пётр, Прохор)? Veamos el plan de consultas:

Las palabras clave de este plan son “ populates temp-file ” . El motor SQL decidió utilizar una estructura temporal para realizar esta consulta. Aunque se llama "file" ("archivo"), en realidad se trata de un *global* process-private y en algunos casos es una matriz local. Con los valores de los subíndices de este *global* se ordenarán, en este caso particular, nombres de personas. Los *globals* process-private se almacenan en la base de datos IRISTEMP y la clasificación predeterminada para los nuevas *globals* en IRISTEMP es IRIS estándar.

Otra pregunta razonable es por qué “ ё ” se devuelve a la parte superior y no a la inferior (recuerda, en IRIS estándar “ ё ” se clasifica después de todas las letras rusas; y “ Ё ” , antes). Los subíndices *global* temporal no son el valor exacto del campo Name, sino los valores en mayúsculas de Name (SQLUPPER es la compilación de SQL predeterminada para las cadenas), y, por tanto, devuelve “ Ё ” antes que otros caracteres.

Al modificar la compilación predeterminada mediante la función %Exact, todavía la recibiríamos de forma incorrecta, pero al menos “ ё ” se clasifica después de otras letras, como un resultado esperado.

Por el momento, no vamos a cambiar la compilación predeterminada de IRISTEMP - vamos a verificar las consultas con la columna Surname. El índice de esta columna se almacena en el *global* ^Collation.testI. La compilación de ese *global* es Cyrillic3, así que deberíamos ver el orden correcto de las filas:

Equivocado de nuevo — “ Ё ” debería ir entre “ А ” y “ И ” . Echa un vistazo al plan de consulta:

Los datos del índice no son suficientes para obtener los valores originales del campo Surname porque SQLUPPER se aplica a los valores del índice SurInd. El motor SQL decidió utilizar los valores de la propia tabla y ordenar los valores en el archivo temporal, tal como hizo antes con la columna Name.

En la consulta podemos indicar que estamos de acuerdo con que los apellidos se escriban en mayúsculas. El orden será el correcto porque las filas se tomarán directamente del índice global ^Collation.testI:

El plan de consulta es como se esperaba:

Ahora hagamos lo que deberíamos haber hecho hace mucho tiempo: cambiar la compilación predeterminada de la base de datos IRISTEMP a Cyrillic3 (o Cyrillic4).

Las consultas que utilizan archivos temporales darán como resultado filas en el orden correcto:

Resumen

- Si no te interesan los matices de los alfabetos locales, utiliza la compilación IRIS estándar.
- Algunas compilaciones (Cyrillic4) tienen un mejor rendimiento que otras (Cyrillic3).
- Comprueba que IRISTEMP tiene la misma compilación que tu base de datos principal y matrices locales. Nota de traducción: para ver una lista con las collations disponibles y las cargadas en la instancia, abre una ventana de Terminal, cambia al namespace % SYS% e ingresa el comando DO ^ COLLATE.

#Globals #ObjectScript #SQL #Caché #InterSystems IRIS