Updated: 14th November 2023

SQL with Sage X3

Mike Shaw — 15th November 2023

Contents

Introduction

This talk seeks to explain how to find the SQL being run from the Sage X3 application point of view and interpret the log files, etc.

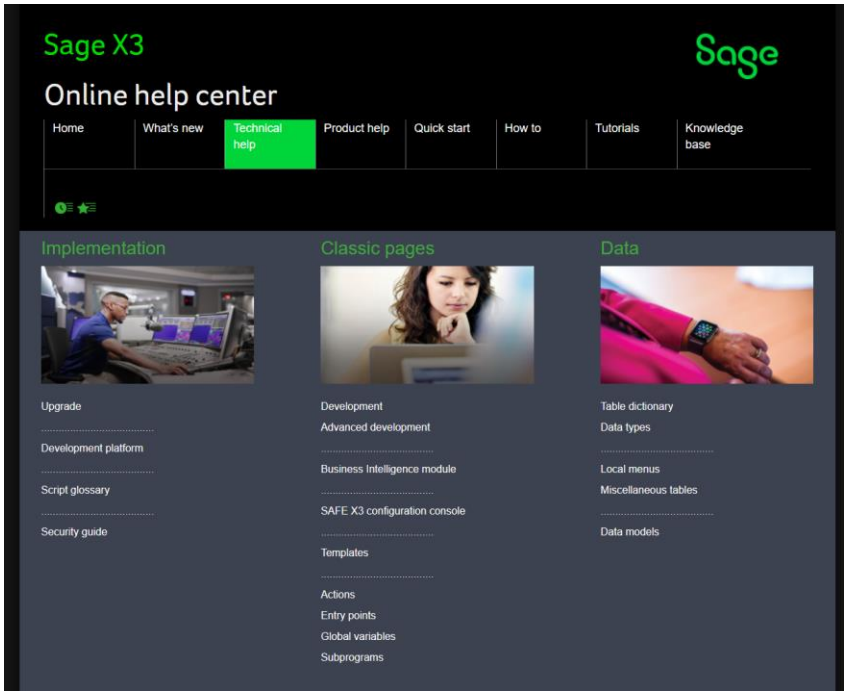
I have pre-configured my VM with some custom X3 objects for the purposes of this presentation. This setup will be provided along with these notes after the presentation, so you can reproduce the same steps as I am going to show you today.

Demonstrations

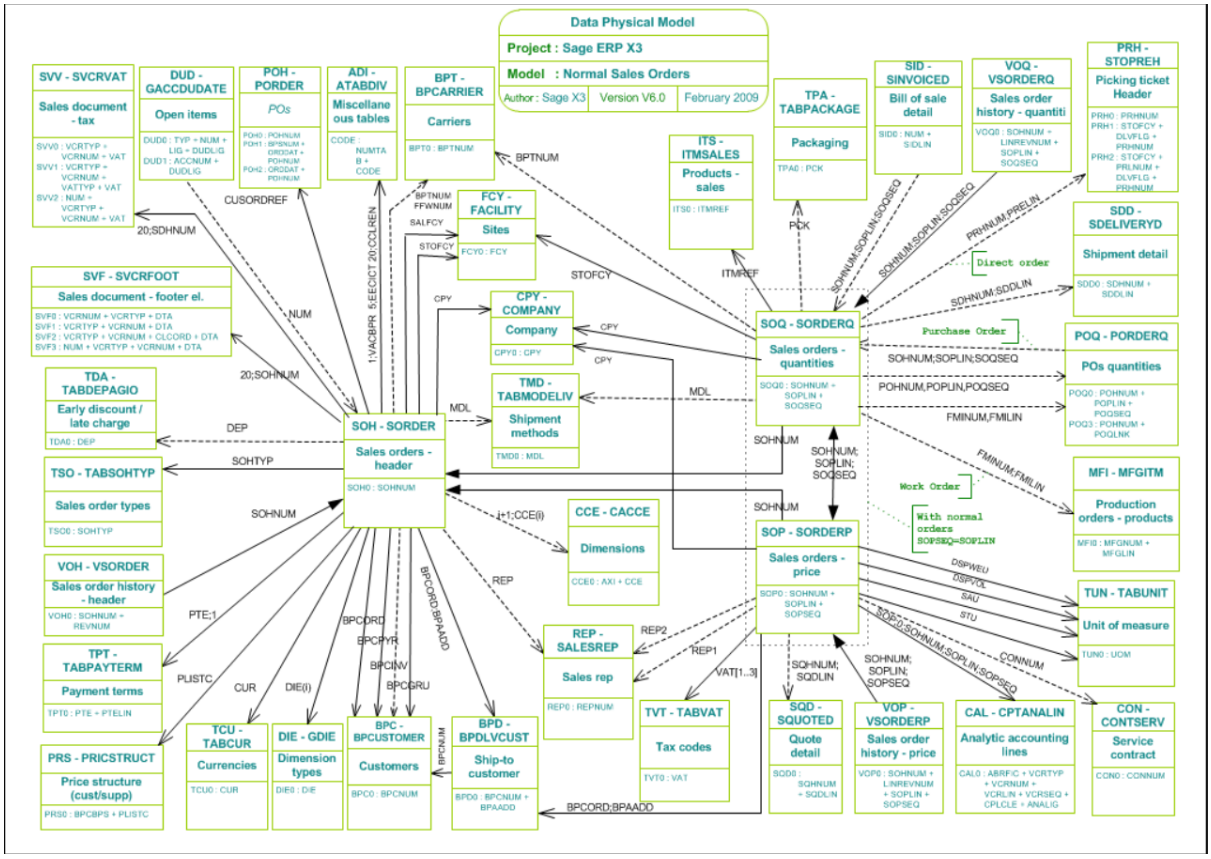
- Demonstrations and discussions

- Data models / Table definitions

You can get a better understanding of how data is stored and used in X3 by reviewing the online help for Data Models and Table Dictionary, in the Technical Help section at https://online-help.sageerpx3.com/erp/12/technical-online-help/ (WARNING: this section may not be fully updated for the latest versions)



e.g. the Normal Sales Order data model (as it was in Feb 2009 anyway…)

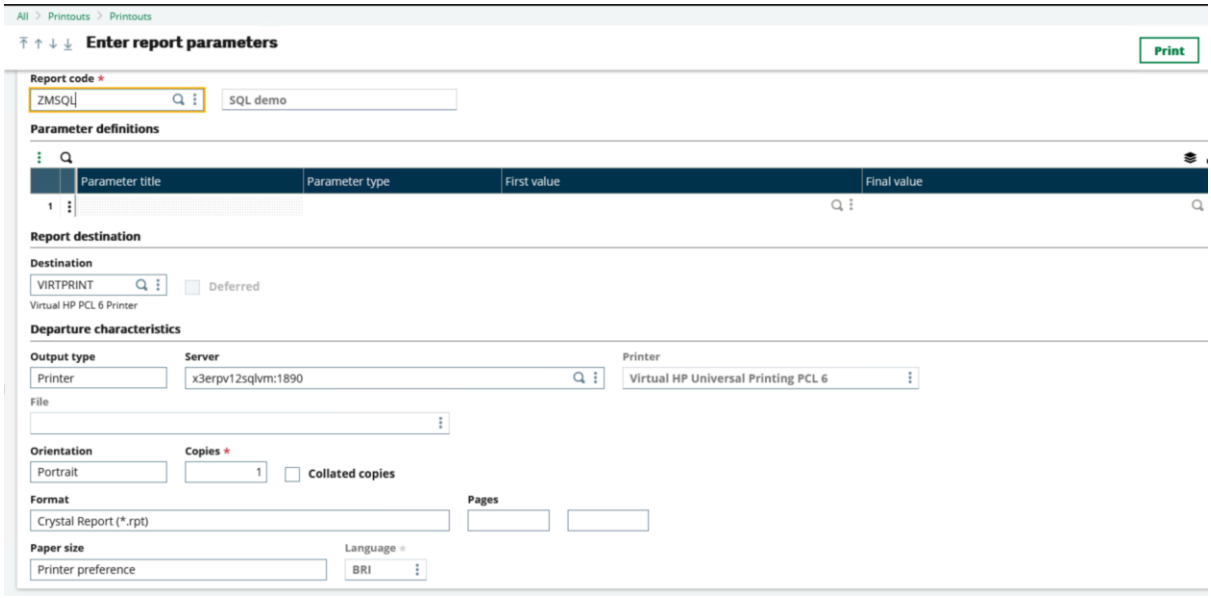


- Crystal reports

I'll start with Crystal Reports as it's relatively easy.

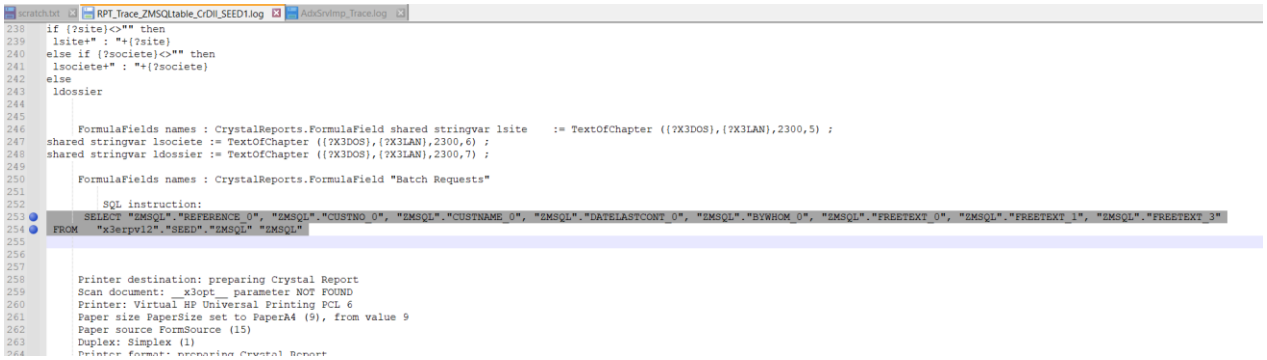
Turn on Print Server debug mode, as described in "Illustrated guide to tracing/debugging Sage X3 Report Server" (https://www.sagecity.com/gb/sage-x3/b/sage-x3-uk-support-insights/posts/illustrated-guide-to-tracing-debugging-sage-x3-report-server)

Then run the report in question, for me this is report code "ZMSQL". The crystal report definition file is included in the training documentation bundle.



This generates a couple of files for us to review, which I've included in "..\SQLcourse\ReportServer".

In the "RPT_Trace_ZMSQLtable_CrDll_SEED1.log" file we can find the exact SQL being run by the report. Crystal Reports always creates one big SQL statement to extract all the needed data to run the report.



```
if {?site}<>"" then
 lsite+" : "+{?site}
else if {?societe}<>"" then
 lsociete+" : "+{?societe}
else
 ldossier

     FormulaFields names : CrystalReports.FormulaField shared stringvar lsite    := TextOfChapter ({?X3DOS},{?X3LAN},2300,5) ;
shared stringvar lsociete := TextOfChapter ({?X3DOS},{?X3LAN},2300,6) ;
shared stringvar ldossier := TextOfChapter ({?X3DOS},{?X3LAN},2300,7) ;

     FormulaFields names : CrystalReports.FormulaField "Batch Requests"

        SQL instruction:
    SELECT "EMSQL"."REFERENCE_0", "EMSQL"."CUSTNO_0", "EMSQL"."CUSTNAME_0", "EMSQL"."DATELASTCONT_0", "EMSQL"."BYWHOM_0", "EMSQL"."FREETEXT_0", "EMSQL"."FREETEXT_1", "EMSQL"."FREETEXT_3"
FROM   "x3erpv12"."SEED"."EMSQL" "EMSQL"


     Printer destination: preparing Crystal Report
     Scan document: __x3opt__ parameter NOT FOUND
     Printer: Virtual HP Universal Printing PCL 6
     Paper size PaperSize set to PaperA4 (9), from value 9
     Paper source FormSource (15)
     Duplex: Simplex (1)
```

If we look at a more complex report log "RPT_Trace_ADOVAL1.log", we can see that we still get the full SQL, with all parameters explicitly used.

```
     FormulaFields names : CrystalReports.FormulaField shared stringvar var3 ;
var3

     FormulaFields names : CrystalReports.FormulaField shared stringvar lsite ;
shared stringvar lsociete ;
shared stringvar ldossier ;
if {?site}<>"" then
 lsite+" : "+{?site}
else if {?societe}<>"" then
 lsociete+" : "+{?societe}
else
 ldossier

     FormulaFields names : CrystalReports.FormulaField shared stringvar lsite    := TextOfChapter ({?X3DOS},{?X3LAN},2300,5) ;
shared stringvar lsociete := TextOfChapter ({?X3DOS},{?X3LAN},2300,6) ;
shared stringvar ldossier := TextOfChapter ({?X3DOS},{?X3LAN},2300,7) ;

        SQL instruction:
    SELECT "ADOPAR"."CHAPITRE_0", "ADOPAR"."NOLIB_0", "ADOVAL"."VALEUR_0", "ADOVAL"."FCY_0", "ADOVAL"."CMP_0", "ADOPAR"."PARAM_0", "ATEXTE"."TEXTE_0", "ATEXTE"."LAN_0", "ADOPAR"."TYPVAL_0", "ATABDIV"."NUMTAB_0", "ATEXTRA"."LANGUE_0",
    "ATEXTRA"."TEXTE_0", "ATEXTRA"."IDENT1_0"
FROM   (((("x3erpv12"."SEED"."ADOVAL" "ADOVAL" INNER JOIN "x3erpv12"."SEED"."ADOPAR" "ADOPAR" ON "ADOVAL"."PARAM_0"="ADOPAR"."PARAM_0") LEFT OUTER JOIN "x3erpv12"."SEED"."ATABDIV" "ATABDIV" ON "ADOPAR"."CHAPITRE_0"="ATABDIV"."CODE_0") LEFT OUTER
  JOIN "x3erpv12"."SEED"."ATEXTE" "ATEXTE" ON "ADOPAR"."NUM_0"="ATEXTE"."NUMERO_0") LEFT OUTER JOIN "x3erpv12"."SEED"."ATEXTRA" "ATEXTRA" ON "ATABDIV"."CODE_0"="ATEXTRA"."IDENT2_0"
WHERE  ("ADOPAR"."CHAPITRE_0">=N'A' AND "ADOPAR"."CHAPITRE_0"<=N'I') AND "ATEXTE"."LAN_0"=N'BRI' AND "ATABDIV"."NUMTAB_0"=901 AND ("ADOVAL"."CMP_0"=N' ' AND "ADOVAL"."FCY_0"=N' ' OR "ADOVAL"."CMP_0"=N'' OR "ADOVAL"."FCY_0"=N'') AND
  "ATEXTRA"."LANGUE_0"=N'BRI' AND "ATEXTRA"."IDENT1_0"=N'901'
ORDER BY "ADOPAR"."CHAPITRE_0", "ADOPAR"."PARAM_0"


     File destination: preparing Crystal Report
     Issue file: D:\Sage\SafeX3\PrtSrv\Temp\SEED-356-20230714153629-96_
     PDF format: preparing Crystal Report
```

- Sage X3 4GL

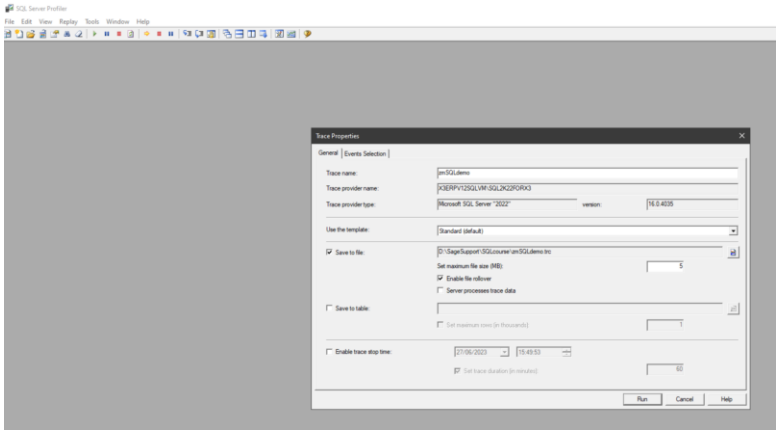
Before we look at X3 4GL code, let's take a look at a simple example run in SSMS that uses cursors and parameters, and procedures… oh and a bit of Dynamic SQL too…!!

This example will try to illustrate roughly how Sage X3 is often working, so we can understand the SQL traces we see from X3.

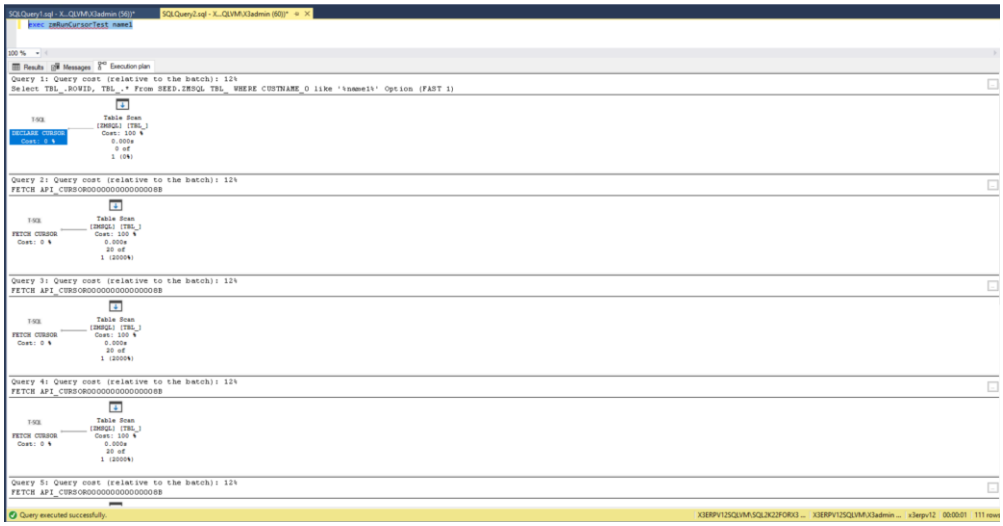
Review "CursorTest\zmProcedure.sql" and I'll provide a cursory explanation of what it is doing

Launch Tools, SQL Server profiler in SSMS

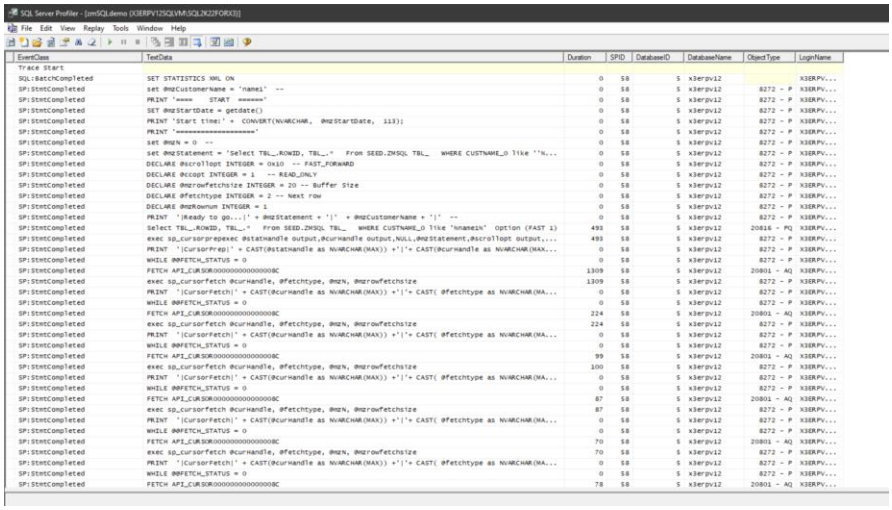
Set tracename to "zmSQLdemo", set "Tuning" template, save to file "D:\SageSupport\SQLcourse\zmSQLdemo.trc" then click "Run"



Turn on "include actual execution plan" then execute the procedure and we can see the SQL which then executes as a cursor which iterates in batches of 20 (default if not specified)



Stop the tracing, and check the SQL trace as well where we see the FETCH API_CURSOR lines



When reviewing SQL traces from Sage X3 performance issues, it can sometimes be the FETCH API_CURSOR which is shown as the top SQL running…

See my output in "..\SQLcourse\CursorTest\SQLoutputs\zmSQLdemo.trc"

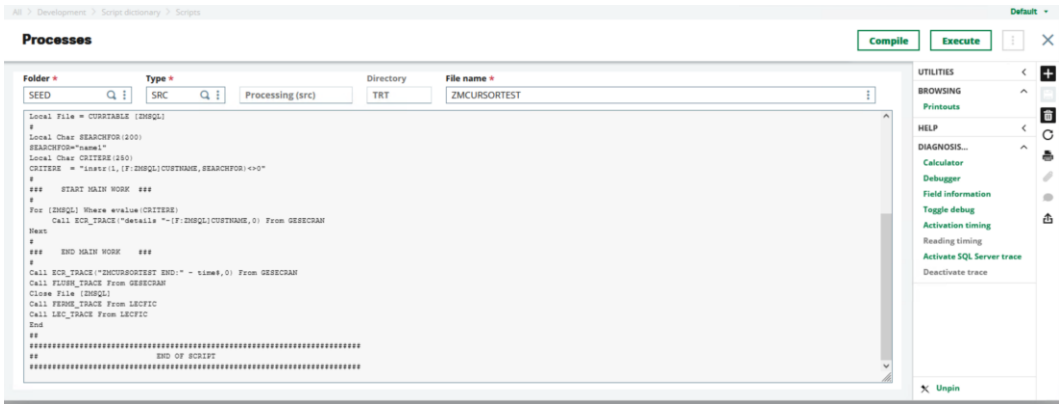
- Finding the SQL in X3

Run following SQL to show basic data:

```
select * from SEED.ZMSQL
```

I have some custom 4GL code to run a query against my custom table "ZMCURSORTEST.src".

Copy this file to "..\folders\SEED\TRT" and compile using Script Editor



Use calculator to create engine trace

openlog("TRA",15)

Activate SQL Server trace, select "Tuning" option, leave rest as default.

Execute the 4GL test case, then stop the logging

closelog()

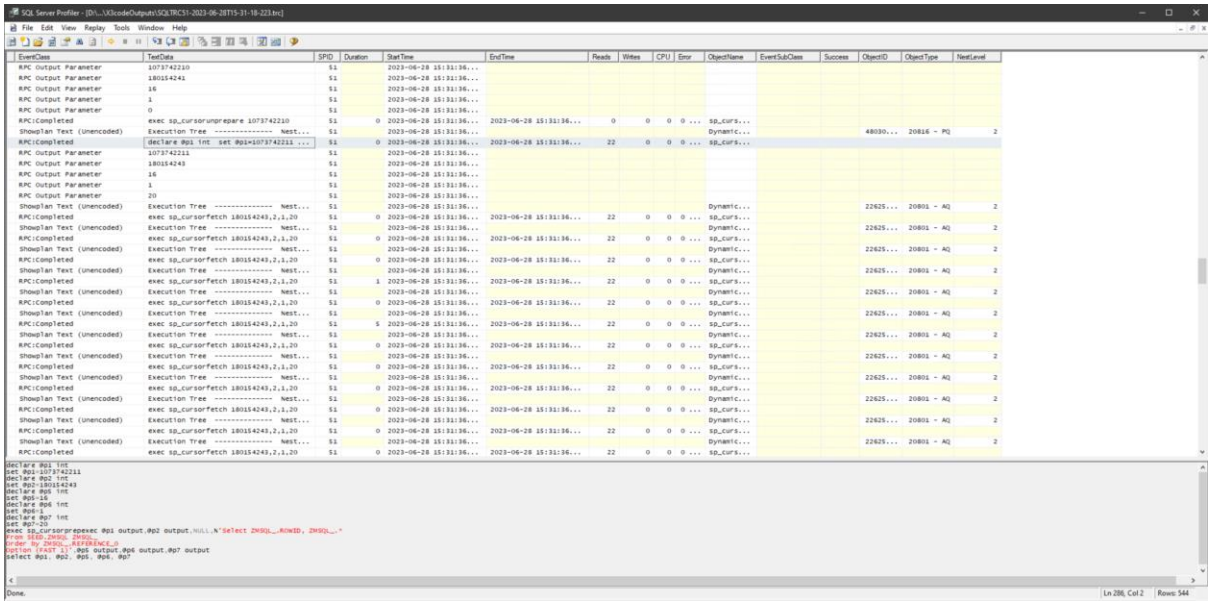
Deactivate trace

See my outputs in "..\SQLcourse\CursorTest\X3codeOutputs"

In F40059.tra we can see 111 records have been returned as expected.

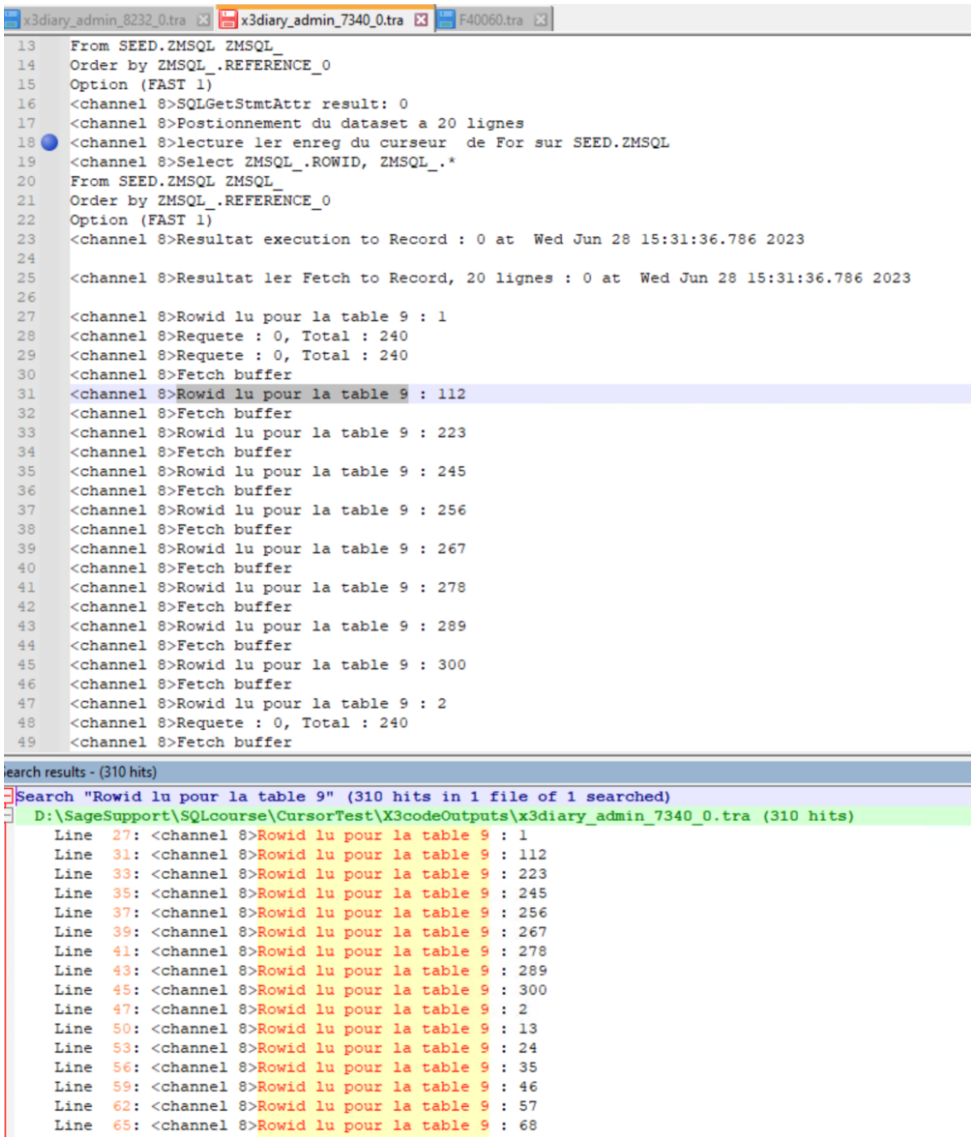
F40060.tra is the SQL trace from X3 but is a bit difficult to read.

In the trace file "SQLTRC51-2023-06-28T15-31-18-223.trc", we can see the cursor. Notice it seems to be doing 342 reads (there are 310 records in the table) however it is overreporting by 2 each time (uses 20 row cursor size):



"x3diary_admin_7340_0.tra" is the sadoss trace output and we see here the calls to SQL Server. Once we find the SQL in line 748 we can review the cursor fetches in a bit more detail… here we can see the cursor opening, the first cursor read "lecture 1er enreg du curseur de For sur SEED.ZMSQL" the fetches of data "Fetch base, nombre d enreg fetches" and finally where the cursor is closed "fermeture du curseur"

This confirms there are 310 records fetched and passed back to the adonix process to be processed "Rowid lu pour la table 9" Notice there is no WHERE clause on the SQL, which is why all 310 rows are returned…

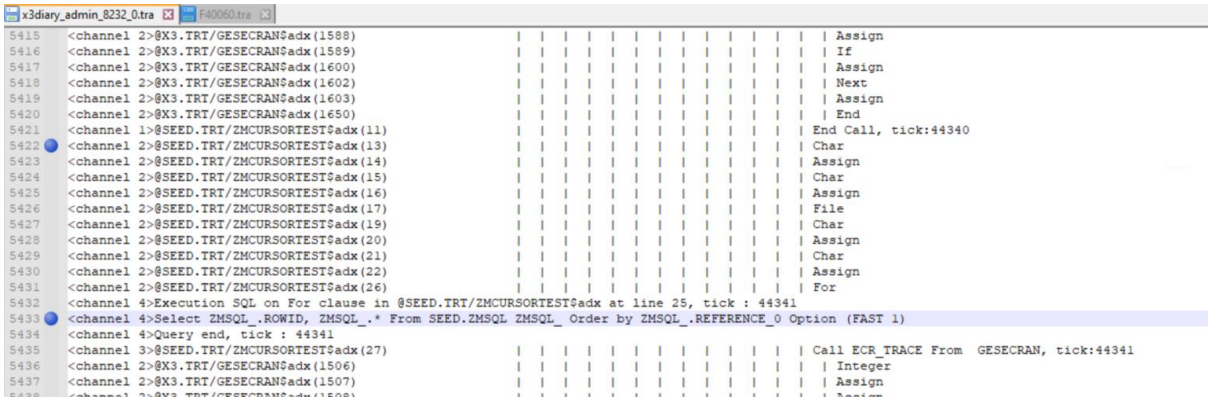


```
x3diary_admin_8232_0.tra | x3diary_admin_7340_0.tra | F40060.tra
13  From SEED.ZMSQL ZMSQL_
14  Order by ZMSQL_.REFERENCE_0
15  Option (FAST 1)
16  <channel 8>SQLGetStmtAttr result: 0
17  <channel 8>Postionnement du dataset a 20 lignes
18  <channel 8>lecture ler enreg du curseur  de For sur SEED.ZMSQL
19  <channel 8>Select ZMSQL_.ROWID, ZMSQL_.*
20  From SEED.ZMSQL ZMSQL_
21  Order by ZMSQL_.REFERENCE_0
22  Option (FAST 1)
23  <channel 8>Resultat execution to Record : 0 at  Wed Jun 28 15:31:36.786 2023
24
25  <channel 8>Resultat ler Fetch to Record, 20 lignes : 0 at  Wed Jun 28 15:31:36.786 2023
26
27  <channel 8>Rowid lu pour la table 9 : 1
28  <channel 8>Requete : 0, Total : 240
29  <channel 8>Requete : 0, Total : 240
30  <channel 8>Fetch buffer
31  <channel 8>Rowid lu pour la table 9 : 112
32  <channel 8>Fetch buffer
33  <channel 8>Rowid lu pour la table 9 : 223
34  <channel 8>Fetch buffer
35  <channel 8>Rowid lu pour la table 9 : 245
36  <channel 8>Fetch buffer
37  <channel 8>Rowid lu pour la table 9 : 256
38  <channel 8>Fetch buffer
39  <channel 8>Rowid lu pour la table 9 : 267
40  <channel 8>Fetch buffer
41  <channel 8>Rowid lu pour la table 9 : 278
42  <channel 8>Fetch buffer
43  <channel 8>Rowid lu pour la table 9 : 289
44  <channel 8>Fetch buffer
45  <channel 8>Rowid lu pour la table 9 : 300
46  <channel 8>Fetch buffer
47  <channel 8>Rowid lu pour la table 9 : 2
48  <channel 8>Requete : 0, Total : 240
49  <channel 8>Fetch buffer

earch results - (310 hits)
Search "Rowid lu pour la table 9" (310 hits in 1 file of 1 searched)
  D:\SageSupport\SQLcourse\CursorTest\X3codeOutputs\x3diary_admin_7340_0.tra (310 hits)
    Line  27: <channel 8>Rowid lu pour la table 9 : 1
    Line  31: <channel 8>Rowid lu pour la table 9 : 112
    Line  33: <channel 8>Rowid lu pour la table 9 : 223
    Line  35: <channel 8>Rowid lu pour la table 9 : 245
    Line  37: <channel 8>Rowid lu pour la table 9 : 256
    Line  39: <channel 8>Rowid lu pour la table 9 : 267
    Line  41: <channel 8>Rowid lu pour la table 9 : 278
    Line  43: <channel 8>Rowid lu pour la table 9 : 289
    Line  45: <channel 8>Rowid lu pour la table 9 : 300
    Line  47: <channel 8>Rowid lu pour la table 9 : 2
    Line  50: <channel 8>Rowid lu pour la table 9 : 13
    Line  53: <channel 8>Rowid lu pour la table 9 : 24
    Line  56: <channel 8>Rowid lu pour la table 9 : 35
    Line  59: <channel 8>Rowid lu pour la table 9 : 46
    Line  62: <channel 8>Rowid lu pour la table 9 : 57
    Line  65: <channel 8>Rowid lu pour la table 9 : 68
```

The adonix engine trace log is “x3diary_admin_8232_0.tra”

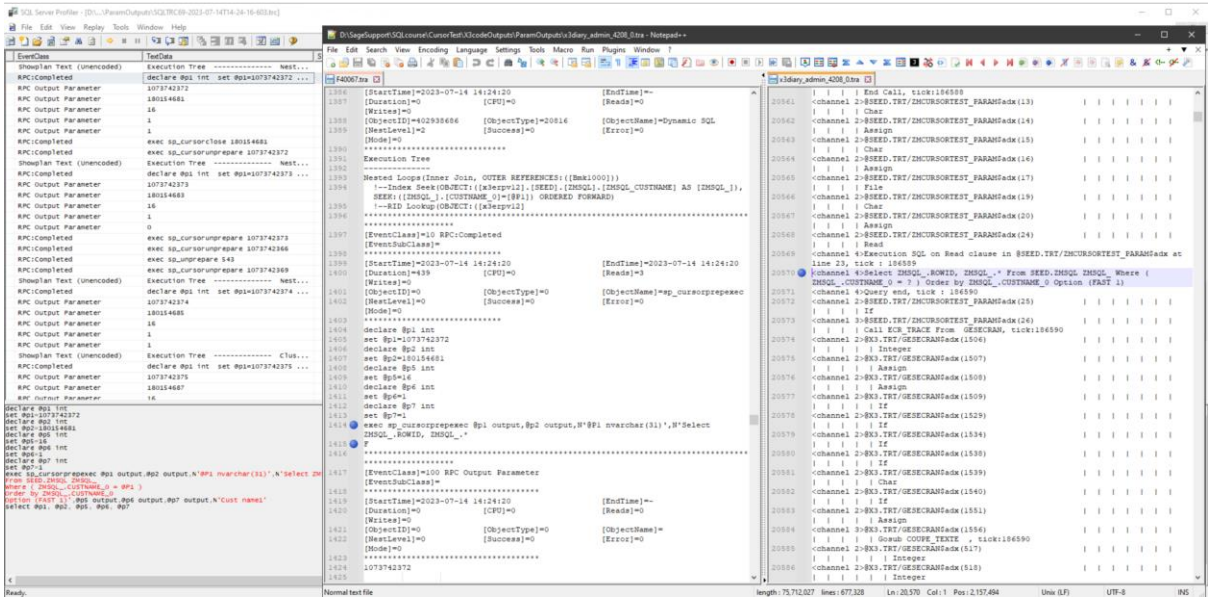
- The adonix trace only show the SELECT statements, not INSERT, UPDATE, DELETE
- You do not see the bind parameters passed into the SQL statement at execution time



```
x3diary_admin_8232_0.tra | F40060.tra
5415  <channel 2>@X3.TRT/GESECRAN$adx(1588)                                                                  | Assign
5416  <channel 2>@X3.TRT/GESECRAN$adx(1589)                                                                  | If
5417  <channel 2>@X3.TRT/GESECRAN$adx(1600)                                                                  | Assign
5418  <channel 2>@X3.TRT/GESECRAN$adx(1602)                                                                  | Next
5419  <channel 2>@X3.TRT/GESECRAN$adx(1603)                                                                  | Assign
5420  <channel 2>@X3.TRT/GESECRAN$adx(1650)                                                                  | End
5421  <channel 1>@SEED.TRT/ZMCURSORTEST$adx(11)                                                              | End Call, tick:44340
5422  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(13)                                                              | Char
5423  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(14)                                                              | Assign
5424  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(15)                                                              | Char
5425  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(16)                                                              | Assign
5426  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(17)                                                              | File
5427  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(19)                                                              | Char
5428  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(20)                                                              | Assign
5429  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(21)                                                              | Char
5430  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(22)                                                              | Assign
5431  <channel 2>@SEED.TRT/ZMCURSORTEST$adx(26)                                                              | For
5432  <channel 4>Execution SQL on For clause in @SEED.TRT/ZMCURSORTEST$adx at line 25, tick : 44341
5433  <channel 4>Select ZMSQL_.ROWID, ZMSQL_.* From SEED.ZMSQL ZMSQL_ Order by ZMSQL_.REFERENCE_0 Option (FAST 1)
5434  <channel 4>Query end, tick : 44341
5435  <channel 3>@SEED.TRT/ZMCURSORTEST$adx(27)                                                              | Call ECR_TRACE From  GESECRAN, tick:44341
5436  <channel 2>@X3.TRT/GESECRAN$adx(1506)                                                                  | Integer
5437  <channel 2>@X3.TRT/GESECRAN$adx(1507)                                                                  | Assign
```

Searching for ZMCURSORTEST in the trace doesn’t really show anything that interesting other than the SQL itself.

Repeat the above steps for “ZMCURSORTEST_PARAM.src” reviewing the outputs from “..\SQLcourse\CursorTest\X3codeOutputs\ParamOutputs” and notice the differences (as we are explicitly looking for one record using an explicit index in the 4GL code.

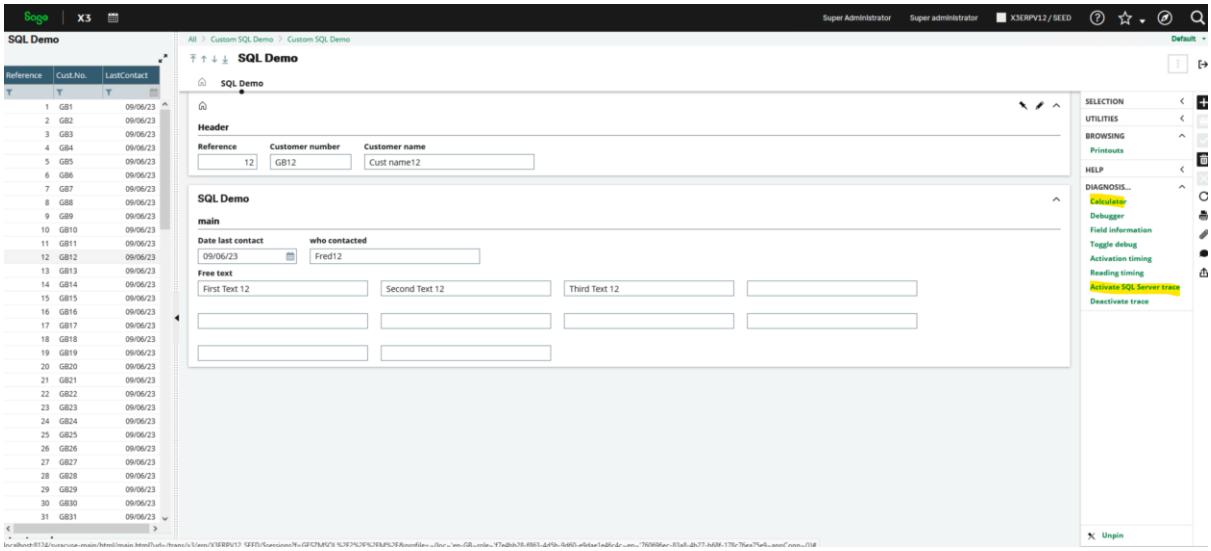


Sadly the F40067.tra truncates the full SQL output, so to find the parameter we need the SQL trace file itself.

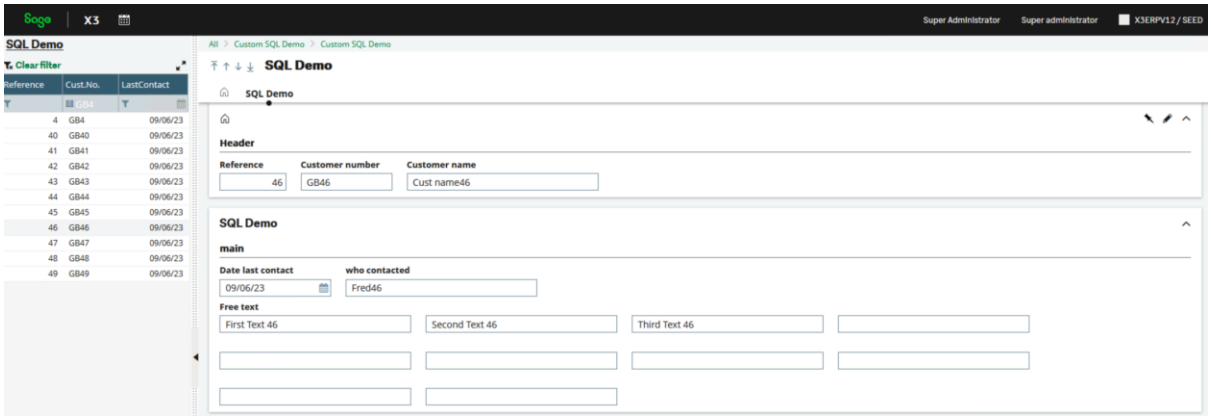
> *NOTE: regards testing X3 generated SQL from within SSMS: if you have SQL which isn’t returning the right records, it’s probably worth trying the generated SQL in SSMS to confirm results and investigate further. If however your concern is a performance issue, then you will likely not see the same issue when running the SQL through SSMS, at least when cursors are used.*

- Finding the SQL used for user input in screens.

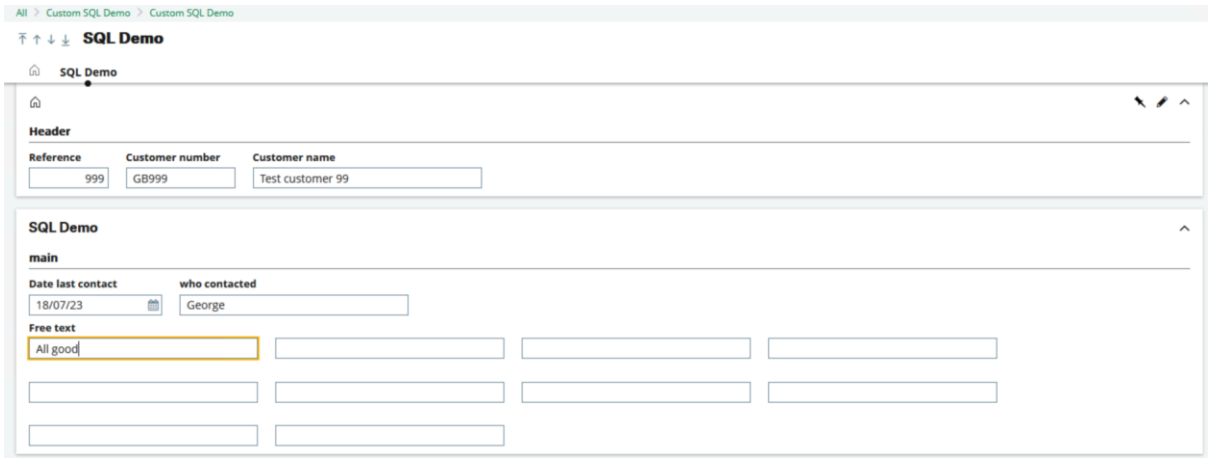
It is pretty much the same as for the 4GL code just discussed, in that we can use the Diagnosis tools in the same way. After launching the custom “SQL Demo screen”, turn on SQL trace and Engine trace.



Query on the left list for Cust.No. “GB4”then select GB46

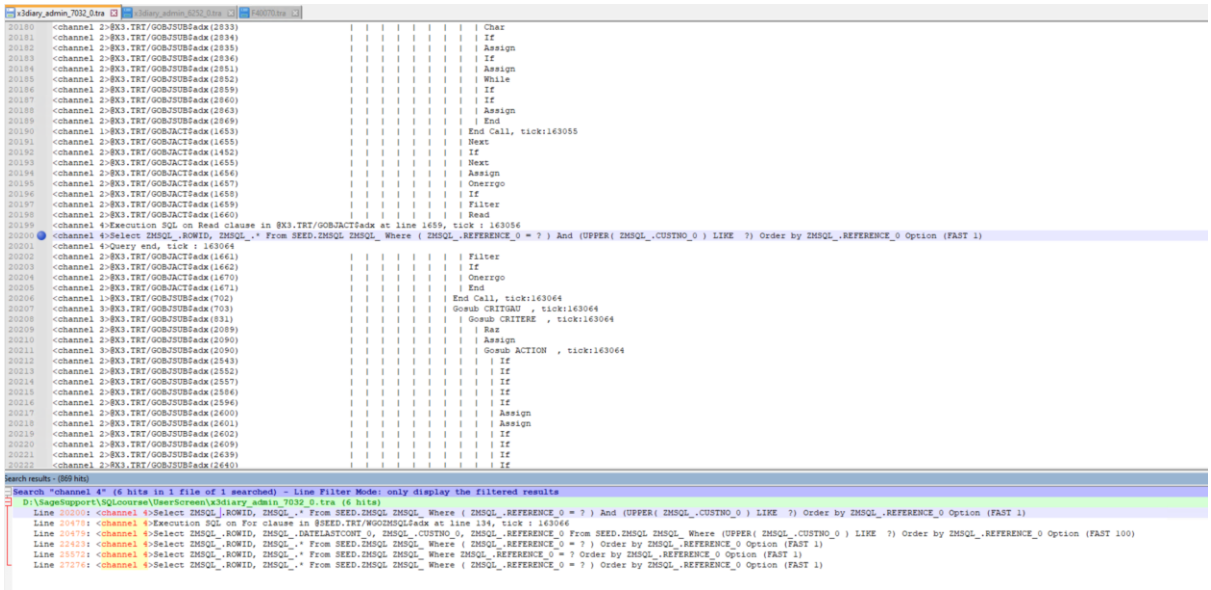


Insert new record

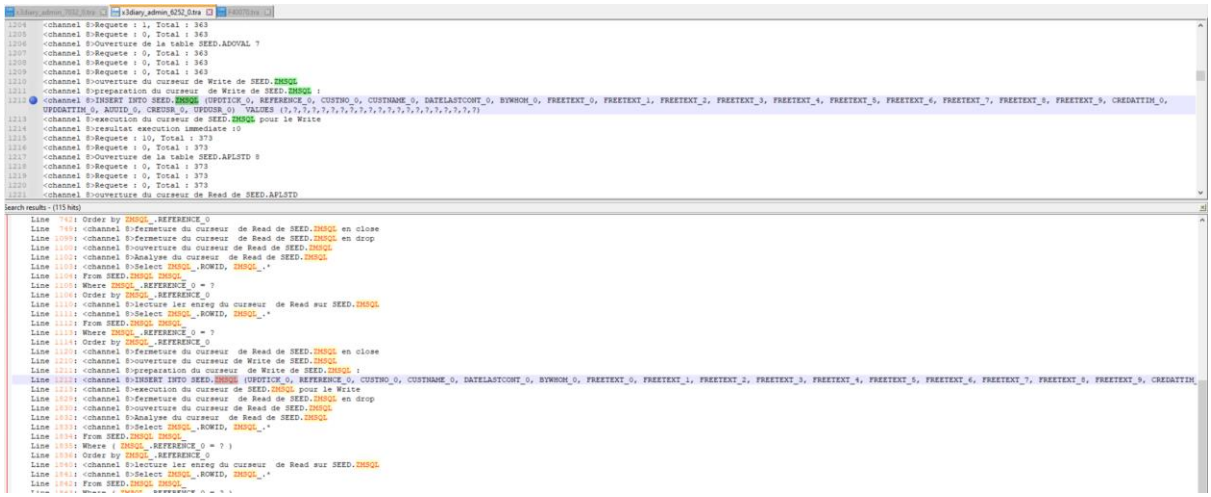


Turn off Engine trace and Deactivate SQL trace, then review the log files.

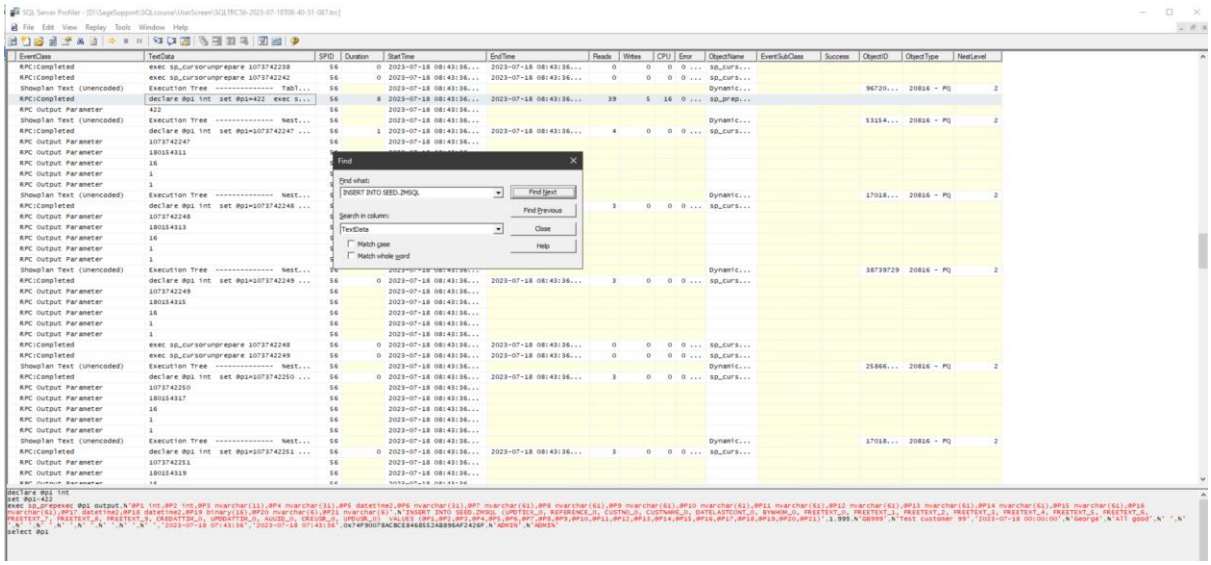
The Adonix trace file show the SELECT SQL statements, but no INSERT



The Sadoss trace shows all the SQL, including the INSERT



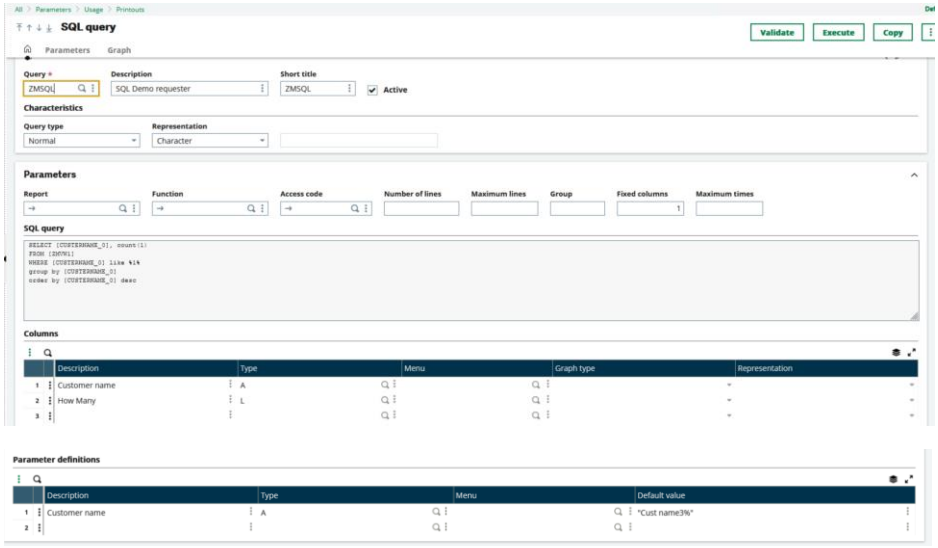
Similarly in the SQL trace we can find the INSERT as well as the other statements, and here we can also see the parameters used.

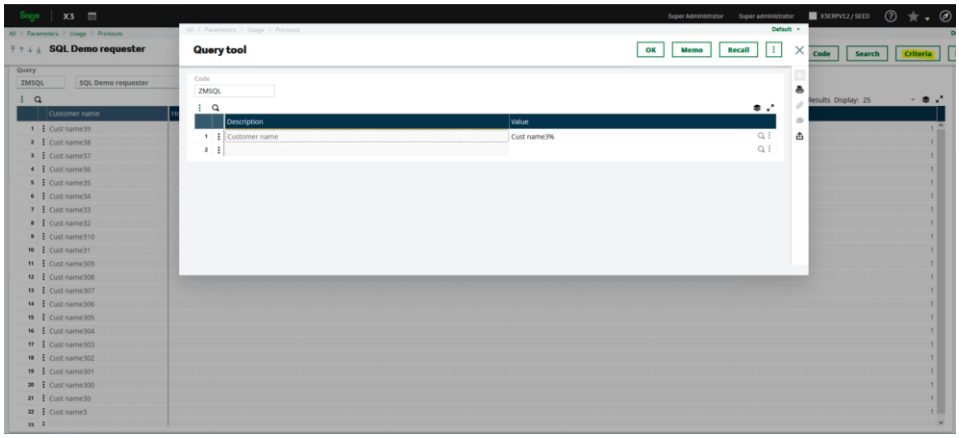


- Requesters

All requesters allow you to create your own enquiries, however I will only touch briefly on SQL Requester – as it is to do with SQL statements.

The main points to note are that there should not be any reference to a folder name in the SQL, you can reference tables and/or views, and there are some restrictions on the more complex SQL you are allowed to use.

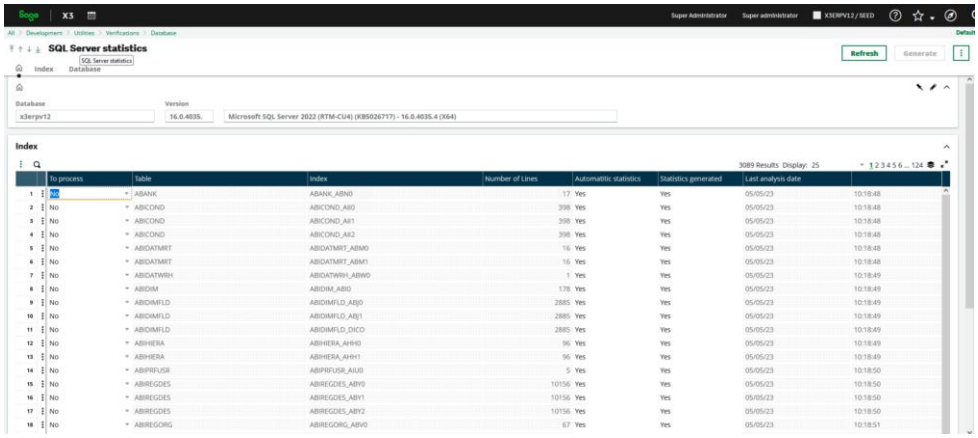




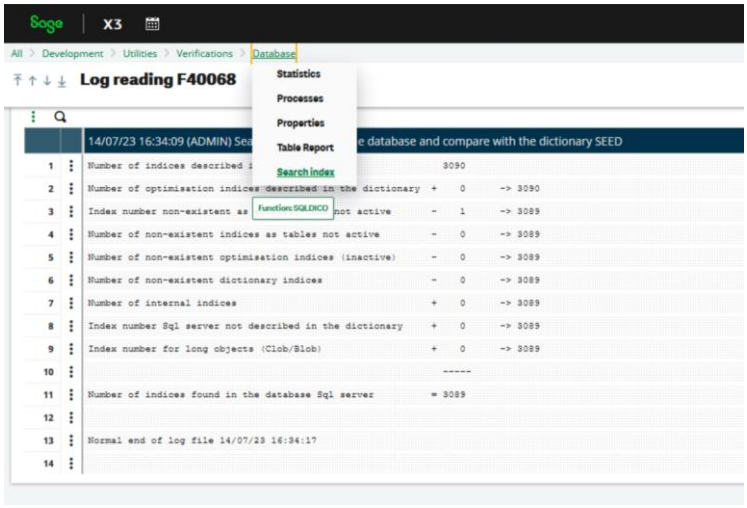
A quick bit of techy

There are some interesting areas for Database Administrators (DBA) which can be reviewed using the SQL server tools directly (SSMS) but are also available via the X3 front end.

- Statistics



- Missing or added indexes

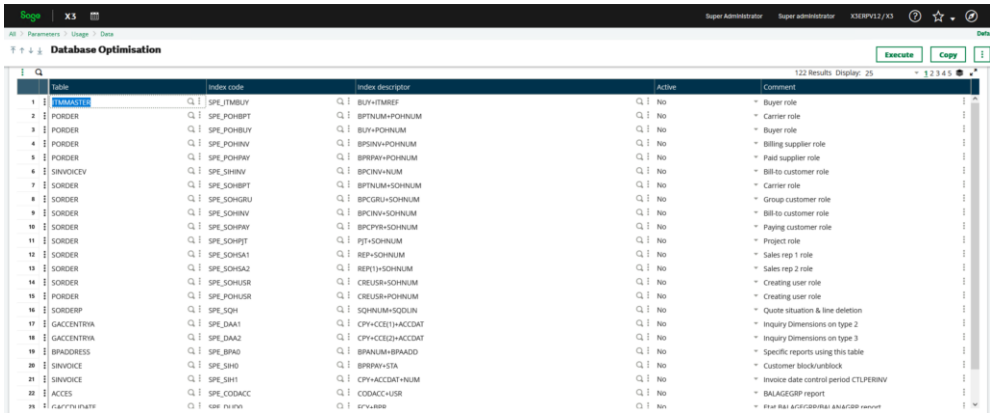


- Database optimisation

Allows you to activate a pre-defined custom index (Copy from X3 folder) or add your own

Connect to X3 folder first

As with adding any custom index, you should perform your own testing to confirm the benefit of adding the index outweighs any disadvantage introduced.



SQL you can use

- Investigation Scripts

See PDF “07 - Investigation Scripts.pdf” from “Index page: Sage X3 Technical Support Tips and Tricks (September 2021)” (https://www.sagecity.com/gb/sage-x3/b/sage-x3-uk-support-insights/posts/index-page-sage-x3-technical-support-tips-and-tricks-september-2021)

Take a look at “..\InvestigationScripts\SQL \mzBatchJobs.sql” if we get time

- Common Tools data model SQL examples

Available from the Sage University training courses “Understanding the Sage X3[Common Tools, Distribution| Manufactoring] data model”

Appendix- Additional learning

Sage University

Using Sage X3 SQL Views to create Requesters/Queries - Level 1

Using Sage X3 SQL Views to create Requesters/Queries - Level 2

Understanding 4GL as an Implementation Consultant – Part 1

Understanding 4GL as an Implementation Consultant – Part 2

Understanding the Sage X3 common tools data model

Understanding the Sage X3 distribution data model

Understanding the Sage X3 manufacturing data model

What's new in SQL 2022

https://learn.microsoft.com/en-us/shows/data-exposed/introduction-to-sql-server-2022-ep1

https://learn.microsoft.com/en-gb/sql/sql-server/what-s-new-in-sql-server-2022?view=sql-server-ver16