

Identifying the dynamic SQL statement which is causing a lock escalation in DB2® for z/OS®

Richard Corrihons
IBM Customer Center - PSSC
Montpellier, France

Table of contents

Disclaimer and Trademarks	1
Introduction	2
DB2 instrumentation enhancement	2
Detecting the lock escalation.....	3
Finding the SQL statement that generated the lock escalation	3
Finding the SQL statement text.....	5
Dynamic statement cache considerations.....	6
Conclusion.....	7
About the author.....	7

Disclaimer and Trademarks

The material in this document is subject to enhancements at some future date, a new release of DB2, or a Programming Temporary Fix.

The information contained in this document has not been submitted to any formal IBM review and is distributed on an "As Is" basis without any warranty either expressed or implied. The use of this information is a customer responsibility.

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries or both: DATABASE 2, DB2, OS/390, MVS/ESA, Redbooks, SYSTEM/390, IBM, Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, z/OS, zSeries, System z.

Other company, product or service names may be trademarks or service marks of others.

Introduction

Lock escalation is the promotion of a lock from a row, page or LOB lock to a table space lock because the number of page locks that are concurrently held on a given resource exceeds a preset limit.

Because fewer locks are taken, lock escalation is an effective way to reduce CPU cost and memory usage; however, because more resources are locked, concurrency can be dramatically reduced, and timeouts or deadlocks can result.

This document details a methodology you can use to identify the dynamic SQL statements involved in a lock escalation. It describes which traces to start, and which jobs to run to analyze the collected traces. It also explains how to analyze the trace report to find the culprit SQL statement. This methodology works for lock escalation that occurs with DB2® for z/OS V8 or above.

This paper is intended to be read by DB2® system administrators but may be of interest to any z/OS® performance specialist. It is assumed that the reader is familiar with DB2 performance but if that is not the case then the DB2 manual “Performance Monitoring and Tuning Guide” is recommended reading and can be found here:

<http://www.ibm.com/support/docview.wss?rs=64&uid=swg27011656>

DB2 instrumentation enhancement

DB2® for z/OS Version 8 introduced a new IFCID 337 to report on lock escalations. If activated, IFCID 337 is written whenever lock escalation occurs. The record contains the following information:

- Database ID
- Page set ID or table OBID
- Lock state that was escalated to (not used with Selective Partition Locking)
- Type of lower level lock used where we escalate from
- The number of lower level locks held that were released by escalation
- Statement number
- Waiters cached statement ID, or zero
- Collection ID (in EBCDIC or Unicode)
- Package name field (in EBCDIC or Unicode)

Note: The IFCID 337 record is part of class 3 of the statistics trace. Make sure this statistics trace class is started.

Detecting the lock escalation

There are several places where a lock escalation is reported.

- In the z/OS system log
- In the DB2 master address space job log

Let's take a look at the z/OS system log

```
M 80 MVL9 09070 11:15:22.87 S0000829 00000080 DSNI031I -DBT1 DSNILKES - LOCK ESCALATION HAS
D                               267 00000080 OCCURRED FOR
D                               267 00000080 RESOURCE NAME = DSN00333.UNIFIBAS
D                               267 00000080 LOCK STATE = X
D                               267 00000080 PLAN NAME : PACKAGE NAME = DSNRRSAF : SYSLH200
D                               267 00000080 COLLECTION-ID = NULLID
D                               267 00000080 STATEMENT NUMBER = 000001
D                               267 00000080 CORRELATION-ID = SEPACLEA
D                               267 00000080 CONNECTION-ID = RRSF
D                               267 00000080 LUW-ID = PSSCL.A6P2DBT1.C3DE56959320
E                               267 00000080 THREAD-INFO = CUSTOM1 : : :
```

Figure 1 z/OS system log

When a lock escalation occurs, DB2 writes messages with timestamps to the system log, as shown in figure 1. These messages provide some information regarding the resource on which the lock escalation happened and the threads associated with the event. However, there is no information regarding the SQL statement that generated the lock escalation. In the next section we describe a way to find that statement.

Finding the SQL statement that generated the lock escalation

On our system, the DB2 statistics trace class 3 is started by default; it is written to SMF. When we get a lock escalation, we run an SMF dump job to save the DB2 statistics records pertaining to the time window where the lock escalation occurred. We then run a locking trace report job against the saved SMF data using Tivoli® OMEGAMON® XE for DB2® Performance Expert on z/OS® batch processor. The parameters used to produce the report are shown in figure 2.

```
//SYSIN DD *
GLOBAL
  TIMEZONE(-1) (This value should be adjusted according to your settings)
LOCKING
  TRACE
  LEVEL(DETAIL)
EXEC
/*
```

Figure 2 Locking trace job sysin

The output of this locking trace job is shown in figure 3.

LOCATION: RDBNDBT0		OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)				PAGE: 1-1							
GROUP: N/P		LOCKING TRACE - DETAIL				REQUESTED FROM: NOT SPECIFIED							
MEMBER: N/P						TO: NOT SPECIFIED							
SUBSYSTEM: DBT1						ACTUAL FROM: 03/11/09 11:14:25.94							
DB2 VERSION: V9		SCOPE: MEMBER				PAGE DATE: 03/11/09							
PRIMAUTH	CORRNAME	CONNNTYPE	ORIGAUTH	CORRNMBR	INSTANCE	EVENT	TIMESTAMP	---		L O C K	R E S O U R C E	---	
PLANNNAME	CONNECT		RELATED	TIMESTMP	EVENT	TYPE	NAME	EVENT SPECIFIC DATA					
'BLANK'	'BLANK'	'BLANK'	11:15:22.873882	LOCK	ROW	(1)	DB=DSN00333	STATE	=N/P	NUMLOCKS=	1001		
'BLANK'	'BLANK'	C3DE56959320		ESCALATN		(2)	OB=UNIFIBAS	STMTNO = 1	(5)	WCSTMTID=X'0000002D'			
'BLANK'	'BLANK'					(3)		COLLID	=NULLID				
						(4)		PACKAGE	=SYSLH200				

Figure 3 Locking trace job output

We can see in figure 3 that there is information regarding the resource on which the lock escalation took place; DBID (1), OBID (2); as well as thread-related information like collection id (3) and package (4). We also get indirect information regarding the SQL statement, in the form of the SQL statement ID (WCSTMTID field (5)). The statement ID is an integer that uniquely identifies a statement that has been cached in the dynamic statement cache. The statement ID field is populated only when the executed SQL statement is a dynamic one. For the lock escalation event shown in figure 3, the statement ID is x'2D', that corresponds to 45 in decimal.

Another way to get the statement ID is to print IFCID 337 records; this can be achieved with a RECTRACE TRACE job as shown in figure 4. This job gives all the details regarding IFCID 337 records stored in the input file (SMF in our case).

```
//SYSIN DD *
GLOBAL
  TIMEZONE(-1) (This value should be adjusted according to your settings)
  RECTRACE
    TRACE
    INCLUDE(IFCID(337))
EXEC
/*
```

Figure 4 Record trace job sysin

The output of this RECTRACE TRACE job is shown in figure 5.

1 LOCATION: RDBNDBT0		OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V4)				PAGE: 1-2	
GROUP: N/P		RECORD TRACE - SHORT				REQUESTED FROM: NOT SPECIFIED	
MEMBER: N/P						TO: NOT SPECIFIED	
SUBSYSTEM: DBT1						ACTUAL FROM: 03/11/09 11:14:25.94	
DB2 VERSION: V9						PAGE DATE: 03/11/09	
PRIMAUTH	CONNECT	INSTANCE	END_USER	WS_NAME	TRANSACT		
ORIGAUTH	CORRNAME	CONNNTYPE	RECORD TIME	DESTNO ACE IFC	DESCRIPTION DATA		
PLANNNAME	CORRNMBR		TCB CPU TIME	ID			
N/P	N/P	C3DE56959320	N/P	N/P	N/P		
N/P	N/P	'BLANK'	11:15:22.87388225	27 1 337	LOCK ESCAL.	NETWORKID:PSSCL LUNAME:A6P2DBT1	
N/P	N/P		N/P		OCCURENCES		
(A)	DATABASE ID : DSN00333		(B)	PAGESET/TABLE ID : UNIFIBAS		STATEMENT NUMBER : 1	
(C)	WAITERS STATMT ID : 45		LOWER LOCK TYPE : ROW LOCK		NUMBER LOWER LOCKS: 1001		
(D)	COLLECTION ID : NULLID						
(E)	PACKAGE NAME : SYSLH200						

Figure 5 Record trace job report output

We can see in figure 5 that there is information regarding the resource on which the lock escalation took place; DBID **(A)**, PAGESET/TABLE ID **(B)**; as well as thread-related information like collection name **(D)** and package name **(E)**. As in figure 3, we also get indirect information regarding the SQL statement, in the form of the SQL statement ID (WAITERS STATMT ID field **(C)**). This time it is shown in decimal (45) as opposed to figure 3 where it is shown in hexadecimal.

Now that we have the statement ID, we need to get the statement itself. To do this we need to extract the content of the global dynamic statement cache. There are several ways to proceed; one way is detailed in the next section.

Finding the SQL statement text

In this section we describe the steps required to get the SQL statement text that corresponds to a given statement ID.

This can be achieved with the help of the EXPLAIN STMTCACHE ALL command. The sysin of a DSNTEP2 job that perform this explain is shown in figure 6.

```
//SYSIN DD *  
SET CURRENT SQLID='SYSADM' ;  
EXPLAIN STMTCACHE ALL ;  
/*
```

Figure 6 DSNTEP2 job sysin

The EXPLAIN STMTCACHE ALL command returns one row for each cached statement to the table DSN_STATEMENT_CACHE_TABLE (the statement cache table). These rows contain information identifying the statements in the cache, and statistics on the execution of the statements by all processes that have executed the statement.

In this table, the STMT_ID column is the statement ID (that is an integer that uniquely identifies a statement that has been cached in the dynamic statement cache) and the STMT_TEXT column is the SQL statement text itself. This column is a CLOB of 2 megabytes, which is the current maximum possible length for an SQL statement. This means the SQL text stored in the STMT_TEXT column does not get truncated when it is extracted from the dynamic statement cache.

Information about creating a statement cache table can be found in the EXPLAIN section of the SQL reference book. This book can be found here:

<http://www.ibm.com/support/docview.wss?rs=64&uid=swg27011656>

Once the statement cache table has been populated, we use the following SQL to retrieve the statement which is causing the lock escalation.

```
Select STMT_ID, STMT_TEXT from DSN_STATEMENT_CACHE_TABLE  
where STMT_ID = 45 ;
```

The statement is shown in figure 7.

```
STMT_ID: 45  
STMT_TEXT: update sepaclea.UNIFIBASE set status=? ,ourmsgid=? where  
fileout=?
```

Figure 7 Statement ID and corresponding SQL statement which is causing the lock escalation

With the statement text we have the information the application developers and the database administrator need to be able to correct our lock escalation problem, if they consider it needs to be fixed.

Dynamic statement cache considerations

In order to use the technique described in this document, you need to be aware of the following considerations regarding global dynamic statement cache:

- Global dynamic statement caching must be active; the DYNCACH parameter must be set to YES in DB2 system DSNZPARM (this has been the default, for several DB2 for z/OS versions).
- An SQL Statement must not have been removed from the global cache, before the EXPLAIN STATEMENT ALL command has been executed. The following events cause a statement to be removed from the dynamic statement cache:
 - No free pages are available in the EDM pool. The statements are discarded on an LRU basis.
 - DROP, ALTER, REVOKE executed for anything the access path is depending on.
 - RUNSTATS executed for objects, the statement is depending on.
 - DB2 has been recycled (stopped then restarted)
- Because of the reasons detailed above, when a lock escalation occurs we must extract the statement cache *before* the statements get removed from it. In a production environment it is recommended to use the automation tools in place to trigger a job that issues an EXPLAIN STMTCACHE ALL command when a lock escalation message (DSNI031I) is captured by the automation processes.
- For the EXPLAIN STMTCACHE ALL command, SYSADM authority is required to explain all statement in the dynamic statement cache. If the user does not have SYSADM authority, only those statements that have the same authorization ID as the user are explained
- Statements in plans or packages bound with REOPT(ALWAYS) are not cached in the dynamic statement cache.
- There is one dynamic statement cache per DB2 member. In case of data sharing, you have to insure you are extracting the dynamic statement cache for the DB2 member that issued the SQL statement.

Conclusion

In this document we described how to benefit from DB2 V8 for z/OS enhancements to identify the dynamic SQL statement involved in a lock escalation. We saw that the LOCKING TRACE command as well as the RECTRACE TRACE command allow us to retrieve the statement ID of the SQL, and that the EXPLAIN STMTCACHE ALL command allows us to retrieve the SQL statement text that correspond to that statement ID.

About the author

Richard Corrhons has been a DB2 for z/OS Performance Specialist with the Mainframe Benchmark Center in IBM Montpellier, France for 10 years. Richard specializes in working with customers on proof of concepts and benchmarks, evaluating the performance of their DB2 implementation, and making recommendations on how to improve their solution. Richard is co-author of a number of IBM Redbooks® publications and papers on DB2-related topics.

Richard's recent papers:

Maximizing offload to zIIP processors with DB2 9 for z/OS native SQL stored procedures.
Download URL: <http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/TD104524>

Identifying the dynamic SQL statements which are causing a deadlock in DB2 for z/OS.
Download URL: <http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/TD104962>