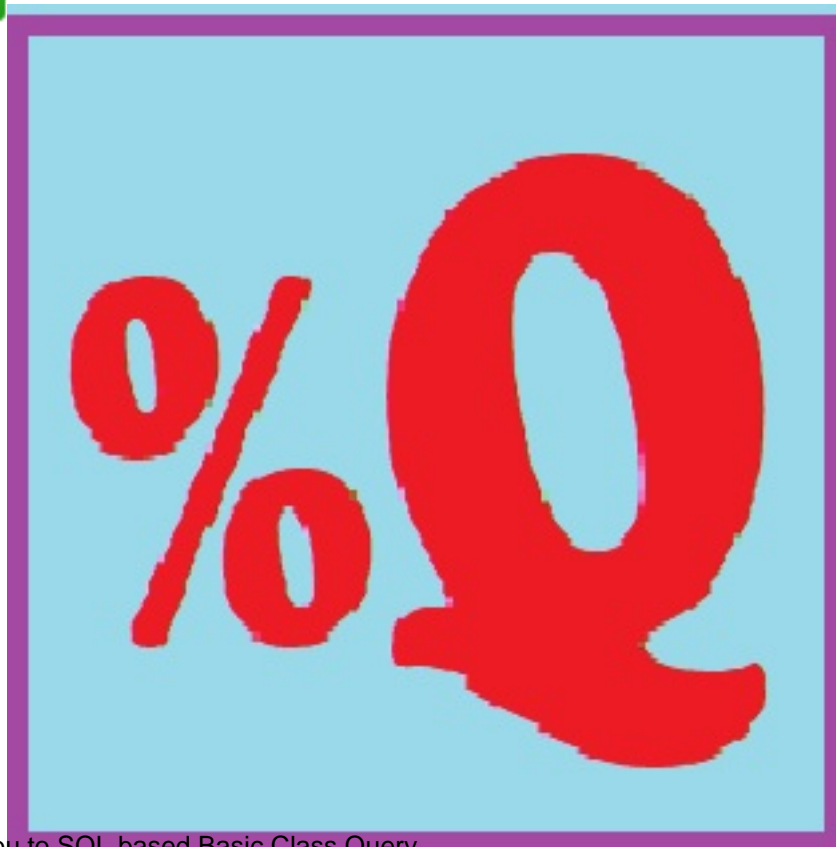
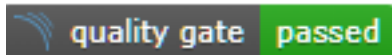


Article

[Robert Cemper](#) · Mar 2, 2023 5m read

[Open Exchange](#)

Tutorial - Working with %Query #2



My previous article introduced you to SQL based Basic Class Query where a clever wizard created all the required code for you and your essential contribution was an SQL statement.

Now we enter the real Custom Class Query that provides more freedom but requires a deeper understanding of the mechanic behind the scene. The full code example is again on [GitHub](#)

Some things haven't changed:

- demo data are the same
- consumption of the query is also unchanged
- all handling of ODBC / JDBC protocol is still generated.

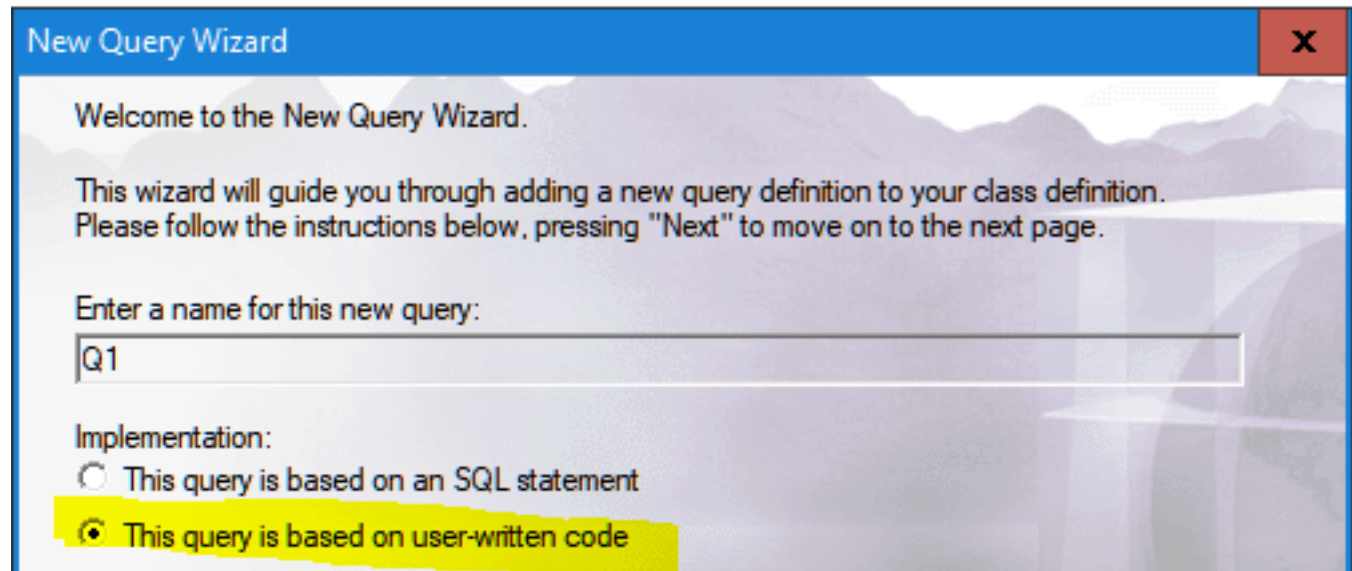
So what is different?

- You need a header QUERY statement to declare your input parameters but
 - also the record layout for your output. ROWSPEC
- you have to provide an Execute method to initialize your query and consume your input parameters
- a Close method to clean up your environment
- and a Fetch method that does the Job.

- it is called row by row until you set AtEnd=1. (it is passed by reference)
- and you return variable Row (also passed by reference) as \$LB() structure
- so it is obvious that the resulting Row is shorter than MAXSTRING
- and this is our challenge to present your stream

so we take our choice:

as before we use the Studio's Query Wizard



New Query Wizard

Welcome to the New Query Wizard.

This wizard will guide you through adding a new query definition to your class definition. Please follow the instructions below, pressing "Next" to move on to the next page.

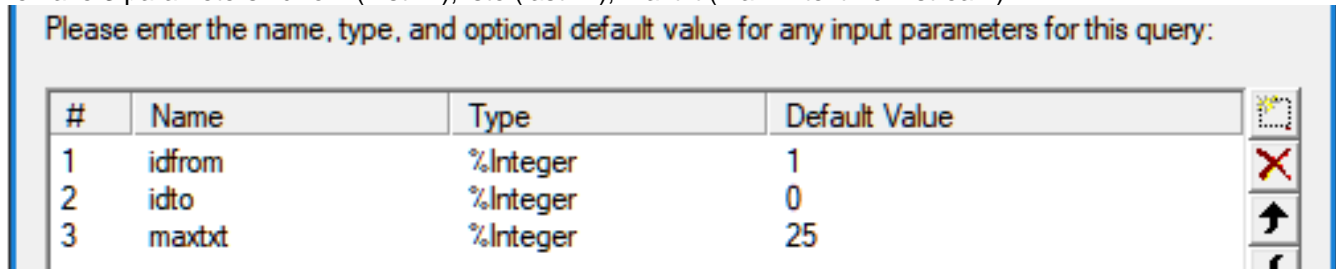
Enter a name for this new query:

Implementation:

☐ This query is based on an SQL statement

☒ This query is based on user-written code

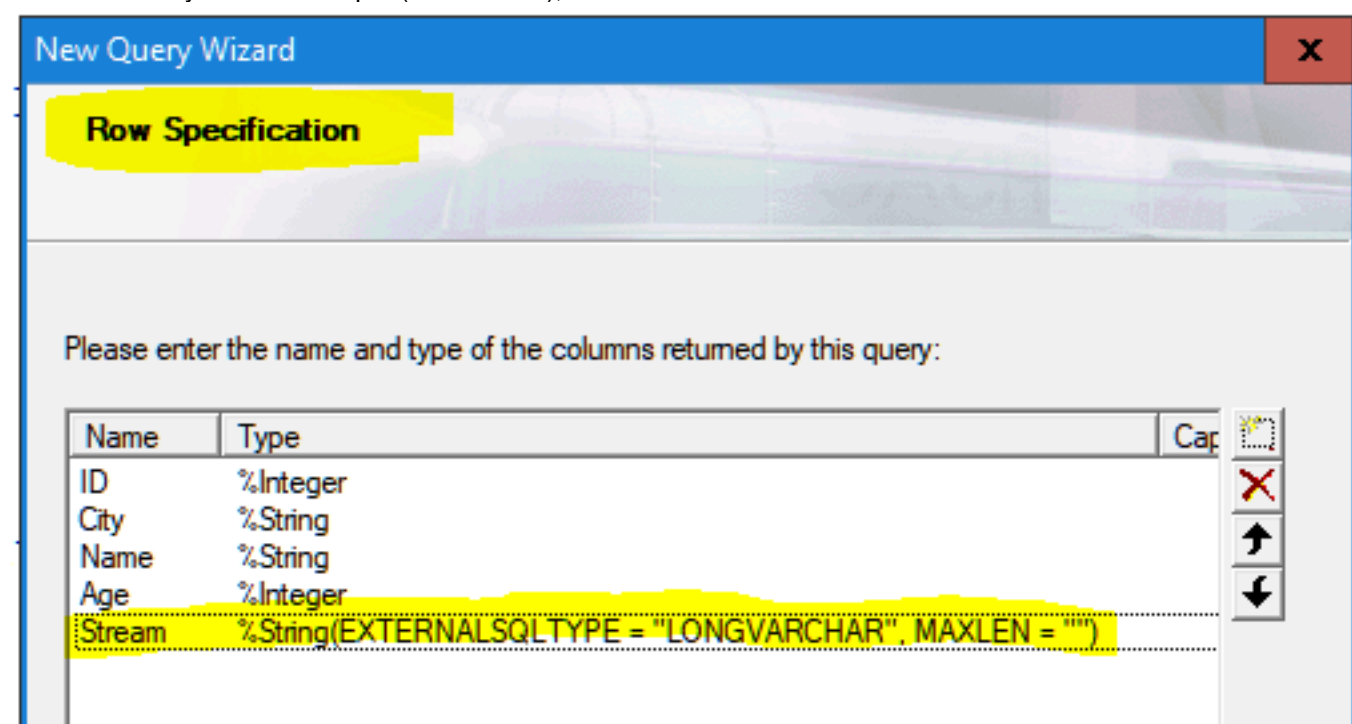
we have 3 parameters: idfrom (first ID), isto (last ID), maxtxt (maxim text from stream)



Please enter the name, type, and optional default value for any input parameters for this query:

#	Name	Type	Default Value
1	idfrom	%Integer	1
2	idto	%Integer	0
3	maxtxt	%Integer	25

and new the layout of our output (ROWSPEC), similar to above



Row Specification

Please enter the name and type of the columns returned by this query:

Name	Type	Cap
ID	%Integer	
City	%String	
Name	%String	
Age	%Integer	
Stream	%String(EXTERNALSQLTYPE = "LONGVARCHAR", MAXLEN = "")	

for our Stream, we need to overwrite %String defaults to match ODBC / JDBC

The type needs to be %String(EXTERNALSQLTYPE = "LONGVARCHAR", MAXLEN = "")

This is the generated code framework:

```
Query Q1(
    idfrom As %Integer = 1,
    idto As %Integer = 0,
    maxtxt As %Integer = 25) As %Query
    (ROWSPEC = "ID:%Integer,City:%String,
        Name:%String,Age:%Integer,
        Stream:%String(EXTERNALSQLTYPE=""LONGVARCHAR"", MAXLEN = """""))
{
}
ClassMethod Q1Execute(
    ByRef qHandle As %Binary,
    idfrom As %Integer = 1,
    idto As %Integer = 0,
    maxtxt As %Integer = 25) As %Status
{
    Quit $$$OK
}
ClassMethod Q1Close(ByRef qHandle As %Binary) As %Status [ PlaceAfter = Q1Execute ]
{
    Quit $$$OK
}
ClassMethod Q1Fetch(
    ByRef qHandle As %Binary,
    ByRef Row As %List,
    ByRef AtEnd As %Integer = 0) As %Status [ PlaceAfter = Q1Execute ]
{
    Quit $$$OK
}
```

we still need to add CONTAINID=1 and [SqlName = Q1, SqlProc]

- Query Q1() is the descriptor used by the interface support
- qHandle is the common data structure for all 3 methods
 - whatever you pass along to the next row for processing needs to be stored there.
 - e.g. first ID, last ID, actual ID,
 - In past this was typically a subscripted variable or some oref
 - As a personal experiment, I tried a JSON object and it worked fine
- The query takes 3 simple input parameters:
 - idfrom = first ID to show
 - idto = last ID to show - missing shos al higher IDs available
 - maxtxt = maximum text from the beginning of the stream. Default = 25
-

```
ClassMethod Q1Execute(
    ByRef qHandle As %Binary,
    idfrom As %Integer = 1,
    idto As %Integer = 0,
    maxtxt As %Integer = 25) As %Status
{
    set qHandle={}
    set qHandle.id=0
    set qHandle.idfrom=idfrom
    set qHandle.idto=idto
    set qHandle.obj=0
```

```

    set qHandle.stream=0
    set qHandle.maxtxt=maxtxt
    Quit $$$OK
}

```

- Q1Fetch is the biggest working bloc
 - I used object access in this example to keep it more readable
 - Accessing the Globals for Date and Stream directly was remarkably faster but really hard to read and to follow.
 - The more important point is that you can do whatever you like, not just collect or select data.
 - Many management routines use it to display Processes, actual Users, ... whatever can be presented as a table.
 - The point is to compose the \$LB() for Row and return it and once you are done,
 - Set AtEnd=1, and the query terminates.
 - In this example, the major challenge is to skip nonexistent objects and to skip no existing streams
 - to avoid empty result lines.

```

• /// that's where the music plays
  /// called for every row delivered
  ClassMethod Q1Fetch(
    ByRef qHandle As %Binary,
    ByRef Row As %List,
    ByRef AtEnd As %Integer = 0) As %Status [ PlaceAfter = Q1Execute ]
  {
    /// first access
    if qHandle.id<qHandle.idfrom set qHandle.id=qHandle.idfrom
    ///
  nextrec
    if qHandle.idto,qHandle.idto<qHandle.id set AtEnd=1
    if qHandle.id>^rcc.TUD set AtEnd=1
    if AtEnd quit $$$OK
    if 'qHandle.obj {
      set obj=##class(rcc.TU).%OpenId(qHandle.id)
      ,qHandle.obj=obj
      ,qHandle.stream=0
    }
    if 'obj set qHandle.id=qHandle.id+1 goto nextrec
    if 'qHandle.stream set qHandle.stream=qHandle.obj.Stream
    set text=qHandle.stream.Read(qHandle.maxtxt)
    set Row=$lb
    (qHandle.id,qHandle.obj.City,qHandle.obj.Name,qHandle.obj.Age,text)
    /// row completed
    set qHandle.id=qHandle.id+1
    set qHandle.stream=0
    set qHandle.obj=0
    Quit $$$OK
  }

```

- and here is a short test

```

[SQL]USER>>call rcc.Q1(4,7)
8.      call rcc.Q1(4,7)

```

Dumping result #1

ID	City	Name	Age	Stream
4	Newton	Evans	61	
5	Hialeah	Zemaiti	47	Resellers of premise-base
6	Elmhurs	Jenkins	29	Enabling individuals and

```
7      Islip    Drabek  61      Building shareholder valu

4 Rows(s) Affected
statement prepare time(s)/globals/lines/disk: 0.0003s/11/685/0ms
        execute time(s)/globals/lines/disk: 0.0010s/18/2156/0ms
                cached query class: %sqlcq.USER.cls59
-----
```

Getting just the beginning of our stream is not always sufficient.

Follow me on to the [next chapter](#) for the extension of this example that will show and control more result lines.

Just a reminder:

All test data are generated using the System method %Populate

So your output will be different. I suggest you run our tests also with other parameters than the shown examples to get the full power of this tool.

The full code example is again available on [GitHub](#)

The full code example is again available on [GitHub](#)

The [Video](#) is available now. See section Custom Class Query

For immediate access, you can use [Demo Server WebTerminal](#) and the related system Management Portal on [Demo Server SMP](#)

I hope you liked it so far and I can count on your votes.

[#Tutorial](#) [#Video](#) [#Other](#)

[Check the related application on InterSystems Open Exchange](#)

Source URL: <https://community.intersystems.com/post/tutorial-working-query-2>