www.DBTechNet.org

On SQL Concurrency Technologies
- for Application Developers

With the support of the EC LLP Transversal programme of the European Union

Martti Laiho[1], Fritz Laux[2]

[1] Dpt. of Business Information Technology, Haaga-Helia University of Applied Sciences, Ratapihantie 13, 00520 Helsinki, Finland

[2] Dpt. of Informatics, Reutlingen University, Alteburgstraße 150, 72762 Reutlingen, Germany,

Disclaimers

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Trademarks of products mentioned are trademarks of the product vendors.

Appendix 3 Concurrency Tests using DB2 LUW 9.7

Let's now experiment with DB2 applying the same tests as we used for SQL Server.

-- We start the test using session in Command Editor:

```
DROP TABLE T;
CREATE TABLE T (
id   INT NOT NULL PRIMARY KEY,
s    CHAR(10)
);
INSERT INTO T VALUES (1,'a');
INSERT INTO T VALUES (2,'b');
INSERT INTO T VALUES (3,'c');
COMMIT;
```

-- First test.

-- Note: The default isolation of 9.7 is CC, but it cannot be set and cannot be seen (?):

```
------------------------------ Commands Entered ------------------------------
VALUES CURRENT ISOLATION;
DECLARE cur1 CURSOR
   FOR SELECT s FROM T;
OPEN cur1;
FETCH FROM cur1;
------------------------------ Results ---------------------------------------
```

```
VALUES CURRENT ISOLATION

1
--

DECLARE cur1 CURSOR FOR SELECT s FROM T
DB20000I  The SQL command completed successfully.

OPEN cur1
DB20000I  The SQL command completed successfully.

FETCH FROM cur1

S
----------
a

  1 record(s) selected.
```

-- Monitoring the locks in a concurrent CLP session:

```
db2 CONNECT TO TEST
db2 "SELECT CAST(AGENT_ID AS SMALLINT) sid,SUBSTR(LOCK_OBJECT_TYPE,1,10)as
lock_type, LOCK_CURRENT_MODE AS
CURRENT_MODE,LOCK_MODE,LOCK_STATUS,SUBSTR(TABNAME,1,10) table,
CAST(LOCK_HOLD_COUNT AS SMALLINT) lock_holds FROM SYSIBMADM.SNAPLOCK WHERE
LOCK_OBJECT_TYPE NOT LIKE 'INTERNAL%%'"

SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------
     7 TABLE_LOCK -            IS         GRNT        T                   0

  1 record(s) selected.
```

-- Continuing the test in Command Editor:

```
------------------------------ Commands Entered ------------------------------
CLOSE cur1;
ROLLBACK;
SET ISOLATION TO UR;
VALUES CURRENT ISOLATION;
DECLARE cur1 CURSOR
   FOR SELECT s FROM T;
OPEN cur1;
FETCH FROM cur1;
------------------------------ Results ---------------------------------------

CLOSE cur1
DB20000I  The SQL command completed successfully.

ROLLBACK
DB20000I  The SQL command completed successfully.

SET ISOLATION TO UR
DB20000I  The SQL command completed successfully.

VALUES CURRENT ISOLATION

1
--
UR

  1 record(s) selected.
```

```
DECLARE cur1 CURSOR FOR SELECT s FROM T
DB20000I  The SQL command completed successfully.

OPEN cur1
DB20000I  The SQL command completed successfully.

FETCH FROM cur1

S
----------
a

  1 record(s) selected.
```

-- Repeating the monitoring of locks in the concurrent CLP session:

```
SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------
     7 TABLE_LOCK -            IN         GRNT        T                   0

  1 record(s) selected.

-- Note:  IN lock means "Intent None" lock on table
-- Lock owner can read even uncommitted data without row locks
-- but cannot change this data.
```

-- Continuing the test in Command Editor:

```
------------------------------ Commands Entered ------------------------------
CLOSE cur1;
ROLLBACK;
SET ISOLATION TO CS;
VALUES CURRENT ISOLATION;
DECLARE cur1 CURSOR
   FOR SELECT s FROM T;
OPEN cur1;
FETCH FROM cur1;
------------------------------------------------------------------------------
CLOSE cur1
DB20000I  The SQL command completed successfully.

ROLLBACK
DB20000I  The SQL command completed successfully.

SET ISOLATION TO CS
DB20000I  The SQL command completed successfully.

VALUES CURRENT ISOLATION

1
--
CS

  1 record(s) selected.


DECLARE cur1 CURSOR FOR SELECT s FROM T
DB20000I  The SQL command completed successfully.

OPEN cur1
DB20000I  The SQL command completed successfully.
```

```
FETCH FROM cur1

S
----------
a

  1 record(s) selected.
```

-- Repeating the monitoring of locks in the concurrent CLP session:

```
SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------
     7 TABLE_LOCK -            IS         GRNT        T                   0

  1 record(s) selected.

-- Note: Intent Share (IS) lock on table T in case of Cursor Stability (CS)
-- Lock owner can read committed data without row locks
-- but cannot change this data.
```

-- Continuing the test in Command Editor:

```
------------------------------ Commands Entered ------------------------------
CLOSE cur1;
ROLLBACK;
SET ISOLATION TO CS;
VALUES CURRENT ISOLATION;
SELECT s FROM T;
------------------------------------------------------------------------------
CLOSE cur1
DB20000I  The SQL command completed successfully.

ROLLBACK
DB20000I  The SQL command completed successfully.

SET ISOLATION TO CS
DB20000I  The SQL command completed successfully.

VALUES CURRENT ISOLATION

1
--
CS

  1 record(s) selected.


SELECT s FROM T

S
----------
a
b
c

  3 record(s) selected.
```

-- Repeating the monitoring of locks in the concurrent CLP session:

```
SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------

  0 record(s) selected.
```

```
-- Note: Locks on rows have been released
-- in case of Cursor Stability (CS) without any active cursors
```

-- Continuing the test in Command Editor:

```
------------------------------ Commands Entered ------------------------------
ROLLBACK;
SET CURRENT ISOLATION TO RS;
VALUES CURRENT ISOLATION;
SELECT s FROM T;
------------------------------------------------------------------------------
ROLLBACK
DB20000I  The SQL command completed successfully.

SET CURRENT ISOLATION TO RS
DB20000I  The SQL command completed successfully.

VALUES CURRENT ISOLATION

1
--
RS

  1 record(s) selected.


SELECT s FROM T

S
----------
a
b
c

  3 record(s) selected.
```

-- Repeating the monitoring of locks in the concurrent CLP session:

```
SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------
     7 ROW_LOCK   -            NS         GRNT        T                   0
     7 ROW_LOCK   -            NS         GRNT        T                   0
     7 ROW_LOCK   -            NS         GRNT        T                   0
     7 TABLE_LOCK -            IS         GRNT        T                   0

  4 record(s) selected.

-- In case of Read Stability (RS):
-- NS-locks on rows, and Intent Share (IS) lock on table T
-- Lock owner can read committed data with row locks
```

-- Continuing the test in Command Editor:

```
------------------------------ Commands Entered ------------------------------
ROLLBACK;
SET CURRENT ISOLATION TO RR;
VALUES CURRENT ISOLATION;
SELECT s FROM T;
------------------------------------------------------------------------------
ROLLBACK
DB20000I  The SQL command completed successfully.
```

```
SET CURRENT ISOLATION TO RR
DB20000I  The SQL command completed successfully.

VALUES CURRENT ISOLATION

1
--
RR

  1 record(s) selected.


SELECT s FROM T

S
----------
a
b
c

  3 record(s) selected.
```

-- Repeating the monitoring of locks in the concurrent CLP session:

```
SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------
     7 TABLE_LOCK -            S          GRNT        T                   0

  1 record(s) selected.

-- Share (S) lock on table T
-- Lock owner can read committed data without row locks
```

Scenario 2: Contest of multiple UPDATEs made in different order

Please verify yourself the following test, in which we apply the contest scenario 2 of Appendix 1 in SQL Server concurrent sessions A and B

-- We start the test using session in Command Editor:

```
CREATE TABLE Test (
id INT NOT NULL PRIMARY KEY,
s VARCHAR(20),
n INT DEFAULT 0
) ;
INSERT INTO Test (id,s,n) VALUES (1,'some text',0);
INSERT INTO Test (id,s,n) VALUES (2,'some text',0);
COMMIT;
```

-- step 1 – Starting session A in CLP

```
db2 +c
db2 => CONNECT TO TEST
db2 => UPDATE Test SET s='new value1',n=n+1 WHERE id=1
```

-- step 2 - Starting session B in concurrent CLP

```
db2 +c
db2 => CONNECT TO TEST
db2 => UPDATE Test SET s='new value2',n=n+2 WHERE id=2
```

-- step 3 - Session A

```
db2 => UPDATE Test SET s='new value3',n=n+1 WHERE id=2
(starts waiting …)
```

-- step 4 - Session B, let's see the locks first

```
db2 => SELECT CAST(AGENT_ID AS SMALLINT) sid,SUBSTR(LOCK_OBJECT_TYPE,1,10)as
lock_type, LOCK_CURRENT_MODE AS CURRENT_MODE, LOCK_MODE, LOCK_STATUS,
SUBSTR(TABNAME,1,10) table, CAST(LOCK_HOLD_COUNT AS SMALLINT) lock_holds FROM
SYSIBMADM.SNAPLOCK WHERE LOCK_OBJECT_TYPE NOT LIKE 'INTERNAL%%'

SID    LOCK_TYPE  CURRENT_MODE LOCK_MODE  LOCK_STATUS TABLE      LOCK_HOLDS
------ ---------- ------------ ---------- ----------- ---------- ----------
    28 ROW_LOCK   -            X          GRNT        TEST                0
    28 TABLE_LOCK -            IX         GRNT        TEST                0
    21 ROW_LOCK   -            X          GRNT        TEST                0
    21 TABLE_LOCK -            IX         GRNT        TEST                0

  4 record(s) selected.
```

Unfortunately we did not manage to see details of session A waiting for the lock of session B, by using the following query:

```
db2 => select AGENT_ID, LOCK_MODE,LOCK_OBJECT_TYPE,AGENT_ID_HOLDING_LK,
LOCK_WAIT_START_TIME, substr(TABNAME,1,6) as tabname from
SYSIBMADM.SNAPLOCKWAIT

  0 record(s) selected.


db2 => UPDATE Test SET s='new value4', n=n+2 WHERE id=1
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0911N  The current transaction has been rolled back because of a deadlock
or timeout.  Reason code "2".  SQLSTATE=40001
```

-- step 5 - Session B, following is an empty transaction

```
db2 => COMMIT
DB20000I  The SQL command completed successfully.
```

-- step 6 - Session A

```
db2 => COMMIT
DB20000I  The SQL command completed successfully.
db2 => SELECT * FROM Test
ID          S                    N
----------- -------------------- -----------
          1 new value1                     1
          2 new value3                     1

  2 record(s) selected.
```