

SQL with Sage X3

Mike Shaw — 15th November 2023



Contents

Introduction

How X3 generates SQL statements

A quick bit of “techy”

SQL you can use

Summary



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.

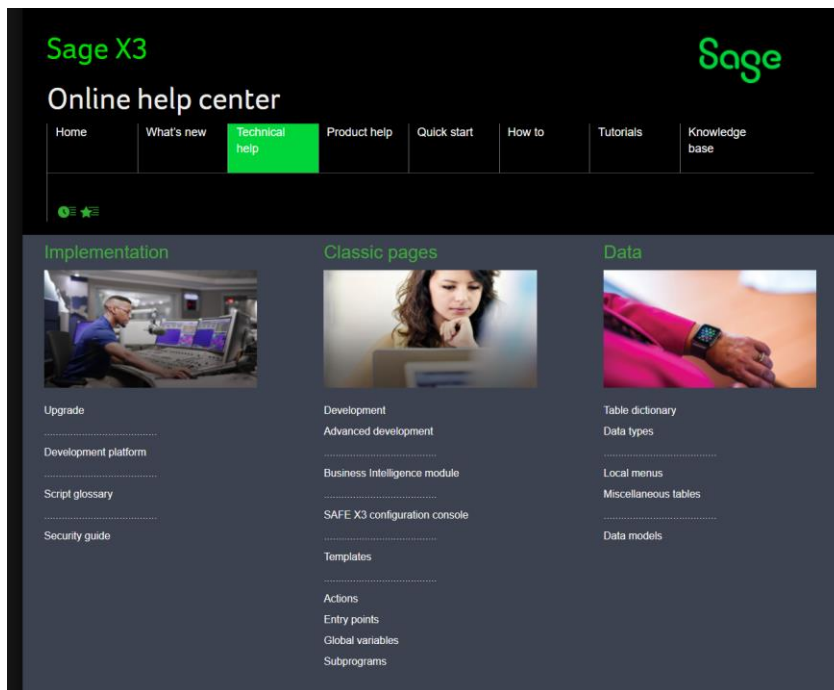
Demonstrations

- Demonstrations and discussions

How X3 generates SQL statements

- 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...)

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>)

Page 4 of 16

All > Printouts > Printouts

Enter report parameters

Report code * ZMSQL SQL demo

Parameter definitions

Parameter title	Parameter type	First value	Final value
1			

Report destination

Destination: VIRTPRINT Virtual HP PCL 6 Printer

Departure characteristics

Output type: Printer Server: x3erpv12sqlvm:1890 Printer: Virtual HP Universal Printing PCL 6

File:

Orientation: Portrait Copies: 1 Collated copies: ☐

Format: Crystal Report (*.rpt) Pages:

Paper size: Printer preference Language: BRI

Print

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

In the "RPT_Trace_ZMSQLtable_CrDII_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.

```

238 if {?site}<"" then
239   lsite*:=*{?site}
240 else if {?society}<"" then
241   lsociety*:=*{?society}
242 else
243   ldossier
244
245
246 FormulaFields names : CrystalReports.FormulaField shared stringvar lsite := TextOfChapter ({?X3DOS},{?X3LAN},2300,5) ;
247 shared stringvar lsociety := TextOfChapter ({?X3DOS},{?X3LAN},2300,6) ;
248 shared stringvar ldossier := TextOfChapter ({?X3DOS},{?X3LAN},2300,7) ;
249
250 FormulaFields names : CrystalReports.FormulaField "Batch Requests"
251
252 SQL instruction:
253 SELECT "ZMSQL"."REFERENCE_0", "ZMSQL"."CUSTNO_0", "ZMSQL"."CUSTNAME_0", "ZMSQL"."DATELASTCONT_0", "ZMSQL"."BYWHOM_0", "ZMSQL"."FRETEXT_0", "ZMSQL"."FRETEXT_1", "ZMSQL"."FRETEXT_3"
254 FROM "x3erpv12"."SEED"."ZMSQL" "ZMSQL"
255
256
257
258 Printer destination: preparing Crystal Report
259 Scan document: _x3opt_ parameter NOT FOUND
260 Printer: Virtual HP Universal Printing PCL 6
261 Paper size PaperSize set to PaperA4 (9), from value 9
262 Paper source FormSource (15)
263 Duplex: Simplex (1)
264 Printer format: preparing Crystal Report
  
```

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.

```

331 FormulaFields names : CrystalReports.FormulaField shared stringvar var3 ;
332 var3
333
334
335 FormulaFields names : CrystalReports.FormulaField shared stringvar lsite ;
336 shared stringvar ldossier ;
337 shared stringvar lsociety ;
338 if {?site}<"" then
339   lsite*:=*{?site}
340 else if {?society}<"" then
341   lsociety*:=*{?society}
342 else
343   ldossier
344
345
346 FormulaFields names : CrystalReports.FormulaField shared stringvar lsite := TextOfChapter ({?X3DOS},{?X3LAN},2300,5) ;
347 shared stringvar lsociety := TextOfChapter ({?X3DOS},{?X3LAN},2300,6) ;
348 shared stringvar ldossier := TextOfChapter ({?X3DOS},{?X3LAN},2300,7) ;
349
350
351 SQL instruction:
352 SELECT "ADOVAL"."CHARITRE_0", "ADOVAL"."HOLIS_0", "ADOVAL"."VALEUR_0", "ADOVAL"."FCV_0", "ADOVAL"."CMI_0", "ADOVAL"."PARAM_0", "ATEXTE"."TEXTE_0", "ATEXTE"."LAN_0", "ADOVAL"."TITRE_0", "ATEXTRA"."MONTAB_0", "ATEXTRA"."LABOUR_0",
353 FROM ((("x3erpv12"."SEED"."ADOVAL" INNER JOIN "x3erpv12"."SEED"."ADOVAL" ON "ADOVAL"."PARAM_0"="ADOVAL"."PARAM_0") LEFT OUTER JOIN "x3erpv12"."SEED"."ATEXTRA" ON "ATEXTRA"."CODE_0"="ADOVAL"."CODE_0") LEFT OUTER
354 JOIN "x3erpv12"."SEED"."ATEXTE" ON "ADOVAL"."CMI_0"="ATEXTE"."CMI_0") LEFT OUTER JOIN "x3erpv12"."SEED"."ATEXTRA" ON "ATEXTRA"."CODE_0"="ATEXTRA"."CODE_0") LEFT OUTER
355 JOIN "ADOVAL"."CHARITRE_0" ON "ADOVAL"."CMI_0"="CHARITRE_0" AND "ATEXTE"."LAN_0"="BRI" AND "ATEXTRA"."MONTAB_0"="0" AND "ADOVAL"."CMI_0"="1" AND "ADOVAL"."FCV_0"="nt" OR "ADOVAL"."CMI_0"="nt" OR "ADOVAL"."FCV_0"="nt") AND
356 ORDER BY "ADOVAL"."CHARITRE_0", "ADOVAL"."PARAM_0"
357
358
359 File destination: preparing Crystal Report
360 Issue file: D:\sage\user\X3\Print\Temp\SEED-356-20230714153629-96_
361 PDF format: preparing Crystal Report
362
  
```

- Sage X3 4GL

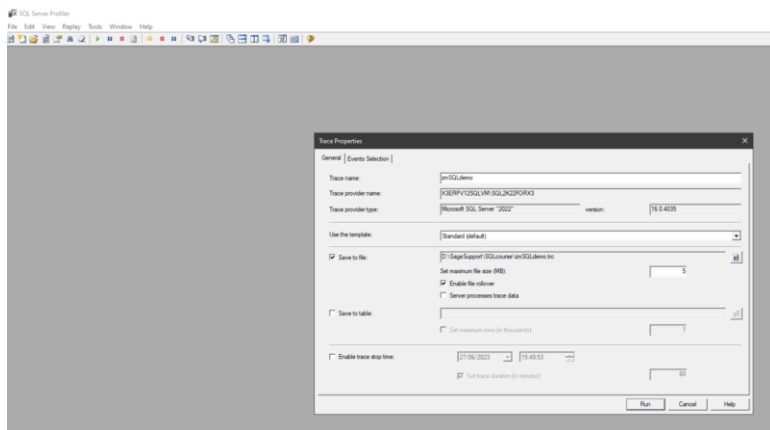
Before we look at X3 4GL code, let's take a look at a simple example run in SSMS that uses cursors and parameters, and procedures... oh and a bit of Dynamic SQL too...!!

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

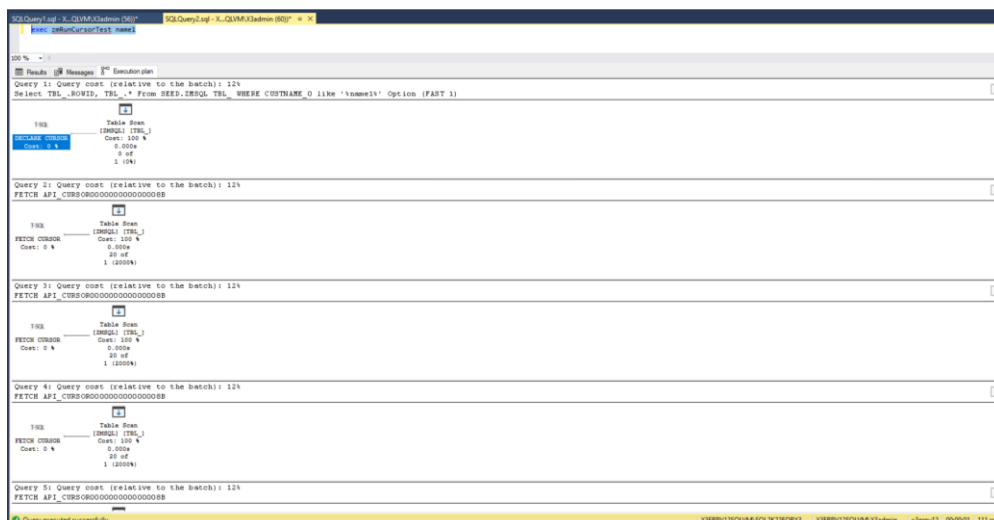
Review "CursorTest\zmProcedure.sql" and I'll provide a cursory explanation of what it is doing

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

EventClass	TextData	Duration	SPID	DatabaseID	DatabaseName	ObjectName	LogName
Trace started							
SQLBatchCompleted	SET STATISTICS XML ON	0.00	58	5	X3BPV12		X3BPV12
SPIDCompleted	set @CustomerId = 'name1' --	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'name1' --	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	SET @StartDate = @StartDate	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'Start time' + CONVERT(NVARCHAR, @StartDate, 113)	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'name1' --	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	set @IDN = 0 --	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	set @Statement = 'select TEL_ROWID, TEL...' from SEED.ZMSQL TEL... WHERE CUSTNAME_ID like 'N...'	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	DECLARE @rowCount INTEGER = 0 -- FAST_FORWARD	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	DECLARE @rowCount INTEGER = 1 -- READ_ONLY	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	DECLARE @rowCount INTEGER = 20 -- Buffer Size	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	DECLARE @rowCount INTEGER = 2 -- Next row	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	DECLARE @rowCount INTEGER = 1	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'Ready to go...' + @Statement + '!' + @CustomerId + '!' --	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	select TEL_ROWID, TEL... from SEED.ZMSQL TEL... WHERE CUSTNAME_ID like 'name1' option (FAST 1)	493.50	58	5	X3BPV12	20814 - PQ	X3BPV12
SPIDCompleted	exec @SQLCursorOpen @Statement output, @rowCount output, NULL, @Statement, @rowCount output, ...	493.50	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'CursorOpen' + CAST(@rowCount as NVARCHAR(MAX)) + '!' + CAST(@rowCount as NVARCHAR(MAX))	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	WHILE @rowCount > 0	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	FETCH API_CURSOR @rowCount @rowCount	1309.50	58	5	X3BPV12	20801 - AQ	X3BPV12
SPIDCompleted	exec @SQLCursorFetch @rowCount, @rowCount, @rowCount	1309.50	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'CursorFetch' + CAST(@rowCount as NVARCHAR(MAX)) + '!' + CAST(@rowCount as NVARCHAR(MAX))	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	WHILE @rowCount > 0	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	FETCH API_CURSOR @rowCount @rowCount	224.50	58	5	X3BPV12	20801 - AQ	X3BPV12
SPIDCompleted	exec @SQLCursorFetch @rowCount, @rowCount, @rowCount	224.50	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'CursorFetch' + CAST(@rowCount as NVARCHAR(MAX)) + '!' + CAST(@rowCount as NVARCHAR(MAX))	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	WHILE @rowCount > 0	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	FETCH API_CURSOR @rowCount @rowCount	99.50	58	5	X3BPV12	20801 - AQ	X3BPV12
SPIDCompleted	exec @SQLCursorFetch @rowCount, @rowCount, @rowCount	100.50	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'CursorFetch' + CAST(@rowCount as NVARCHAR(MAX)) + '!' + CAST(@rowCount as NVARCHAR(MAX))	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	WHILE @rowCount > 0	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	FETCH API_CURSOR @rowCount @rowCount	87.50	58	5	X3BPV12	20801 - AQ	X3BPV12
SPIDCompleted	exec @SQLCursorFetch @rowCount, @rowCount, @rowCount	87.50	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'CursorFetch' + CAST(@rowCount as NVARCHAR(MAX)) + '!' + CAST(@rowCount as NVARCHAR(MAX))	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	WHILE @rowCount > 0	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	FETCH API_CURSOR @rowCount @rowCount	70.50	58	5	X3BPV12	20801 - AQ	X3BPV12
SPIDCompleted	exec @SQLCursorFetch @rowCount, @rowCount, @rowCount	70.50	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	PRINT 'CursorFetch' + CAST(@rowCount as NVARCHAR(MAX)) + '!' + CAST(@rowCount as NVARCHAR(MAX))	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	WHILE @rowCount > 0	0.00	58	5	X3BPV12	8272 - P	X3BPV12
SPIDCompleted	FETCH API_CURSOR @rowCount @rowCount	78.50	58	5	X3BPV12	20801 - AQ	X3BPV12
SPIDCompleted							

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”

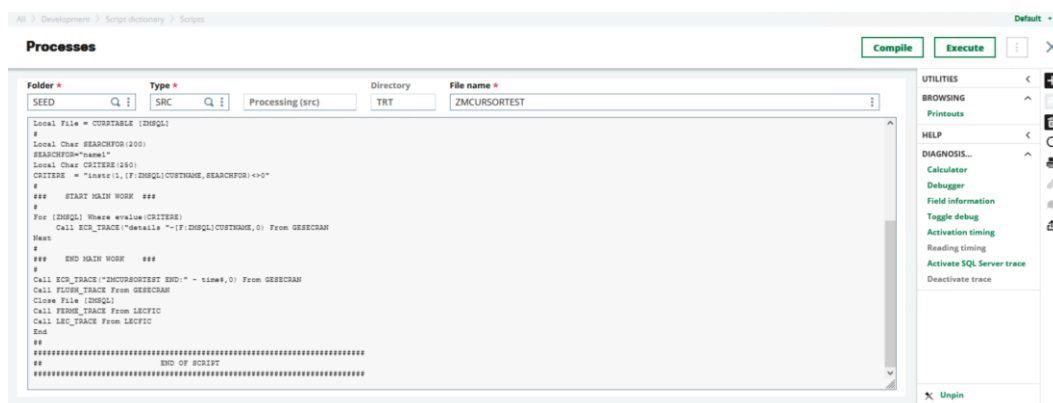
- Finding the SQL in X3

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 “ZMCURSORTTEST.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):

EventClass	TaskClass	SFD	Duration	StartTime	EndTime	Reads	Writes	CPU	Error	ObjName	EventSubClass	Success	ObjID	ObjType	TestLevel
APC Output: Parameter	1073742210	0		2023-06-28 15:31:36...											
APC Output: Parameter	101014243	0		2023-06-28 15:31:36...											
APC Output: Parameter	16	0		2023-06-28 15:31:36...											
APC Output: Parameter	1	0		2023-06-28 15:31:36...											
APC Output: Parameter	0	0		2023-06-28 15:31:36...											
APC/Completed	exec sp_cursorprepare 1073742210	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	0	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			48030...	20616 - PQ	2
APC/Completed	exec sp_cursorfetch 1073742210	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
APC Output: Parameter	1073742211	0		2023-06-28 15:31:36...											
APC Output: Parameter	101014243	0		2023-06-28 15:31:36...											
APC Output: Parameter	16	0		2023-06-28 15:31:36...											
APC Output: Parameter	1	0		2023-06-28 15:31:36...											
APC Output: Parameter	20	0		2023-06-28 15:31:36...											
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					
Showplan Text (Unencoded)	Execution Tree ----- Nest...	0		2023-06-28 15:31:36...						Dynam...			22625...	20601 - AQ	2
APC/Completed	exec sp_cursorfetch 18014242,1,1,20	0	0	2023-06-28 15:31:36...	2023-06-28 15:31:36...	22	0	0	0	sp_cursor...					

```

SELECT * FROM T1
SET ROWCOUNT 42211
SET ROWCOUNT 16
SET ROWCOUNT 1
SET ROWCOUNT 0
DECLARE @obj INT
DECLARE @obj INT
SELECT @obj INT
EXEC sp_cursorprepare @
```

"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 1er 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 1er 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

Search 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 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

```

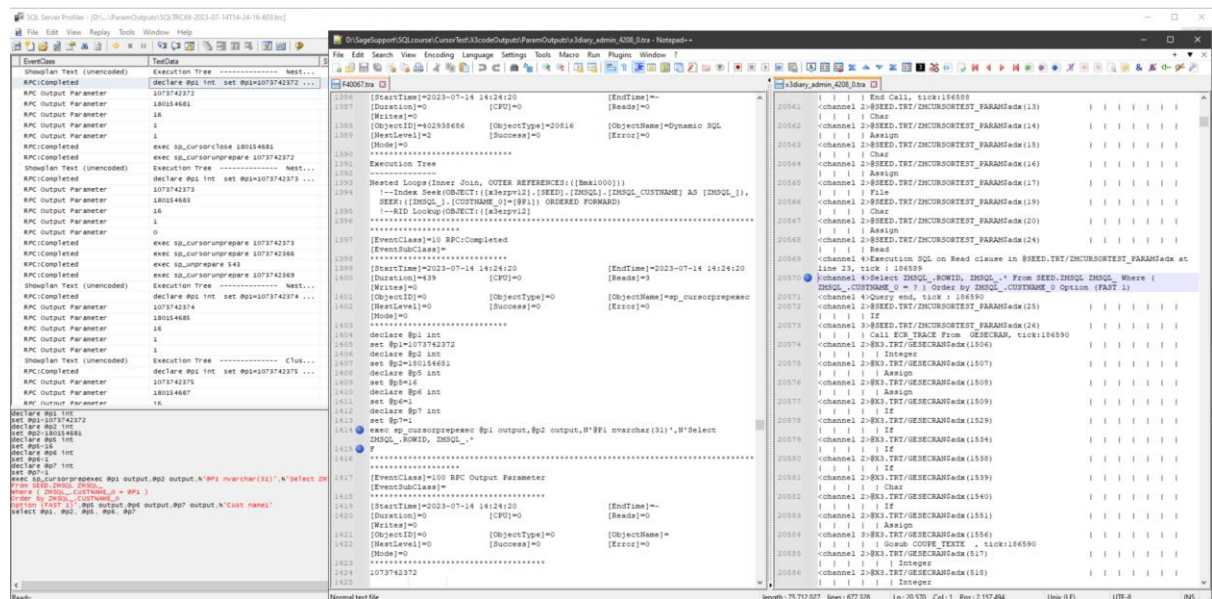
The adonix engine trace log is "x3diary admin 8232 0.tra"

- 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.trt	F40060.trt
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/ZMCURSORTTEST?adx(11)	End Call, tick:44340
5422 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(13)	Char
5423 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(14)	Assign
5424 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(15)	Char
5425 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(16)	Assign
5426 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(17)	File
5427 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(19)	Char
5428 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(20)	Assign
5429 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(21)	Char
5430 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(22)	Assign
5431 <channel 2>@SEED.TRT/ZMCURSORTTEST?adx(26)	For
5432 <channel 4>Execution SQL on For clause in @SEED.TRT/ZMCURSORTTEST?adx at line 25, tick : 44341	
5433 <channel 4>Select ZMSQL_ROWID,ZMSQL_* From SEED.ZMSQL Order by ZMSQL_REFERENCE_0 Option (FAST 1)	
5434 <channel 4>Query end, tick : 44341	
5435 <channel 3>@SEED.TRT/ZMCURSORTTEST?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).

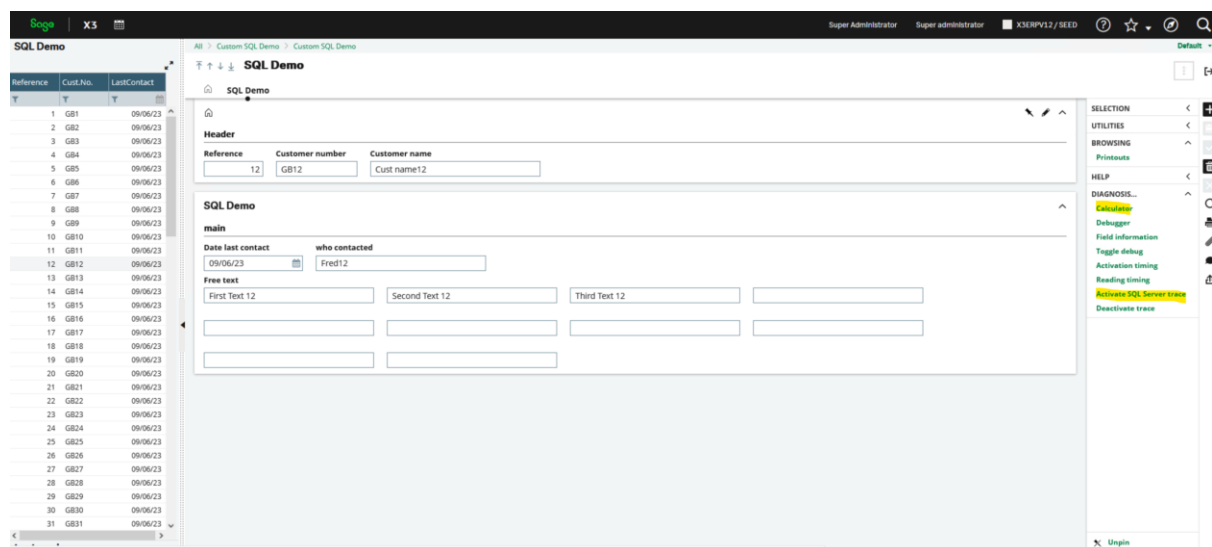


Sadly the F40067.tra truncates the full SQL output, so to find the parameter we need the SQL trace file itself.

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

Sogo

X3

SQL Demo

Clear Filter

Reference	Cust.No.	LastContact
4	GB4	09/06/23
40	GB40	09/06/23
41	GB41	09/06/23
42	GB42	09/06/23
43	GB43	09/06/23
44	GB44	09/06/23
45	GB45	09/06/23
46	GB46	09/06/23
47	GB47	09/06/23
48	GB48	09/06/23
49	GB49	09/06/23

SQL Demo

Header

Reference

Customer number

Customer name

46

GB46

Cust name-46

SQL Demo

main

Date last contact

who contacted

09/06/23

Fred46

Free text

First Text 46

Second Text 46

Third Text 46

Insert new record

[All](#) > [Custom SQL Demo](#) > [Custom SQL Demo](#)


[SQL Demo](#)

Header

Reference	Customer number	Customer name
999	GB999	Test customer 99

SQL Demo

main

Date last contact: 18/07/23 

who contacted: George

Free text: All good

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

```

21019 |> channel 2>B3X.TRT/GOB/SUBStad(2033) | | | | | | | Char
21018 |> channel 2>B3X.TRT/GOB/SUBStad(2034) | | | | | | | If
21017 |> channel 2>B3X.TRT/GOB/SUBStad(2035) | | | | | | | Assign
21016 |> channel 2>B3X.TRT/GOB/SUBStad(2036) | | | | | | | If
21015 |> channel 2>B3X.TRT/GOB/SUBStad(2037) | | | | | | | Assign
21014 |> channel 2>B3X.TRT/GOB/SUBStad(2038) | | | | | | | While
21013 |> channel 2>B3X.TRT/GOB/SUBStad(2039) | | | | | | | If
21012 |> channel 2>B3X.TRT/GOB/SUBStad(2040) | | | | | | | If
21011 |> channel 2>B3X.TRT/GOB/SUBStad(2041) | | | | | | | Assign
21010 |> channel 2>B3X.TRT/GOB/SUBStad(2042) | | | | | | | End
21009 |> channel 1>B3X.TRT/GOB/ACTStad(1653) | | | | | | | End Call, tick:163055
21008 |> channel 2>B3X.TRT/GOB/ACTStad(1655) | | | | | | | Next
21007 |> channel 2>B3X.TRT/GOB/ACTStad(1656) | | | | | | | If
21006 |> channel 2>B3X.TRT/GOB/ACTStad(1657) | | | | | | | Next
21005 |> channel 2>B3X.TRT/GOB/ACTStad(1658) | | | | | | | Assign
21004 |> channel 2>B3X.TRT/GOB/ACTStad(1659) | | | | | | | Overgro
21003 |> channel 2>B3X.TRT/GOB/ACTStad(1660) | | | | | | | If
21002 |> channel 2>B3X.TRT/GOB/ACTStad(1661) | | | | | | | Filter
21001 |> channel 4>Function SQL on Read of class line tick: 163064 | | | | | | | Read
21000 |> channel 4>Select ZMSQL_ROWID, ZMSQL_* From SEED.ZMSQL_ZMSQL Where ( ZMSQL_REFERENCE_Option = ? ) And (UPPER(ZMSQL_CUSTMO_0) LIKE ?) Order by ZMSQL_REFERENCE_Option (FAST 1)
20999 |> channel 4>Query end, tick: 163064
20998 |> channel 2>B3X.TRT/GOB/ACTStad(1662) | | | | | | | Filter
20997 |> channel 2>B3X.TRT/GOB/ACTStad(1663) | | | | | | | If
20996 |> channel 2>B3X.TRT/GOB/ACTStad(1664) | | | | | | | Overgro
20995 |> channel 2>B3X.TRT/GOB/ACTStad(1665) | | | | | | | End
20994 |> channel 2>B3X.TRT/GOB/SUBStad(702) | | | | | | | End Call, tick:163064
20993 |> channel 3>B3X.TRT/GOB/SUBStad(703) | | | | | | | Send CHITAD , tick:163064
20992 |> channel 3>B3X.TRT/GOB/SUBStad(831) | | | | | | | Send CHITRE , tick:163064
20991 |> channel 2>B3X.TRT/GOB/SUBStad(2089) | | | | | | | Rax
20990 |> channel 2>B3X.TRT/GOB/SUBStad(2090) | | | | | | | Assign
20989 |> channel 3>B3X.TRT/GOB/SUBStad(2090) | | | | | | | Good ACTION , tick:163064
20988 |> channel 2>B3X.TRT/GOB/SUBStad(2543) | | | | | | | If
20987 |> channel 2>B3X.TRT/GOB/SUBStad(2544) | | | | | | | If
20986 |> channel 2>B3X.TRT/GOB/SUBStad(2545) | | | | | | | If
20985 |> channel 2>B3X.TRT/GOB/SUBStad(2546) | | | | | | | If
20984 |> channel 2>B3X.TRT/GOB/SUBStad(2547) | | | | | | | If
20983 |> channel 2>B3X.TRT/GOB/SUBStad(2548) | | | | | | | If
20982 |> channel 2>B3X.TRT/GOB/SUBStad(2549) | | | | | | | If
20981 |> channel 2>B3X.TRT/GOB/SUBStad(2550) | | | | | | | If
20980 |> channel 2>B3X.TRT/GOB/SUBStad(2551) | | | | | | | If
20979 |> channel 2>B3X.TRT/GOB/SUBStad(2552) | | | | | | | If
20978 |> channel 2>B3X.TRT/GOB/SUBStad(2553) | | | | | | | If
20977 |> channel 2>B3X.TRT/GOB/SUBStad(2554) | | | | | | | If
20976 |> channel 2>B3X.TRT/GOB/SUBStad(2555) | | | | | | | If
20975 |> channel 2>B3X.TRT/GOB/SUBStad(2556) | | | | | | | If
20974 |> channel 2>B3X.TRT/GOB/SUBStad(2557) | | | | | | | If
20973 |> channel 2>B3X.TRT/GOB/SUBStad(2558) | | | | | | | If
20972 |> channel 2>B3X.TRT/GOB/SUBStad(2559) | | | | | | | If
20971 |> channel 2>B3X.TRT/GOB/SUBStad(2560) | | | | | | | If
20970 |> channel 2>B3X.TRT/GOB/SUBStad(2561) | | | | | | | If
20969 |> channel 2>B3X.TRT/GOB/SUBStad(2562) | | | | | | | If
20968 |> channel 2>B3X.TRT/GOB/SUBStad(2563) | | | | | | | If
20967 |> channel 2>B3X.TRT/GOB/SUBStad(2564) | | | | | | | If
20966 |> channel 2>B3X.TRT/GOB/SUBStad(2565) | | | | | | | If
20965 |> channel 2>B3X.TRT/GOB/SUBStad(2566) | | | | | | | If
20964 |> channel 2>B3X.TRT/GOB/SUBStad(2567) | | | | | | | If
20963 |> channel 2>B3X.TRT/GOB/SUBStad(2568) | | | | | | | If
20962 |> channel 2>B3X.TRT/GOB/SUBStad(2569) | | | | | | | If
20961 |> channel 2>B3X.TRT/GOB/SUBStad(2570) | | | | | | | If
20960 |> channel 2>B3X.TRT/GOB/SUBStad(2571) | | | | | | | If
20959 |> channel 2>B3X.TRT/GOB/SUBStad(2572) | | | | | | | If
20958 |> channel 2>B3X.TRT/GOB/SUBStad(2573) | | | | | | | If
20957 |> channel 2>B3X.TRT/GOB/SUBStad(2574) | | | | | | | If
20956 |> channel 2>B3X.TRT/GOB/SUBStad(2575) | | | | | | | If
20955 |> channel 2>B3X.TRT/GOB/SUBStad(2576) | | | | | | | If
20954 |> channel 2>B3X.TRT/GOB/SUBStad(2577) | | | | | | | If
20953 |> channel 2>B3X.TRT/GOB/SUBStad(2578) | | | | | | | If
20952 |> channel 2>B3X.TRT/GOB/SUBStad(2579) | | | | | | | If
20951 |> channel 2>B3X.TRT/GOB/SUBStad(2580) | | | | | | | If
20950 |> channel 2>B3X.TRT/GOB/SUBStad(2581) | | | | | | | If
20949 |> channel 2>B3X.TRT/GOB/SUBStad(2582) | | | | | | | If
20948 |> channel 2>B3X.TRT/GOB/SUBStad(2583) | | | | | | | If
20947 |> channel 2>B3X.TRT/GOB/SUBStad(2584) | | | | | | | If
20946 |> channel 2>B3X.TRT/GOB/SUBStad(2585) | | | | | | | If
20945 |> channel 2>B3X.TRT/GOB/SUBStad(2586) | | | | | | | If
20944 |> channel 2>B3X.TRT/GOB/SUBStad(2587) | | | | | | | If
20943 |> channel 2>B3X.TRT/GOB/SUBStad(2588) | | | | | | | If
20942 |> channel 2>B3X.TRT/GOB/SUBStad(2589) | | | | | | | If
20941 |> channel 2>B3X.TRT/GOB/SUBStad(2590) | | | | | | | If
20940 |> channel 2>B3X.TRT/GOB/SUBStad(2591) | | | | | | | If
20939 |> channel 2>B3X.TRT/GOB/SUBStad(2592) | | | | | | | If
20938 |> channel 2>B3X.TRT/GOB/SUBStad(2593) | | | | | | | If
20937 |> channel 2>B3X.TRT/GOB/SUBStad(2594) | | | | | | | If
20936 |> channel 2>B3X.TRT/GOB/SUBStad(2595) | | | | | | | If
20935 |> channel 2>B3X.TRT/GOB/SUBStad(2596) | | | | | | | If
20934 |> channel 2>B3X.TRT/GOB/SUBStad(2597) | | | | | | | If
20933 |> channel 2>B3X.TRT/GOB/SUBStad(2598) | | | | | | | If
20932 |> channel 2>B3X.TRT/GOB/SUBStad(2599) | | | | | | | If
20931 |> channel 2>B3X.TRT/GOB/SUBStad(2600) | | | | | | | If
20930 |> channel 2>B3X.TRT/GOB/SUBStad(2601) | | | | | | | If
20929 |> channel 2>B3X.TRT/GOB/SUBStad(2602) | | | | | | | If
20928 |> channel 2>B3X.TRT/GOB/SUBStad(2603) | | | | | | | If
20927 |> channel 2>B3X.TRT/GOB/SUBStad(2604) | | | | | | | If
20926 |> channel 2>B3X.TRT/GOB/SUBStad(2605) | | | | | | | If
20925 |> channel 2>B3X.TRT/GOB/SUBStad(2606) | | | | | | | If
20924 |> channel 2>B3X.TRT/GOB/SUBStad(2607) | | | | | | | If
20923 |> channel 2>B3X.TRT/GOB/SUBStad(2608) | | | | | | | If
20922 |> channel 2>B3X.TRT/GOB/SUBStad(2609) | | | | | | | If
209
```

The Sadoss trace shows all the SQL, including the INSERT

- 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.

The screenshot shows the Sage SQL query editor. At the top, there are tabs for 'Parameters', 'Usage', and 'Previous'. Below these, there are buttons for 'Validate', 'Execute', and 'Copy'. The main area is divided into several sections:

- Query:** A text area containing the SQL query: `SELECT (CUSTOMERNAME_01, orders(1) FROM (SELECT WHERE (CUSTOMERNAME_01) LIKE 'A' group by (CUSTOMERNAME_01) order by (CUSTOMERNAME_01) desc`
- Columns:** A table with columns: Description, Type, Menu, Graph type, Representation. It lists 'Customer name' and 'How Many'.
- Parameter definitions:** A table with columns: Description, Type, Menu, Default value. It lists 'Customer name' with a default value of 'Cust name3%'

The screenshot shows the Sage SQL Demo requester interface. On the left, there is a list of customer names: 'Cust name39', 'Cust name38', 'Cust name37', 'Cust name36', 'Cust name35', 'Cust name34', 'Cust name33', 'Cust name32', 'Cust name31', 'Cust name30', 'Cust name29', 'Cust name28', 'Cust name27', 'Cust name26', 'Cust name25', 'Cust name24', 'Cust name23', 'Cust name22', 'Cust name21', 'Cust name20', 'Cust name19', 'Cust name18', 'Cust name17', 'Cust name16', 'Cust name15', 'Cust name14', 'Cust name13', 'Cust name12', 'Cust name11', 'Cust name10', 'Cust name9', 'Cust name8', 'Cust name7', 'Cust name6', 'Cust name5', 'Cust name4', 'Cust name3', 'Cust name2', 'Cust name1'. On the right, there is a 'Query tool' window with a 'Code' text area containing the SQL query: `SELECT (CUSTOMERNAME_01, orders(1) FROM (SELECT WHERE (CUSTOMERNAME_01) LIKE 'A' group by (CUSTOMERNAME_01) order by (CUSTOMERNAME_01) desc`. Below the code, there is a table with columns: Description, Type, Menu, Default value. It lists 'Customer name' with a default value of 'Cust name3%'. On the far right, there is a 'Results Display' window showing a list of customer names.

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

To process	Table	Index	Number of Lines	Automatic statistics	Statistics generated	Last analysis date
1	ABANK	ABANK_ABN0	17	Yes	Yes	05/05/23
2	ABICOND	ABICOND_ABN0	398	Yes	Yes	05/05/23
3	ABICOND	ABICOND_ABN1	398	Yes	Yes	05/05/23
4	ABICOND	ABICOND_ABN2	398	Yes	Yes	05/05/23
5	ABIGATMET	ABIGATMET_ABN0	16	Yes	Yes	05/05/23
6	ABIGATMET	ABIGATMET_ABN1	16	Yes	Yes	05/05/23
7	ABIGATMET	ABIGATMET_ABN2	1	Yes	Yes	05/05/23
8	ABIDIM	ABIDIM_ABN0	178	Yes	Yes	05/05/23
9	ABIDIMFLD	ABIDIMFLD_ABN0	2885	Yes	Yes	05/05/23
10	ABIDIMFLD	ABIDIMFLD_ABN1	2885	Yes	Yes	05/05/23
11	ABIDIMFLD	ABIDIMFLD_ABN2	2885	Yes	Yes	05/05/23
12	ABIHIERA	ABIHIERA_ABN0	96	Yes	Yes	05/05/23
13	ABIHIERA	ABIHIERA_ABN1	96	Yes	Yes	05/05/23
14	ABIRFLUSR	ABIRFLUSR_ABN0	5	Yes	Yes	05/05/23
15	ABIRFLUSR	ABIRFLUSR_ABN1	10156	Yes	Yes	05/05/23
16	ABIRFLUSR	ABIRFLUSR_ABN2	10156	Yes	Yes	05/05/23
17	ABIRFLUSR	ABIRFLUSR_ABN3	10156	Yes	Yes	05/05/23
18	ABIRFLUSR	ABIRFLUSR_ABN4	67	Yes	Yes	05/05/23

- Missing or added indexes

Statistics	Processes	Properties	Table Report
1	Number of indices described in the dictionary	+	3090
2	Number of optimisation indices described in the dictionary	+	0
3	Index number non-existent as Function:SQLDICO not active	-	1
4	Number of non-existent indices as tables not active	-	0
5	Number of non-existent optimisation indices (inactive)	-	0
6	Number of non-existent dictionary indices	-	0
7	Number of internal indices	+	0
8	Index number Sql server not described in the dictionary	+	0
9	Index number for long objects (Clob/Blob)	+	0
10	Number of indices found in the database Sql server	=	3089

- 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.

ID	Table	Index Name	Index Description	Active	Comment
1	SPES	SPES	SPES	Yes	
2	SPES	SPES	SPES	No	
3	SPES	SPES	SPES	No	
4	SPES	SPES	SPES	No	
5	SPES	SPES	SPES	No	
6	SPES	SPES	SPES	No	
7	SPES	SPES	SPES	No	
8	SPES	SPES	SPES	No	
9	SPES	SPES	SPES	No	
10	SPES	SPES	SPES	No	
11	SPES	SPES	SPES	No	
12	SPES	SPES	SPES	No	
13	SPES	SPES	SPES	No	
14	SPES	SPES	SPES	No	
15	SPES	SPES	SPES	No	
16	SPES	SPES	SPES	No	
17	SPES	SPES	SPES	No	
18	SPES	SPES	SPES	No	
19	SPES	SPES	SPES	No	
20	SPES	SPES	SPES	No	
21	SPES	SPES	SPES	No	
22	SPES	SPES	SPES	No	
23	SPES	SPES	SPES	No	
24	SPES	SPES	SPES	No	

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| Manufacturing] data model”

Summary

Introduction

How X3 generates SQL statements

A quick bit of “techy”

SQL you can use

Summary

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>

Thank you!