Article
Brendan Bannon · Jul 15, 2021 6m read

Embedded SQL new in InterSystems IRIS

Benjamin De Boe wrote this great article about Universal Cached Queries, but what the heck is a Universal Cached Query (UCQ) and why should I care about it if I am writing good old embedded SQL? In Caché and Ensemble, Cached Queries would be generated to resolve xDBC and Dynamic SQL. Now in InterSystems IRIS embedded SQL has been updated to use Cached Queries, hence the Universal added to the name. Now any SQL executed on IRIS will be done so from a UCQ class.

Why did InterSystems do this? Good Question! The big win here is flexibility in a live environment. In the past, if you add an index or ran TuneTable, the SQL in Cached Queries would make use of this new information right away while embedded SQL would remain unchanged until the class or routine was compiled manually. If your application used deployed classes or only shipped OBJ code, recompiling on the customer system was not an option. Now all SQL statements on a system will be using the latest class def. and the latest tuning data available. In the future, InterSystems IRIS will have optional tools that can monitor and tune your production systems on a nightly basis customizing the SQL plans based on how the tables are being queried. As this toolset grows the power of the Universal Cached Query will grow as well.

Is my embedded SQL slower now? Yes and no. Calling a tag in a different routine is a little more expensive than calling a tag in the same routine, so that is slower, but UCQ code generation was different from embedded, and getting to use those changes more than makes up for the expense of calling a different routine. Are there cases where the UCQ code is slower? Yes, but overall you should see better performance. I am an embedded SQL guy from way back so I always like to point out that Embedded SQL is faster than Dynamic SQL. It still is faster, but with all the work that has been done to make objects faster the margin between the 2 styles is small enough that I will not make fun of you for using dynamic SQL.

How do I check for errors now? Error handling for Embedded SQL has not changed. SQLCODE will be set to a negative number if we hit an error and %msg will be set to the description of that error. What has changed are the types of errors you can get. The default behavior now is that the SQL will not be compiled until the first time the query is run. This means if you misspell a field or table in the routine the error will not get reported when you compile that routine, it will be reported the first time you execute the SQL, same as dynamic SQL. SQLCODE is set for every SQL command but if you are lazy like me you only ever check SQLCODE after a FETCH. Well, now you might want to start checking on the OPEN as well.

```
&SQL(DECLARE cur CURSOR FOR
SELECT Name,junk
into :var1, :var2
FROM Sample.Person)
&SQL(OPEN cur)
write !,"Open Status: ",SQLCODE,?20,$G(%msg)
for  {
        &SQL(FETCH cur)
        write !,"Fecth Status: ",SQLCODE,?20,$G(%msg)
        QUIT:SQLCODE'=0
        w !,var1
}
&SQL(CLOSE cur)
```

```
        write !,"Close Status: ",SQLCODE,?20,$G(%msg)
        QUIT
```

In the code above I have an invalid field in the SELECT. Because we do not compile the SQL when we compile the routine this error is not reported. When I execute the code the OPEN reports the compile error while the FETCH and CLOSE report a cursor not open error. %msg does not get changed so if you check that at any point you will get helpful info:

```
USER>d ^Embedded

Open Status:  -29    Field 'JUNK' not found in the applicable tables^DECLARE cur CURSOR FOR SELECT Name , junk INTO compiling embedded cached query from Embedded.mac

Fetch Status:  -102  Field 'JUNK' not found in the applicable tables^DECLARE cur CURSOR FOR SELECT Name , junk INTO compiling embedded cached query from Embedded.mac

Close Status:  -102  Field 'JUNK' not found in the applicable tables^DECLARE cur CURSOR FOR SELECT Name , junk INTO compiling embedded cached query from Embedded.mac
```

What if I don ' t want my embedded SQL to change? You can still do this using Frozen Query Plans. A quick side note, every major IRIS upgrade you do will freeze all SQL Statements so nothing will change if you don ' t let it. You can read more about that here.

Now back to dealing with UCQ stuff. Here are 3 ways you could freeze embedded SQL plans in your application:

1. If you ship an IRIS.DAT:
    a. Do $SYSTEM.OBJ. GenerateEmbedded() to generate the UTC for embedded SQL
    b. Freeze the plans: do $SYSTEM.SQL.Statement.FreezeAll()
    c. Ship the IRIS.DAT
2. If you use xml files:
    a. Do $SYSTEM.OBJ. GenerateEmbedded() to generate the UTC for embedded SQL
    b. Freeze the plans: do $SYSTEM.SQL.Statement.FreezeAll()
    c. Export the frozen plans: do $SYSTEM.SQL.Statement.ExportAllFrozenPlans()
    d. After loading your application, load the frozen plans: do $SYSTEM.SQL.Statement.ImportFrozenPlans()
3. Freeze UTC Plans on the customer site:
    a. Load the code with embedded SQL on the customer system
    b. Do $SYSTEM.OBJ. GenerateEmbedded() to generate the UTC for embedded SQL
    c. Freeze all the plans that got generated: do $SYSTEM.SQL.Statement.FreezeAll()

Can I go back to the old behavior? Nope, this is the way it is now. From a developer's point of view, you can get the old behavior back by adding the flag /compileembedded=1 to your compiler options. This will tell the compiler to generate the UCQ class while compiling the class or routine. If there is an issue with the SQL it will be reported at compile time as it did in the past.

```
Compiling routine : Embedded.mac
ERROR: Embedded.mac(5) : SQLCODE=-29 : Field 'JUNK' not found in the applicable tables^DECLARE cur CURSOR FOR SELECT Name , junk INTO compiling embedded cached query from Embedded.mac
Detected 1 errors during compilation in 0.034s.
```

If you are concerned about the overhead of generating the UCQ classes the first time embedded SQL is run you could add this step as part of your application install to generate them all in advance: do $SYSTEM.OBJ. GenerateEmbedded()

This is a very high-level overview of Universal Cached Queries and Embedded SQL. I did not get into any of the real details about what happens under the covers. I just tried to talk about stuff people would run into as they work with Embedded SQL on IRIS. Overall moving to UCQ should make SQL performance more consistent across all types of SQL and it should make updating SQL on a production system easier. There will be some adjustments. Adding the compiler flag will be a big help for me. Now I just need to get used to looking for the generated code in a new place. If you have any questions, comments, concerns about this, or anything else related to SQL on InterSystems IRIS please let me know.

#SQL #InterSystems IRIS