Rule DB2-305: Lock escalation was not effective

Finding: Lock escalation was not effective because a relatively large number of lock escalations resulted in timeout suspensions or deadlocks.

Impact: This finding can have a MEDIUM IMPACT, or HIGH IMPACT on the performance of the DB2 subsystem.

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

Lock escalation releases a large number of page or row locks, held by an application process on a single table or table space, to acquire a table or table space lock, or a set of partition locks, of mode S or X.

Lock escalation balances concurrency with performance by using page or row locks while a process accesses relatively few pages or rows, then changing to table space, table, or partition locks when the process accesses many pages or rows.

- In individual application processes lock only at the row or page level, other application processes can access the non-locked parts of the tablespace or table. This allows concurrency of access.

- Page or row locking uses *significant* processing time. Once the performance cost of locking is unacceptable, DB2 changes (or escalates) the locks to lock the entire table space, table, or partition (thus holding a single lock) and releases all the individual page or row locks. This lock escalation can reduce concurrency because other applications *may* not be able to access the table space, table, or partition (depending on the locking required). However, the lock escalation improves performance because the overhead of managing many locks is exchanged for the overhead of managing only a single lock.

IBM states that, as a rough estimate, lock escalation is not effective if more than one quarter of the lock escalations cause timeouts or deadlocks.

CPExpert computes the percent of ineffective lock escalations (PCTESC) by the following algorithm:

$$PCTESC = \frac{Timeouts + Deadlocks}{Shared\ locks\ escalated + Exclusive\ locks\ escalated}$$

CPExpert compares the computed PCTESC with the **PCTESC** guidance variable in USOURCE(DB2GUIDE). CPExpert produces Rule DB2-305 when the percent of lock escalations that were not effective exceeds the value specified by the **PCTESC** guidance variable.

The default value for the **PCTESC** guidance variable is 25, indicating that CPExpert should produce Rule DB2-305 when more than 25% of the lock escalations were not effective.

The following example illustrates the output from Rule DB2-305:

```
RULE DB2-305: LOCK ESCALATION WAS NOT EFFECTIVE

   Lock escalation releases a large number of page or row locks, held by
   an application process on a single table or table space, to acquire a
   table or table space lock, or a set of partition locks, of mode S or X.
   Lock escalation balances concurrency with performance by using page or
   row locks while a process accesses relatively few pages or rows, then
   changing to table space, table, or partition locks when the process
   accesses many pages or rows.  IBM states that, as a rough estimate,
   escalation is not effective if more than one quarter of the lock
   escalations cause timeouts or deadlocks.  CPExpert detected that more
   than 25% of the escalations caused timeouts or deadlocks, during the
   intervals shown below:

                              NUMBER OF LOCKS       TIMEOUTS OR
   MEASUREMENT INTERVAL         ESCALATED           DEADLOCKS     PERCENT
    2:47- 3:17, 16SEP1999             2                   1        50.0
   11:15-11:45, 16SEP1999             2                   1        50.0
   11:45-12:15, 16SEP1999             2                   1        50.0
```

Suggestion: If Rule DB2-305 is produced regularly, CPExpert suggests that you consider the following alternatives:

- You might alter the table to increase the LOCKMAX specification. This would decrease the number of escalations. You should set the value of LOCKMAX high enough that, when lock escalation occurs, one application already holds so many locks that it significantly interferes with others. For example, if an application holds half a million locks on a table with a million rows, it probably already locks out most other applications. Yet lock escalation can prevent it from potentially acquiring another half million locks.

- You might let the process that is causing the lock escalations initially lock the entire table space, using the statement LOCK TABLE. This action would prevent concurrency, but it is a reasonable solution for some

end-of-month or end-of-year situations when a process updates more pages than it normally does.

- You can alter CPExpert's analysis by modifying the **PCTESC** guidance variable in USOURCE(DB2GUIDE).

Reference: DB2 for OS/390 Version 3: Administration Guide
Section 2.4.1.23 (IRLM Panel 2: DSNTIPJ)
Section 5.7.4.5.3 (Lock Escalation)

DB2 for OS/390 Version 4: Administration Guide
Section 5.7.4.5.3 (Lock Escalation)
Section 5.7.6.3.3 (LOCKMAX Clause)

DB2 for OS/390 Version 5: Administration Guide
Section 5.7.4.5.3 (Lock Escalation)
Section 5.7.5.3.3 (LOCKMAX Clause)

DB2 for OS/390 Version 6: Administration Guide
Section 5.7.4.5.3 (Lock Escalation)
Section 5.7.5.3.3 (LOCKMAX Clause)

DB2 UDB for OS/390 and z/OS, Version 7: Administration Guide
Section 5.7.4.5.3 (Lock Escalation)
Section 5.7.5.3.3 (LOCKMAX Clause)

DB2 UDB for z/OS Version 8: Administration Guide
Chapter 30. Improving concurrency
Aspects of Transaction Locks
Lock escalation
Other options that affect locking

DB2 UDB for z/OS Version 9: Performance Monitoring and Tuning Guide
Chapter 8. Improving concurrency
Aspects of Transaction Locks
Lock Escalation

DB2 10 for z/OS: Managing Performance
Chapter 27. Programming for concurrency
Options for tuning locks

DB2 11 for z/OS: Managing Performance
Chapter 17. Concurrency and locks
Lock escalation

