

## DB2 Version 8 - Concurrency

DB2 Quickstart Education

Maintained by Paul Yip (ypaul@ca.ibm.com)

February 2003

IBM Software Group

## Possible Concurrent "Situations"

### Lost Update

App1 updates a row  
App2 updates the same row  
App1 commits  
App2 commits  
What happened to App1's update?

### Nonrepeatable Read

App1 opens a cursor  
App2 deletes row that qualified for cursor  
App2 commits changes  
App1 closes and reopens cursor  
Does App1 need the same data on successive fetches?

### Uncommitted Read

App1 updates a row  
App2 reads the new value from that row  
App1 rolls back it's changes to that row  
Is the data App1 is using still valid?

### Phantom Read

App1 opens a cursor  
App2 adds a row to the database that would qualify for the cursor  
App2 commits changes  
App1 closes and reopens cursor  
Can App1 handle new rows in successive fetches?

## Isolation Levels

- DB2 provides different levels of protection to isolate data
  - ▶ Uncommitted Read (UR)
  - ▶ Cursor Stability (CS)
  - ▶ Read Stability (RS)
  - ▶ Repeatable Read (RR)
- Cursor Stability is the default isolation level
- Isolation level can be specified at many levels
  - ▶ Session (application)
  - ▶ Connection
  - ▶ Statement
- For embedded SQL, the level is set at bind time
- For dynamic SQL, the level is set at run time

## Isolation Levels - Uncommitted Read

- Uncommitted Read is also known as DIRTY READ
- Lowest level of isolation
- Provides highest degree of concurrency
  - ▶ no row locks are obtained on read operations
    - unless other application attempts to drop or alter table
  - ▶ update operations act as if using Cursor Stability
- Possible Situations
  - ▶ Uncommitted Read (duh)
  - ▶ Nonrepeatable Read
  - ▶ Phantom Read
- Situations Prevented
  - ▶ Loss of Update

## Isolation Levels - Cursor Stability

- Cursor Stability is the default isolation level
  - ▶ minimal degree of locking
- Locks the "current" row of a cursor
- If the row is only read
  - ▶ the lock is held until a new row is fetched or the unit of work is terminated
- If the row is updated
  - ▶ the lock is held until the unit of work is terminated
- Possible Situations
  - ▶ Nonrepeatable Read
  - ▶ Phantom Read
- Prevented Situations
  - ▶ Loss of Update
  - ▶ Uncommitted Read

## Isolation Levels - Read Stability

- Locks all the rows an application retrieves within a unit of work
  - ▶ for a given cursor, it lock all rows that qualify for a result set
  - ▶ moderate degree of locking
- Possible Situations
  - ▶ Phantom Read
- Prevented Situations
  - ▶ Loss of Update
  - ▶ Uncommitted Read
  - ▶ Nonrepeatable Read

## Isolation Levels - Repeatable Read

- Highest isolation level, least concurrency
  - ▶ Same query issued by the application more than once in a unit of work will give the same result each time
  - ▶ high degree of locking
- Locks held on all rows processed to build the result set
  - ▶ i.e. rows not necessarily in the final result set may be locked
- No other application can update, delete, or insert a row that would affect the result set until the unit of work completes
- Possible Situations
  - ▶ none
- Prevented Situations
  - ▶ Loss of Update
  - ▶ Uncommitted Read
  - ▶ Nonrepeatable Read
  - ▶ Phantom Read

## Locking - Description


- Locking is controlled by the isolation level
- By default, DB2 uses row level locking
- Database, table spaces, and tables can be explicitly locked
  - ▶ Database lock  
CONNECT TO dbname IN EXCLUSIVE MODE
  - ▶ Table space lock  
QUIESCE TABLESPACES FOR TABLE tablename INTENT FOR UPDATE
  - ▶ Table lock  
LOCK TABLE tablename IN EXCLUSIVE MODE
- Database, tables, and rows can be implicitly locked
  - ▶ Database lock
    - During full database restore
  - ▶ Table lock
    - Through lock escalation or at the discretion of the optimizer
  - ▶ Row lock
    - Through normal data modification as directed by the access plan

## Table Locking - Reading

- IN - Intent None
  - ▶ owner of the lock can read any data, committed or uncommitted, in the table
  - ▶ other applications can read or update the table
- IS - Intent Share
  - ▶ owner of the lock can read any data in the table and obtains an S or NS lock on each row read
  - ▶ other applications can read or update rows in the table
- S - Share
  - ▶ owner of the lock can read any data in the table and will not obtain row locks
  - ▶ other applications can read (not update) the table data

## Table Locking - Writing

- IX - Intent to Change
  - ▶ owner of the lock can read any data in the table if a U, S, NS, or X lock can be obtained on rows
  - ▶ owner can change any data in the table if a lock can be obtained on rows
- U - Update
  - ▶ owner of the lock can read any data in the table and can change data if an X lock on the table can be obtained prior to the update
  - ▶ other applications can only read the table data.
- X - Changed
  - ▶ owner of the lock can read or update any data in the table
  - ▶ no row locks are obtained
  - ▶ only other applications using UR can read rows



## Row Locking

---

- S - Share
  - ▶ owner of the lock can read but not update the locked data
  - ▶ other applications can read, but not update, the locked data.
- U - Update
  - ▶ row is being read by one application with intent to update. It is available for read-only by concurrent applications. The lock owner will acquire X locks on the rows prior to update
- X - Changed
  - ▶ The row is changed and is not available for concurrent applications, except for those with UR.



## LOCK Manager parameters

---

- LOCKLIST
  - ▶ Max storage for lock list (4KB)
- MAXLOCKS
  - ▶ Percent. of lock lists per application

## Lock Escalation (BAD)

- A record of each lock is kept in the LOCKLIST
  - ▶ size set by LOCKLIST database configuration parameter
- Each DB2 lock requires 36 or 72 bytes of memory
  - ▶ 72 bytes if there is only one lock on an object
  - ▶ 36 bytes if there is more than one lock
- Each application is only allowed a percentage of the list
  - ▶ this percentage is set by the MAXLOCKS database parameter
- If the lock list gets full or an application reaches MAXLOCKS
  - ▶ The DB manager may escalate multiple row locks in the same table into a single table lock
    - ─ Reduces resource contention
    - ─ Decreases concurrency
    - ─ Increases chances of deadlock

DB2 Data Management Software



## Lock escalation indication in db2diag.log

2001-10-02-23.04.43.699000 Instance:DB2 Node:000  
PID:984(db2syscs.exe) TID:1720 Appid:\*LOCAL.DB2.011003030417  
data\_management sqldEscalateLocks Probe:1 Database:SAMPLE

-- Start Table Lock Escalation.

-- Lock Count, Target : 28, 14

7570 6461 7465 2065 6d70 6c6f 7965 6520  
7365 7420 6669 7273 746e 6d65 3d27 6162  
6327 c'

update employee  
set firstnme='ab'

2001-10-02-23.04.43.699001 Instance:DB2 Node:000  
PID:984(db2syscs.exe) TID:1720 Appid:\*LOCAL.DB2.011003030417  
data\_management sqldEscalateLocks Probe:2 Database:SAMPLE

-- Lock Count, Target : 28, 14

-- Table (ID) Name : (2;5) ADMINISTRATOR.EMPLOYEE

-- Locks, Request Type : 25, X

-- Result (0 = success): 0

## Lock Snapshot

- View locks currently held by an application
- UPDATE MONITOR SWITCHES USING LOCK ON
- GET SNAPSHOT FOR LOCKS FOR APPLICATION AGENT ID <handle>

DB2 Data Management Software

### Application Lock Snapshot

Snapshot timestamp = 11-05-2002 00:09:08.672586

Application handle = 9  
Application ID = \*LOCAL.DB2.00B9C5050843  
Sequence number = 0001  
Application name = db2bp.exe  
Authorization ID = ADMINISTRATOR  
Application status = UOW Waiting  
Status change time = Not Collected  
Application code page = 1252  
Locks held = 4  
Total wait time (ms) = 0

#### List Of Locks

Lock Name = 0x050007000480010000000000052  
Lock Attributes = 0x00000000  
Release Flags = 0x40000000  
Lock Count = 255  
Hold Count = 0  
Lock Object Name = 98308  
Object Type = Row  
Tablespace Name = TEST4K  
Table Schema = ADMINISTRATOR  
Table Name = T2  
Mode = X