Article
Matthew Giesmann · Jan 28, 2022 2m read

# Embedded SQL Bug and Workaround

We recently encountered an Embedded SQL issue while upgrading to IRIS 2021.1, and thought the issue and workaround might be interesting to share.

**Key takeaway**: Host variables in an ORDER BY clause of an embedded SQL query that is inside of a method don't work as expected. IRIS versions starting with 2020.1 are affected. As a workaround, add the host variable to the Method's PublicList list and "new" them so the embedded query has access to them.

Consider this data and following embedded query:

| id | Foo |
|---|---|
| 1 | bar |
| 2 | baz |

```
ClassMethod Run() {
set filter = "baz"
&SQL(SELECT TOP 1 Foo INTO :foo FROM Test.Items ORDER BY CASE Foo WHEN :filter THEN 0
 ELSE 1 END)
write !,"foo=",foo  // unexpectedly prints "foo=bar"
```

If we run this query with a filter of 'baz', we expect that row 2 should ORDER before row 1, and therefore the TOP 1 result will return with foo=baz.  We get the wrong result, so what's going on?

Because the embedded SQL is running in a method, the host variables should be made available to the generated query code - but we found that host variables in the ORDER BY clause are not.

This is clear by looking at "view other code" to see the compiled code for the method in studio. Note "foo" is in the public list, but "filter" is not

```
zRun() [ SQLCODE,foo ] public { New SQLCODE,foo
```

While a fix will be on its way, we can work around this issue by adding the host variable to the public list manually. Variables in the public list are visible to any invoked methods (in this case, the query generated from the embedded SQL):

```
ClassMethod Run() [ PublicList = filter ]
{
new filter
set filter = "baz"
&SQL(SELECT TOP 1 Foo INTO :foo FROM Test.Items ORDER BY CASE Foo WHEN :filter THEN 0
 ELSE 1 END)
write !,"foo=",foo  // prints "foo=baz" as expected
```

Our query returns the expected result, and if we check the compiled code for the method, we now see the correct public list:

```
zRun() [ filter,SQLCODE,foo ] public { New SQLCODE,foo
new filter
```

#SQL #InterSystems IRIS

**Source URL:**https://community.intersystems.com/post/embedded-sql-bug-and-workaround