

Controlling Dynamic SQL with DSCC

By: Susan Lawson and Dan Luksetich



Controlling Dynamic SQL with DSCC

By: Susan Lawson and Dan Luksetich

In today's high performance computing environments we are bombarded with dynamic SQL more than we ever were in the past. Queries coming into our subsystems from products like DB2 Connect and Websphere are executed as dynamic SQL. This dynamic SQL creates problems such as unpredictable access paths, or security holes. For many years, some organizations have not allowed dynamic SQL to be used because of its unpredictable nature, and the potential problems it creates from a performance perspective. However, saying "no" to dynamic SQL may not be an option with today's applications.

Monitoring dynamic SQL presents a challenge. Traditional performance monitoring tools that we have used in the past are not able to produce the information we need, and in a manner that is helpful to our tuning efforts. We end up doing "on the fly" monitoring of dynamic SQL, resulting in an incomplete picture of the dynamic SQL in our environment, and an incomplete tuning effort.

There is a new tool in the DB2 marketplace that helps provide the information necessary for performance tuning in this environment, and does it without expense traces.

Dynamic SQL

Where is the majority of the dynamic coming from? Popular ERP (Enterprise Resource Planning) applications such as SAP, Peoplesoft, and Siebel use dynamic SQL. Even tools we have been using for years such as ODBC, QMF and SPUI use dynamic SQL. With the prevalence of script-based languages, Java JDBC, and built-in application database API's, the amount of dynamic SQL against our DB2 for z/OS databases is going to increase substantially.

There are other reasons why dynamic SQL is becoming more common in our applications today. Among these are flexible search screens. In situations where a search screen offers a complex array of input to search, dynamic SQL may be an alternative to many static cursors for each search variation. However, this enhanced and necessary flexibility can cause performance problems when the values become skewed. IBM provides features such as REOPT (VARS), or REOPT(ALWAYS/ONCE) in V8, that will give the optimizer the current and best information for access path selection. This re-optimization is done at run-time, and gives us an unpredicted access path that can change between executions. Additionally, there is a CPU cost associated with the mini-bind operation necessary for run time re-optimization.

Whether it is an ERP application, or our own dynamic application, we require the use of dynamic SQL. At the same time, we must be able to manage the unpredictable nature of the access paths and the resulting performance impact from the execution. It is vital to know the execution frequency of the dynamic statements, and the resources that are used

by particular statements. This enables us to focus our tuning efforts upon the significant performance issues.

The preparation of the dynamic SQL statement for execution may cause a great deal of overhead. For this reason, DB2 provides a facility called the dynamic statement cache. The cache is used to store dynamic SQL statements and their associated access paths.

The dynamic statement cache allows the reuse of an existing access path when a new incoming statement, with the same authorization id, matches one previously stored in the cache. The existence of a previously determined access path can allow it to be used without the overhead of re-preparing the statement. This saves a lot of CPU overhead for frequently executed dynamic SQL statements.

However, this is just one part of the solution to the performance issues of dynamic SQL. We still have to be concerned about the performance during execution - since in most cases, even though the statement is cached, we do not know its access path.

Performance traces are one way to get the information we need about the execution of a dynamic statement, but performance traces are not the perfect answer. First, not all of us have the resources or authorization to run a performance trace. Second, who has the time to analyze all the data produced from a performance trace? In DB2 V8, we can obtain access path information from the dynamic statement cache by “explaining” it if we have either the statement ID or token, but this two-step manual process only gives us the access path, and we still need better execution information. Additionally, explaining the dynamic statement cache only provides information about a specific entry in the dynamic statement cache. If there are many different processes using a similar SQL statement, but with different literal values, there will be multiple entries in the dynamic statement cache. This will result in explain information for all of these statements. Having a way to aggregate these executions, to get a more realistic view of statement executions, is critical for performance tuning.

DSCC – Dynamic SQL Cache Control

Now there is a software solution that provides a way for us to control this unpredictable environment. DSCC – Dynamic SQL Cache Control is a tool from Responsive Systems that provides a very easy way to view our dynamic SQL. It provides a way to analyze snapshots of the dynamic statement cache, which can be done for a specific period of time. The DSCC software is available for DB2 versions 7 and 8, and provides a capability known as Single Cache Extracts. These Cache Extracts show us detailed information about our dynamic SQL without having to run expensive traces. Through these extracts you can order the SQL statements to view the heaviest CPU consumers first, or those that have the highest number of getpages, or rows returned. Similar statements stored in the cache can be grouped or aggregated together to give a better overall view of the statements executed over a time period.

The Cache Extracts allow us analyze performance using perspectives that we have not been able to see previously, and to performance analysis that has not been possible in the

past, even with V8 of DB2. DSCC also provides the ability to capture views of the dynamic statement cache over time, allowing the history of the dynamic statements to be kept and evaluated.

Traditional monitors do not provide us with this ability because you have to view the statement while it is executing, and there is no easy way to collect the performance information over time. Using DB2 performance traces makes the understanding and tuning of dynamic SQL very difficult. DSCC removes the analysis problems by capturing the statements, aggregating them, comparing them over time, and presenting a realistic view of the dynamic environment to DBAs and application developers concerned with performance.

With the ability to collect dynamic statement cache history, we can better evaluate our tuning efforts for these statements to see if we are improving the access paths and execution events. Because we have the ability to capture statements from the cache over time, we can compare previous statement executions with newer executions. We can easily identify changes due to performance tuning efforts, or other environmental factors. This makes it easy for DBAs to justify their performance tuning efforts, and report on improvements to management. There are many optional thresholds that can be set within the product that allow the data to be presented in a way best suited for your tuning efforts (e.g. number of getpages).

Using DSCC

DSCC works by scanning the dynamic statement cache for one or many DB2 subsystems with its own capturing process. This process can be performed using filtering rules defined by the user, and stores the information in its own database. If necessary, the capturing process discards any unnecessary information. The DSCC tool can help you perform tasks such as problem solving for high CPU (i.e. during peak time), or elapsed time consumers for existing applications, or to evaluating the performance of a new application.

The cache capturing process can be run on a predetermined basis to see when problems are occurring, but it is more optimal to use this specifically for performance tuning efforts. This may be a bit of a different approach from traditional monitors. However, as we know, dealing with dynamic SQL performance itself requires a different approach. The past techniques and tools do not providing the details we need. DSCC can also be used to compare daily executions to ensure that abnormal increases in CPU are not occurring.

Cache Extracts

The cache extracts provided by DSCC are categorized in 4 ways: Time Analysis, Emergency Analysis, Group Accumulations, and Single Cache Extracts. All cache views provide an easy interface that lets the user organize the SQL statements in several different ways, and to sort the display by information that is relevant to the user.

Time Analysis

Time Analysis can be used to help determine the differences between two or more Single Cache Extracts or Group Accumulations. This can be done by using an older Single Cache View Extract or Group Accumulation as a starting point, and then comparing activity from that point forward. This is very helpful for viewing changes of activity during peak periods, or viewing the results of performance tuning efforts.

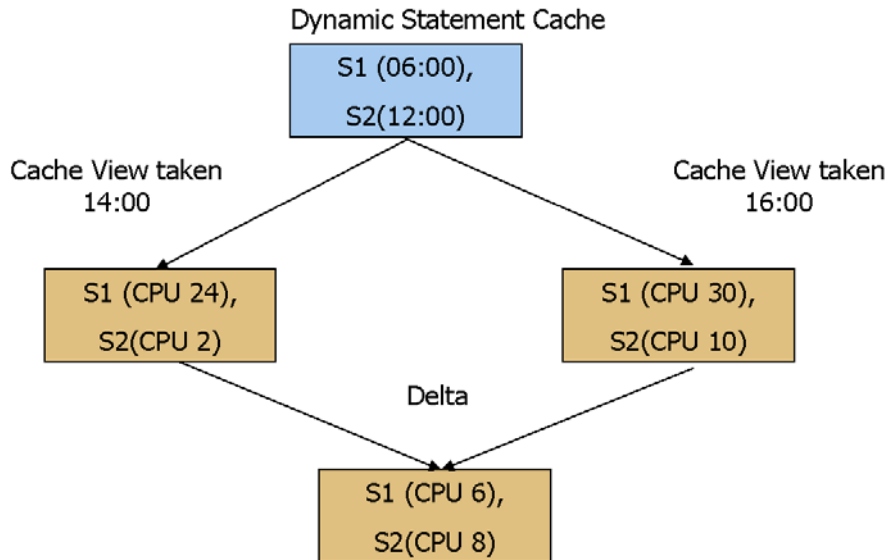


Figure 1. – Time Analysis Process

The time analysis process can also be used to track the execution of particular statements over time. This will enable the user to identify statements that are becoming progressively worse over time, allowing for a proactive tuning effort rather than a reactive effort. Statement degradation can be detected and corrected before it becomes a critical problem, enabling SLAs to be maintained, and user response times to remain consistent. There is an amazing value for being able to detect these degradations before they become an emergency. In addition to time analysis, DSCC also provides the ability to detect and report on problem SQL when users report performance problems - by using the Emergency Analysis capability.

In a Time Analysis – but also all other options – you can specify if you want to view all the individual caches, or you can aggregate the information by statement, user, transaction, program or tables. This aggregation allows the user to organize the performance data about dynamic statements in a manner that is meaningful to that user. Do you want to see single statement executions? All the statements accessing a specific table? All the statements issued by a specific user? How about all the statements issued by a transaction? All of this is possible within the Time Analysis or any other option within DSCC. This lifts the veil from the actions of our dynamic SQL applications, and their activity in the subsystem. The information for the each of the statements displayed will include the total CPU, elapsed time, number of executions, user, program, table and SQL type.

COMMAND ==>		SCROLL ==> CSR					
System ID: TIMEPRD DB2 : DB2X		Extract from: 2005.04.28 16:00.20					
		to : 2005.04.28 16:30.22					
Line Commands: PI Performance Information		T SQL Statement Text					
CI Client Information		TE / TV / TB Edit/View/Browse Text					
LC	Total CPU	Total Elapsed	Executed	Client	Package	Table	SQL Type
				*	*	*	*
	00:01:16.073744	00:01:43.291269		1	Z8AV001	SQLLF00 TZQWP1	SELECT
	00:00:25.849002	00:00:35.579057		1	Z8AV001	SQLLF00 TZQWP1	SELECT
	00:00:16.214618	00:00:20.835924		2	Z8AV001	SQLLF00 TZQWP1	SELECT
	00:00:14.777522	00:00:19.482550		2	Z8AV001	SQLLF00 TZQWP1	SELECT
	00:00:09.302420	00:00:11.741984		18	Z841468	IFDDB2E TIFC07	SELECT
	00:00:08.692256	00:00:22.636543		2	Z8AV001	SQLLF00 TQAP1	SELECT
	00:00:06.864766	00:00:08.230346		9	Z841468	IFDDB2E TIFD02	SELECT
	00:00:04.357856	00:00:15.298851		1	Z8RB002	P9RBID0 TRBCRP	SELECT
	00:00:03.914669	00:00:05.472271		9	Z8AZ001	SYSSH20 TQZBPL	SELECT
	00:00:03.801890	00:00:05.164712		18	Z841468	IFDDB2E TQC06	SELECT
	00:00:03.398091	00:00:40.017350		1	Z8RB002	P6RBED5 SEL1	SELECT
	00:00:03.164464	00:00:04.006261		18	Z841468	IFDDB2E TQFC96	SELECT
	00:00:02.226280	00:00:02.662589		3	Z8SD002	SQLLF00 JSDZ17	SELECT

Figure 2. – Time Analysis – SQL Information Display

The information in figure 2 shows the statements in the cache sorted in descending order by CPU Time consumed. The view information can be sorted by any of the column headers. By utilizing the *Performance Information* (PI) line command for any statement in the list, further performance information for a statement can be obtained. This information includes such items as wait time for synchronous I/O, getpage operations, number of rows examined, number of rows returned, and number of index scans or tablespace scans performed. Information that is very useful for understanding what the SQL statement is doing during execution, and to see how efficient the statement is. The actual SQL text that was executed can be viewed by using the *SQL Statement Text* (T) line command. You can also go into ISPF edit on the SQL text in order to edit or explain it, and test any improvements.

Emergency Analysis

Emergency Analysis cache extracts can be used for quick, fast analysis efforts. DSCC does this by comparing two Single Cache Extracts for short period of time. Emergency Analysis extracts are best utilized when an end user has reported a critical performance problem. All of the same statement, table, user, and aggregate statement information are available so that the problem statement, user, or table can be quickly identified.

Single Cache Extract

Single Cache Extracts can be used to capture and aggregate the information in the dynamic SQL cache for a particular subsystem. This feature can be used for preventative measures to determine if there is a problem SQL statement on a subsystem, and works best for monitoring SQL during a peak period. The Single Cache Extract can be used to

gather specific information about the SQL executions during a specific time frame, and can also be used to compare to other extracts to view differences.

Group Accumulations

Group Accumulations allow us to have single extracts grouped together by aggregating the information of multiple concurrent cache views taken from different DB2 subsystems according to user definitions. The same type of information and aggregation that can be done with the Single Cache Extracts can now be obtained at the specified group level. This is extremely useful in data sharing environments. The group view provides the ability to view the SQL statement cache in much the same way that the time analysis provides (as in figure 2). However, now the statements included span all of the subsystems, can be sorted across subsystems, and most importantly can be aggregated across subsystems. This gives the user a more global picture of the SQL statements executed across all subsystems, and provides a better overall view of the statements, tables, users, etc. that are the high resource consumers. Also, these Group Accumulations can be used to create a Time Analysis over a whole user defined set of DB2 subsystems within a data sharing environment.

Aggregate and Performance Information

One of the most powerful features of DSCC is its ability to aggregate similar SQL statements. This aggregated information provides a global picture of statement execution performance across multiple systems, multiple users, and multiple statements. Just as with all DSCC options, aggregated information can be organized and reported by statement, user, table, or transaction.

DSCC 1.4.1 - Aggregated SQL Overview				Row 1 from 1211
COMMAND ==>			SCROLL ==> CSR	
System ID: TIMEPRD DB2 : DB2X		Extract from: 2005.04.28 16:00.20 to : 2005.04.28 16:30.22		
Line Commands: PI Performance Information		T Show SQL Statement Text		
CE Cache Entries		TE / TV / TB Edit/View/Browse Text		
LC	Total CPU	Total Elapsed	Executed	CE SQL Type
-----	-----	-----	-----	-----
	00:01:16.073744	00:01:43.291269	1	1 SELECT
	00:00:25.849002	00:00:35.579057	1	1 SELECT
	00:00:19.140926	00:01:47.206934	29	25 SELECT
	00:00:18.642881	00:00:22.251533	29	25 SELECT
	00:00:16.214618	00:00:20.835924	2	1 SELECT
	00:00:14.777522	00:00:19.482550	2	1 SELECT
	00:00:09.302420	00:00:11.741984	18	1 SELECT
	00:00:08.692256	00:00:22.636543	2	1 SELECT
	00:00:08.439215	00:00:11.408840	33	22 SELECT
	00:00:06.999525	00:00:08.982909	10	9 SELECT
	00:00:06.864766	00:00:08.230346	9	1 SELECT
	00:00:06.346869	00:00:07.930730	10	9 SELECT
	00:00:05.969679	00:00:07.932119	8	7 SELECT

Figure 3. – Aggregated SQL Overview

Using this screen the user can display the aggregated SQL text (with literal values displayed as parameter markers), the performance information, or the individual cached entries for each aggregated statement. This enables the user to see the details of each literal form of the aggregated statement, as well as the aggregated statement. In this way, it can be determined if the overall statement has a performance problem, or if there is a problem related to specific users and/or literal values. Statement cache information can be aggregated by user, transaction, or program. This allows the user to identify all of the statements being issued by a particular user, transaction, or program. Not only can we identify potential SQL problems, we can extend that analysis to find particular users or applications that may be a continual source of performance problems.

DSCC 1.4.1 - Aggregated Client Overview				Row 1 from 72
COMMAND ==>				SCROLL ==> CSR
System ID: TIMEXPDR DB2 : DB2X		Extract from: 2005.04.28 16:00.20		
		to : 2005.04.28 16:30.22		
Line Commands: PI Performance Information				
CE Cache Entries				
LC	Total CPU	Total Elapsed	Executed	CE Client
	000:02:34.508671	000:04:33.339071	3720	454 QQAV001
	000:01:31.230539	000:04:24.678439	4237	1395 QQSD002
	000:00:29.522866	000:01:12.054566	36727	2420 QQAZ001
	000:00:23.259689	000:00:29.695189	108	24 Q041468
	000:00:21.934734	000:02:16.585934	249	212 QQRB002
	000:00:12.902253	000:01:38.421353	5697	1310 QQLT003
	000:00:08.398007	000:00:54.124907	4071	2439 QQB2E01
	000:00:06.058938	000:00:11.846638	505	28 QQPA009
	000:00:04.589340	000:00:09.404740	201	55 QQWGPS1
	000:00:03.503143	000:00:06.608743	307	306 QX05286
	000:00:02.445032	000:00:04.485632	8	5 Q184224
	000:00:02.278320	000:00:12.928220	13712	3 QQRC003
	000:00:01.859953	000:00:05.504353	12	9 QX06457

Figure 4. – Aggregated Client Overview

In figure 4 for example, client QQAV001 is using the most CPU, and perhaps worth some investigation.

Performance Information

Within each view the SQL statement, text can be displayed, with performance information about that SQL statement. This performance information gives a solid view of the performance impact of a particular SQL statement, and should form the basis for further investigation into statement performance problems. The performance information can be displayed for a particular statement or cache entry. It can also be displayed for aggregate statement information and also within group views, which gives the user a better overall perspective of the performance of a SQL statement across applications, users, and subsystems in a data sharing group. This gives an excellent and complete performance picture of all dynamic SQL.

```

DSCC 1.4.1 - Performance Information for Aggregated SQL
COMMAND ==>>                                SCROLL ==>> CSR

System ID: TIMEPRD DB2 : DB2X      Extract from   : 2005.04.28 16:00.20
                                     to               : 2005.04.28 16:30.22
Aggr. ID : 16.965.739              in Cache since : 2005.04.28 16:09:39
SQL Type : SELECT                   # Stmt execs   : 1
Total # Bytes in Cache : 623        # Cache entries : 1
-----
CPU Time .....:                Total           Average
Elapsed Time .....:            0:01:16.073744    0:01:16.073744
# Synchronous Buffer reads .....:                780
# Getpage Operations .....:            772.861      772.861
# Rows examined for Statement .....:          1.733.867    1.733.867
# Rows returned for Statement .....:             1.482      1.482
# Sorts performed for Statement ....:              2          2
# IDX Scans performed for Statement .:           147.823    147.823
# TS Scans performed for Statement .:              3          3

Wait Time Synchronous I/O .....:            0:00:01.120618    0:00:01.120618
Wait Time LOCK and LATCH request ...:            0:00:00.233216    0:00:00.233216
Wait Time Sync. Exec. Unit Switch ..:            0:00:00.000000    0:00:00.000000
Wait Time Global Locks .....:            0:00:00.000000    0:00:00.000000
Wait Time Read by another Thread ...:            0:00:00.828313    0:00:00.828313
Wait Time Write by another Thread ..:            0:00:00.000000    0:00:00.000000

*** End of Performance Information ***

```

Figure 5.– Performance Information Screen 1

```

Isolation Bind Option on init.prep. : CS (CURSOR STABILITY)
Currentdata Bind Option .....: NO
Dynamicrules Bind Option .....: RUN
Current Degree .....: 1
Current Rules .....: DB2
Current Precision .....: DEC15
Init. Prepare for CURSOR WITH HOLD .: YES

```

Figure 6 – Performance Information Screen 2

This detailed performance information allows the user to identify potential index screening issues by using the information about rows examined and processed (figure 5). Additionally, particular potential performance issues such as sorts, index and table space scans (figure 5). Bind parameters, such as the current data rules, and isolation level (figure 6), that are extremely important for performance, are now exposed. The only way to get this information previously was to ask the application developers or users.

Conclusion

In order to provide the best tuning possible for an environment with dynamic SQL, a specific kind of tool is required. DSCC is that tool. It provides a very easy way to get the information needed about the execution of dynamic SQL , and provides additional ways to see the effects of performance tuning efforts over time. The aggregation of statements

across time, users, programs, and tables provide information not available from DB2 or any other monitor product, and helps provide a global view of statement performance impact. All of this is done without expensive performance traces, and the information is presented in a very logical and useful manner. This tool certainly helps to bring control to a very unpredictable, and sometimes uncontrollable environment.

Authors: Susan Lawson and Dan Luksetich are DB2 performance consultants for YL&A. They both have worked with DB2 for over 15 years and have developed and tuned some of the world's largest and most complex DB2 database applications. They can be reached via email at susan_lawson@ylassoc.com and dan_luksetich@ylassoc.com. Or visit www.db2expert.com or www.ylassoc.com.

For additional information about the DSCC product please contact:

Responsive Systems Company
281 Highway 79
Morganville, NJ 07751

db2xpert@responsivesystems.com
(732) 972-1261
(800) DB2-EXPErt
(732) 972-9416 FAX