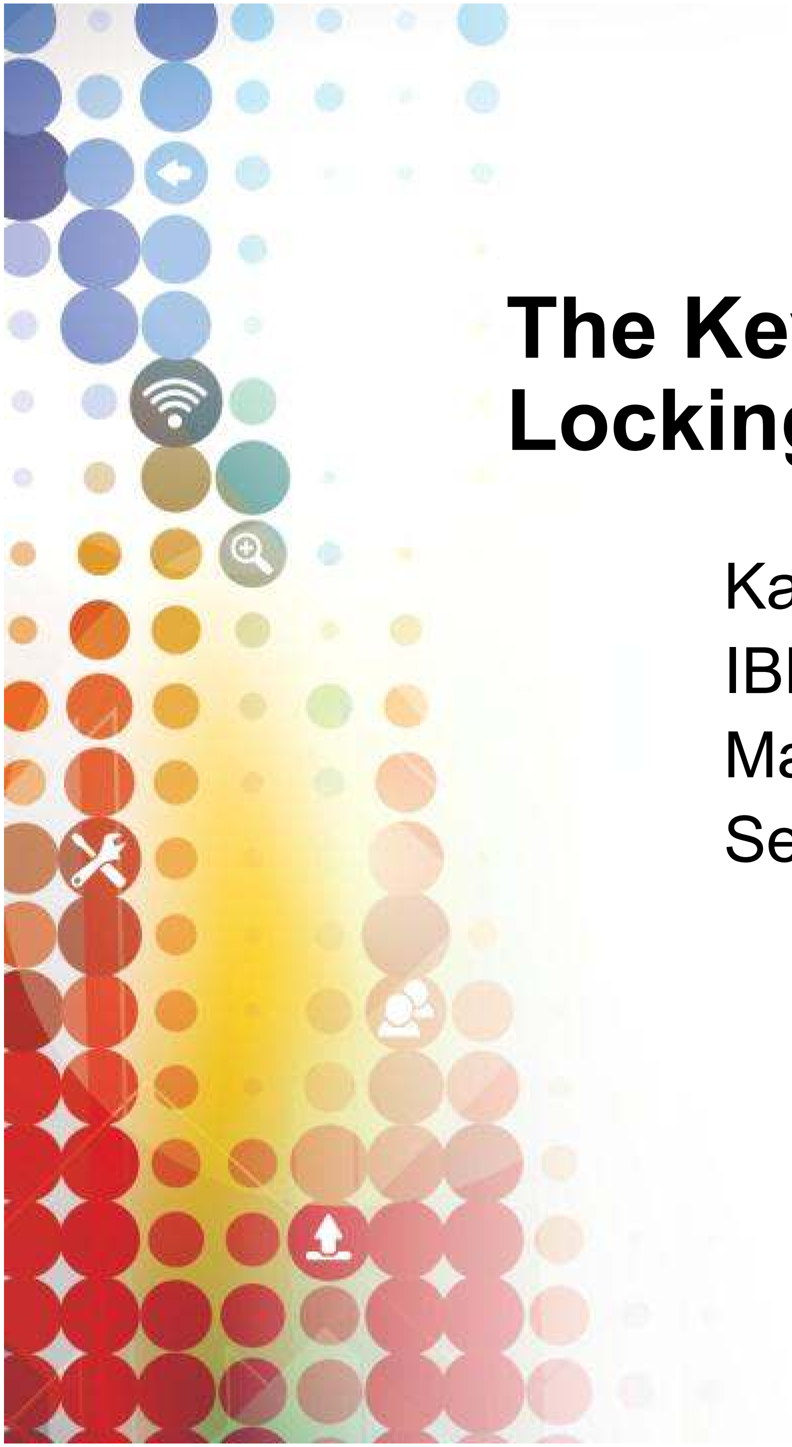


The Keys to Understanding Locking for DB2 for z/OS

Karelle Cornwell
IBM

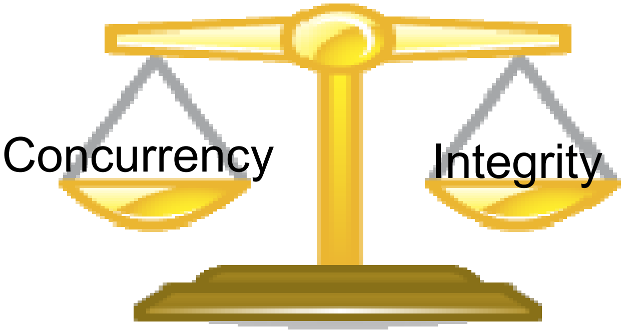
March 10, 2014
Session 14627

Agenda

- The Basics
- How you can influence DB2 locking
- Monitoring locking
- What's new in locking in V10

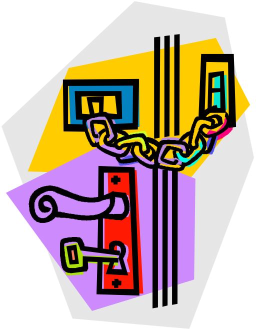
The Basics

- DB2 gets locks to preserve data integrity
- Sometimes locks can cause suspensions, time-outs and deadlocks
- Goal: allow maximum concurrency without jeopardizing data integrity



Lock State and Compatibility

- Lock state – the strength of a lock



Lock State	Owner can read data	Owner can update data	Others can read data	Others can update data
IS – Intent Share	✓		✓	✓
IX – Intent Exclusive	✓	✓	✓	✓
S – Share	✓		✓	
U – Update	✓	✓	✓	
SIX – Share / Intent Exclusive	✓	✓	✓	
X - Exclusive	✓	✓		

Lock compatibility – which locks can be held concurrently by different transactions.

	IS	IX	S	U	SIX	X
IS	✓	✓	✓	✓	✓	
IX	✓	✓				
S	✓		✓	✓		
U	✓		✓			
SIX	✓					
X						

What does DB2 lock?

Table spaces
IS, IX, S, U,SIX, X

Tables
IS, IX, S, U,SIX, X

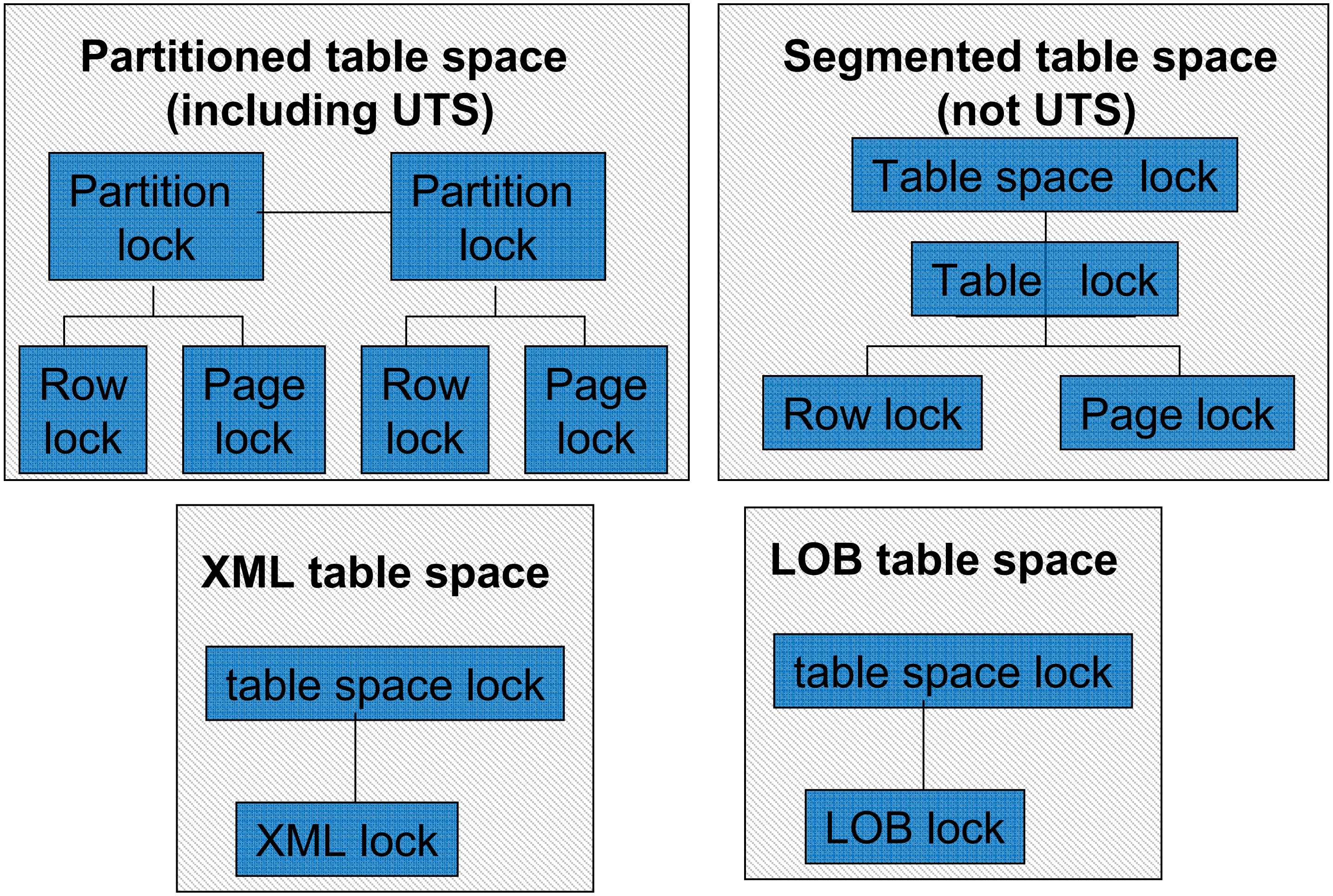
Partitions
IS, IX, S, U,SIX, X

Pages
S, U or X

Rows
S, U or X

LOBs
S or X

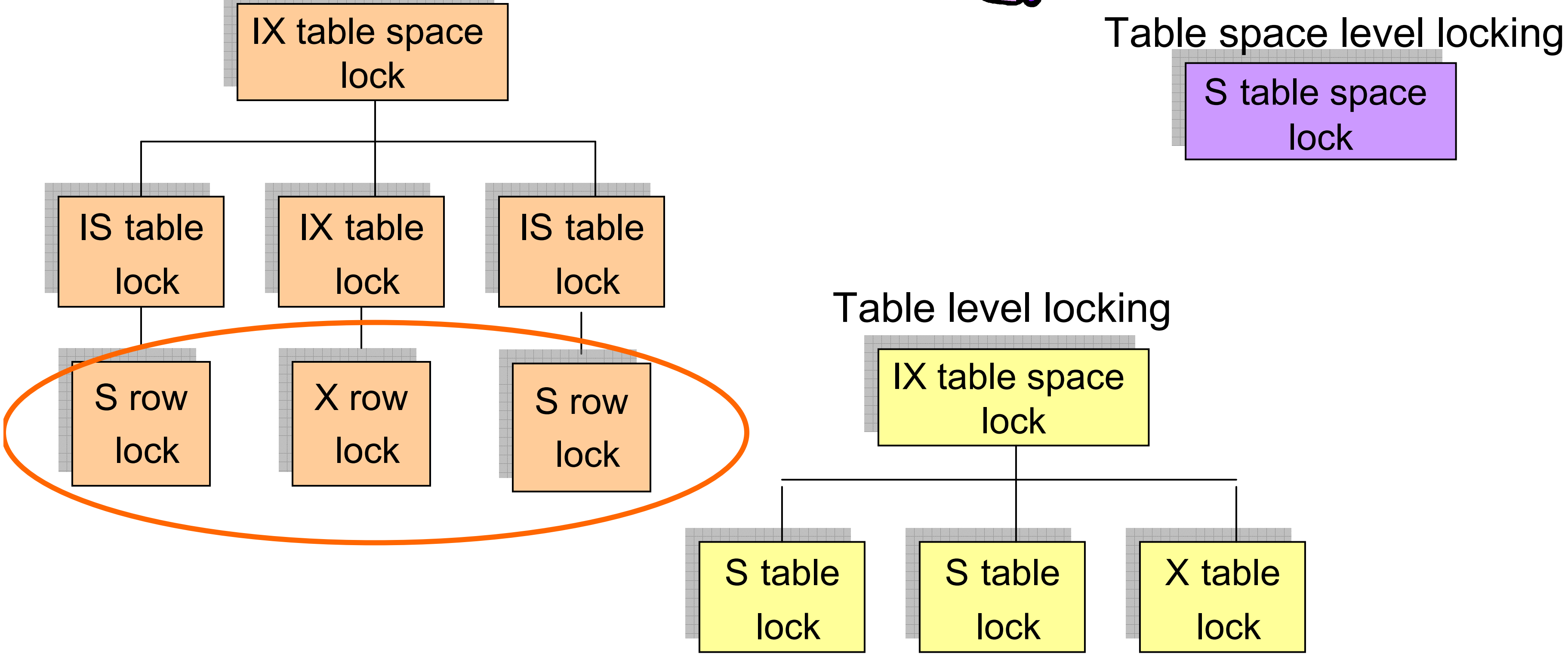
XMLs
S or X



Lock Scope

Segmented, not partitioned

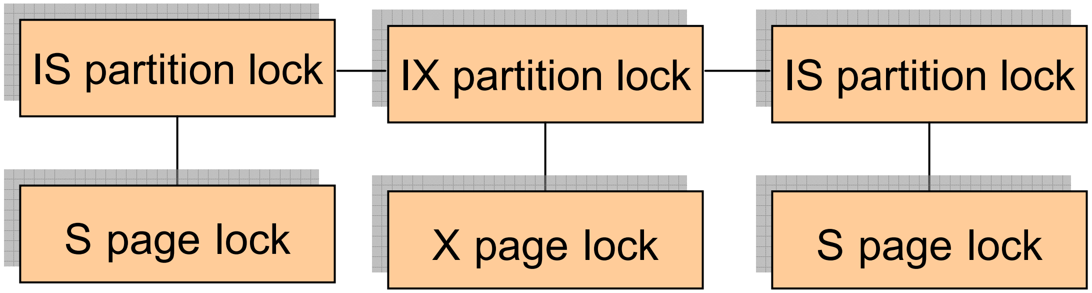
Row level locking



Lock Scope

Partitioned, including UTS

Page level locking



IS partition lock

IX partition lock

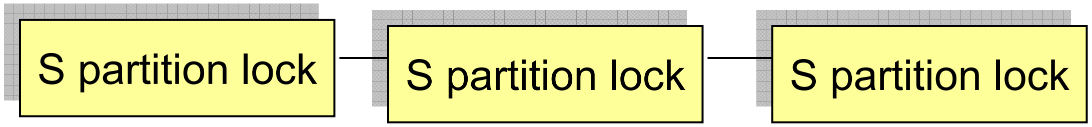
IS partition lock

S page lock

X page lock

S page lock

Table space level locking



S partition lock

S partition lock

S partition lock

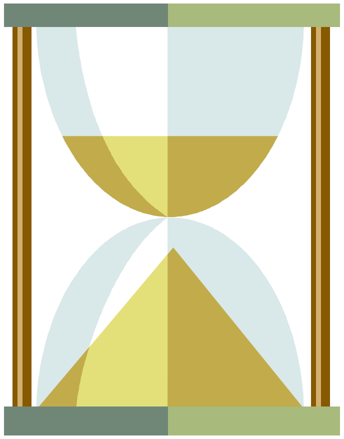
How long is a lock held ?

Page and row locks

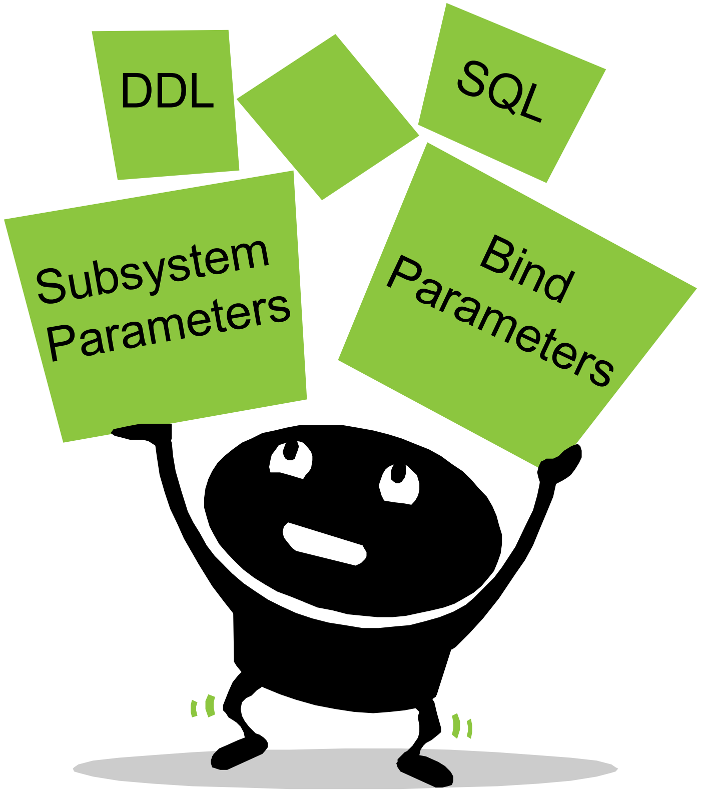
- Acquired when needed
- Fetch: released at next fetch or commit
- Update: held until commit

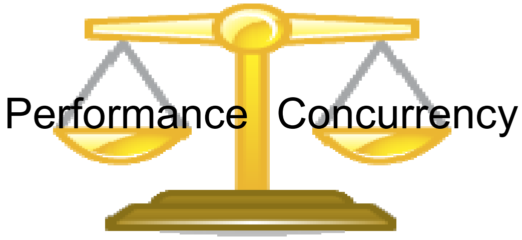
Table space, table & partition locks

- Acquired on first access
- Released at commit or when the application terminates.



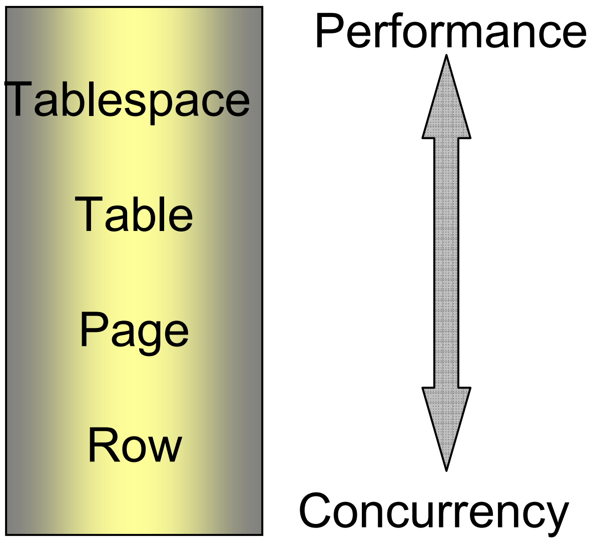
What affects lock states, duration and size?

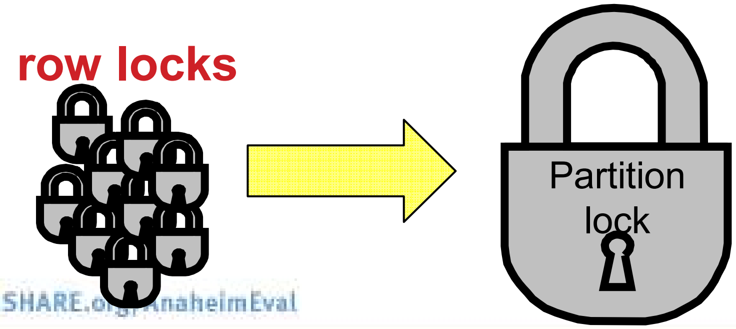




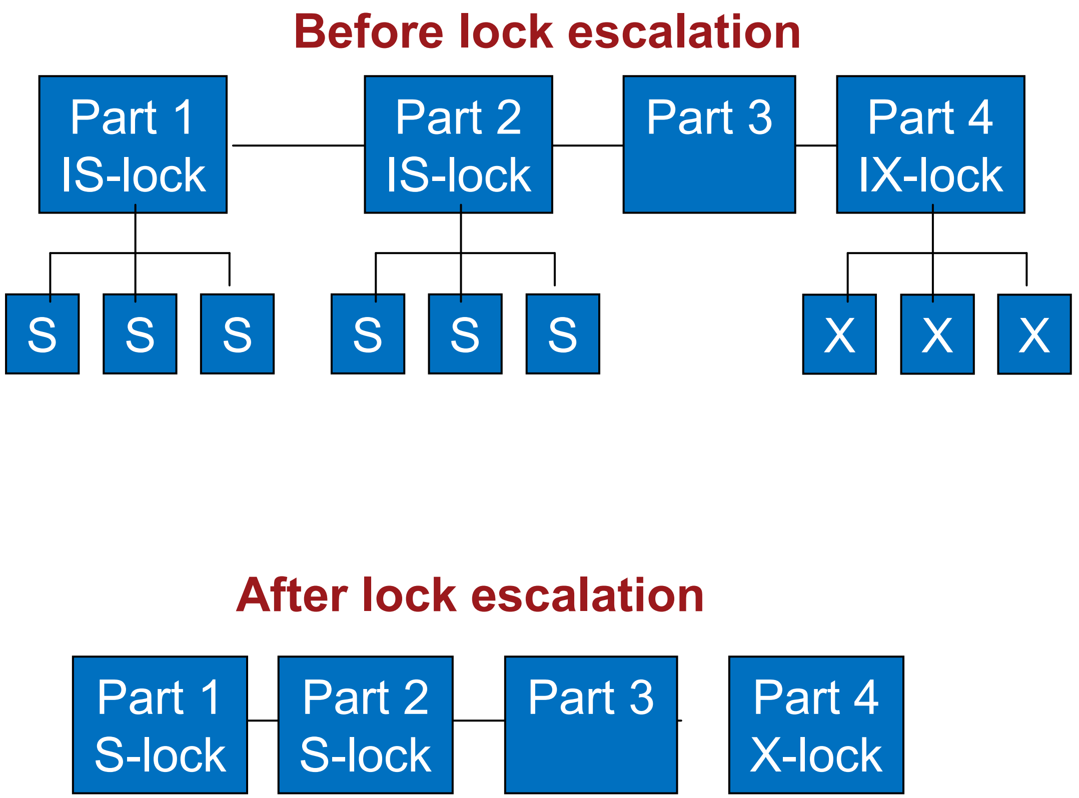
DDL options that affect locks

- CREATE/ALTER TABLESPACE LOCKSIZE
  - Allows you to choose a locksize: tablespace, table, page, row, LOB or XML
  - A smaller lock size generally provides better concurrency
  - A larger lock size generally provides better performance
- CREATE/ALTER TABLESPACE LOCKMAX
  - Allows you to choose the number of low-level locks (page, row, LOB or XML) per table space or table
  - Can be used to enable or disable lock escalation





Lock Escalation



Before lock escalation

Part 1 IS-lock

Part 2 IS-lock

Part 3

Part 4 IX-lock

S S S S S S X X X

After lock escalation

Part 1 S-lock

Part 2 S-lock

Part 3

Part 4 X-lock

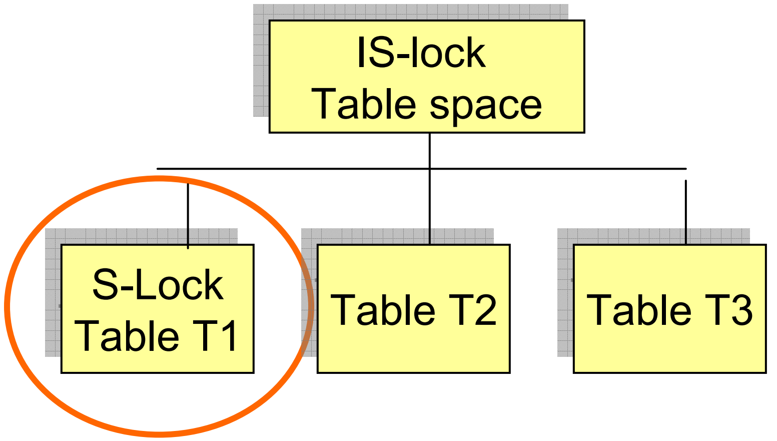
- Occurs when the number of lower level locks (page, row, LOB or XML) reaches the number specified in LOCKMAX
- DB2 acquires a gross lock on the table space, table or partition and releases the lower level locks
- IS escalates to S
- IX and SIX escalate to X
- Locks on all partitions that are locked escalate to a gross lock.
- Improves performance
- Can impact concurrency

SQL – LOCK TABLE statement

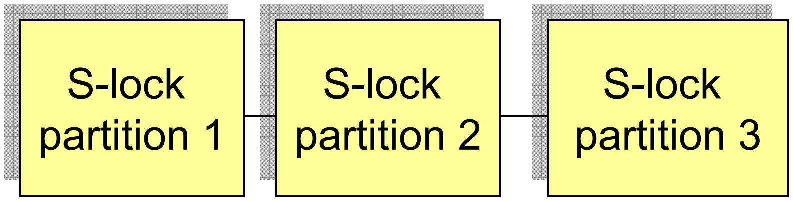
LOCK TABLE T1 IN SHARE MODE → Lock out updaters

LOCK TABLE T1 IN EXCLUSIVE MODE → Lock out readers (with some exceptions) and updaters

LOCK TABLE T1 PARTITION(5) IN SHARE MODE → Locks a single partition



For classic segmented TS, locks only the specified table



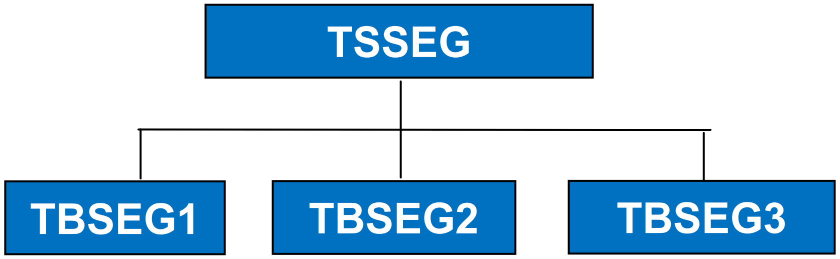
For partitioned TS, locks all Partitions unless PARTITION keyword used

Bind Option - Release

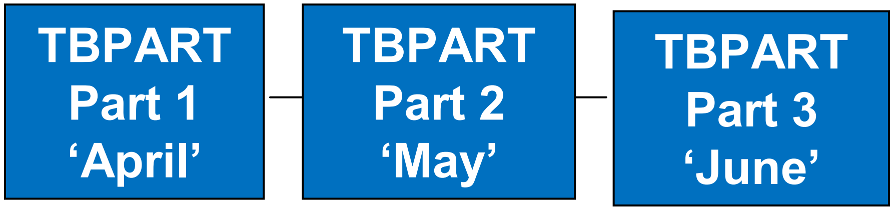
- Release(Commit) – release table space, table, partition locks at commit
  - Exception: Locks held across commit for cursor with hold
- Release(Deallocate) – release table space, table, partition locks when the plan completes.
  - Has no effect on dynamic SQL unless
    - Using KEEPDYNAMIC(YES), and subsystem parameter CACHEDYN=YES

Bind Option: Release

Segmented table space TSSEG
With tables TBSEG1, TBSEG2
and TBSEG3

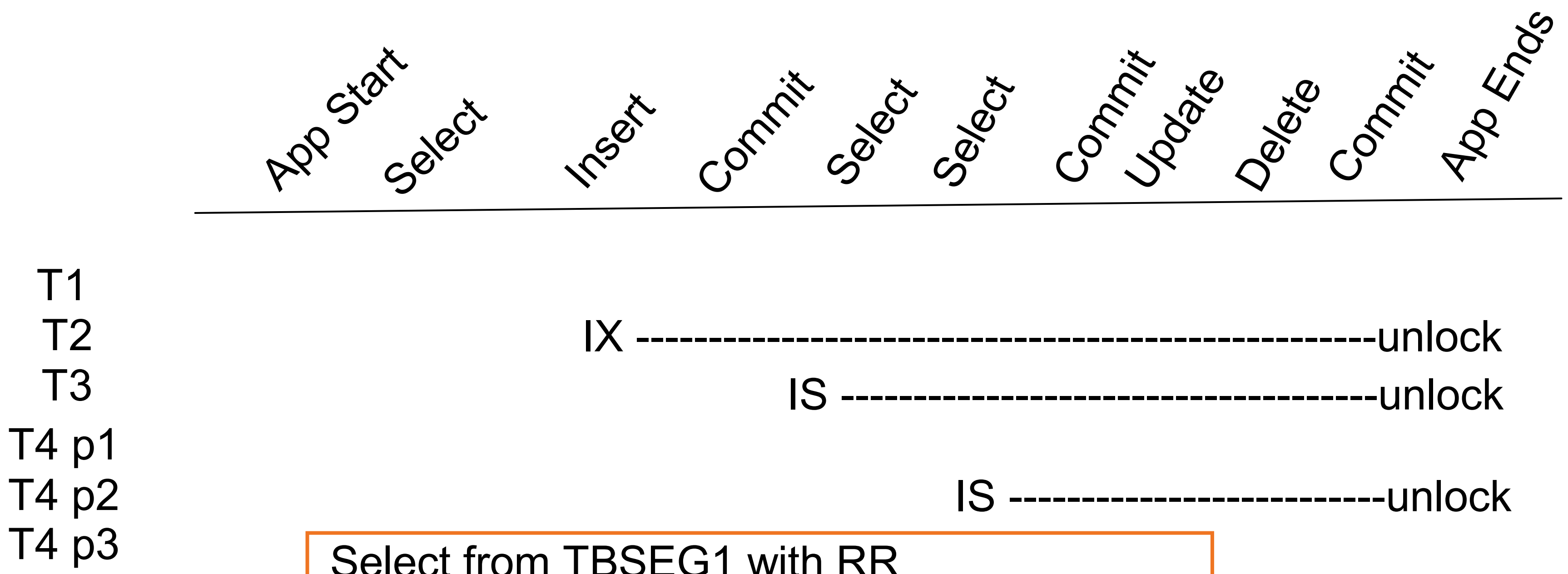


Partitioned table space TSPART
With table TBPART



Select from TBSEG1 with RR
Insert into TBSEG2
Commit
Select from TBSEG3
Select from TBPART where month = 'May'
Commit
Update T3
Delete from T2
Commit

Acquire(Use) Release(Deallocate)


App Start
Select
Insert
Commit
Select
Select
Commit
Update
Delete
Commit
App Ends
T1
T2
T3
T4 p1
T4 p2
T4 p3
IX ---unlock
IS -------------------------------------unlock
IS --------------------------unlock

Select from TBSEG1 with RR

Insert into TBSEG2

Commit

Select from TBSEG3

Select from TBPART where month = ‘May’

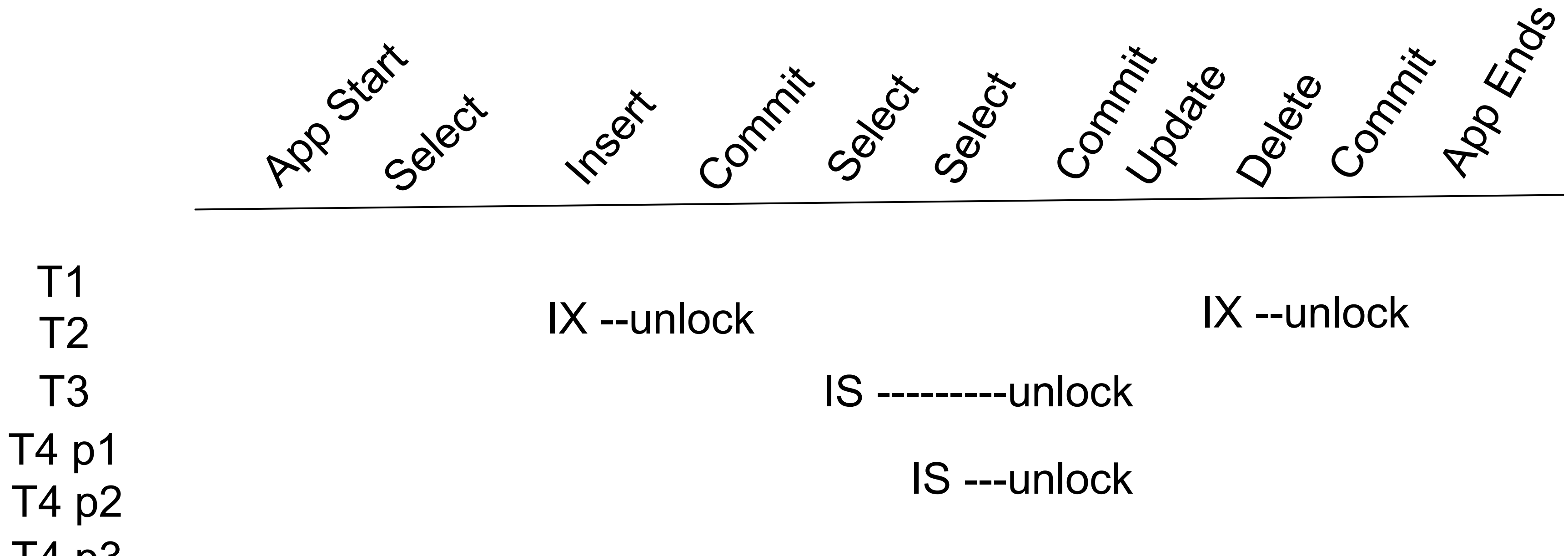
Commit

Update TBSEG3

Delete from TBSEG2

Commit

Acquire(Use) Release(Commit)



Select from TBSEG1 with RR

Insert into TBSEG2

Commit

Select from TBSEG3

Select from TBPART where month = ‘May’

Commit

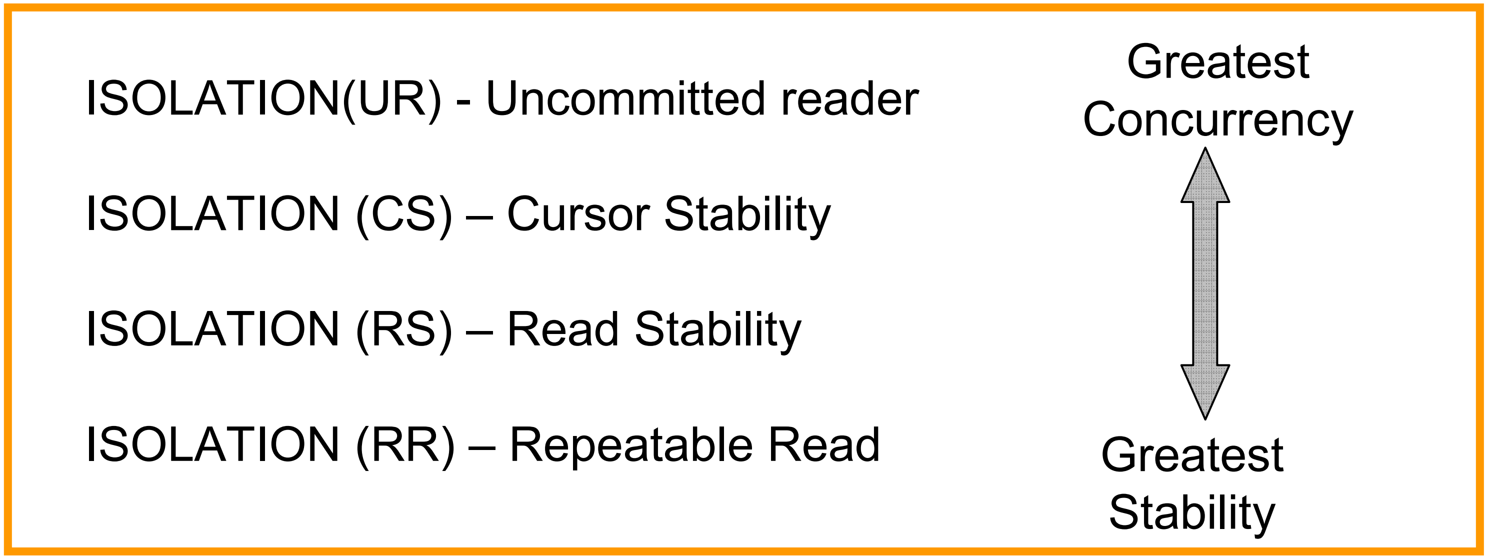
Update TBSEG3

Delete from TBSEG2

Commit

Isolation

Isolation is the degree to which one transaction is isolated from other transactions



ISOLATION(UR) - Uncommitted reader

ISOLATION (CS) – Cursor Stability

ISOLATION (RS) – Read Stability

ISOLATION (RR) – Repeatable Read

Greatest Concurrency

Greatest Stability

Can be specified

- as bind option for a plan or package
- on an SQL statement

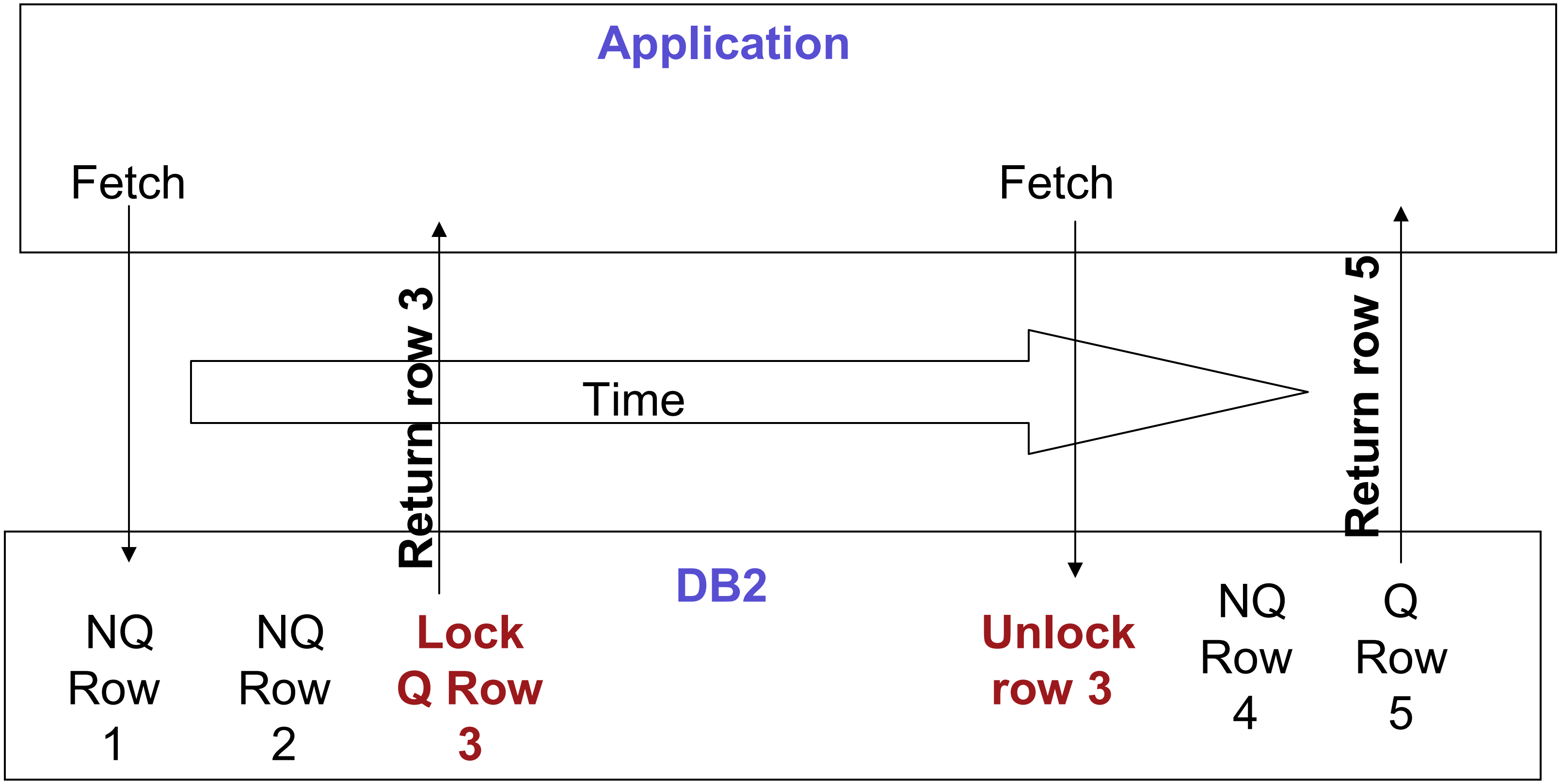
SELECT AVG(SALARY) FROM EMPLOYEE_TABLE WITH UR

Isolation UR

- OK to read data that is not committed
- Does not acquire table space, table, partition, row or page locks.   Does need XML locks.
- Only use if application can tolerate uncommitted data

Isolation CS

The previous row is unlocked when the next row is fetched

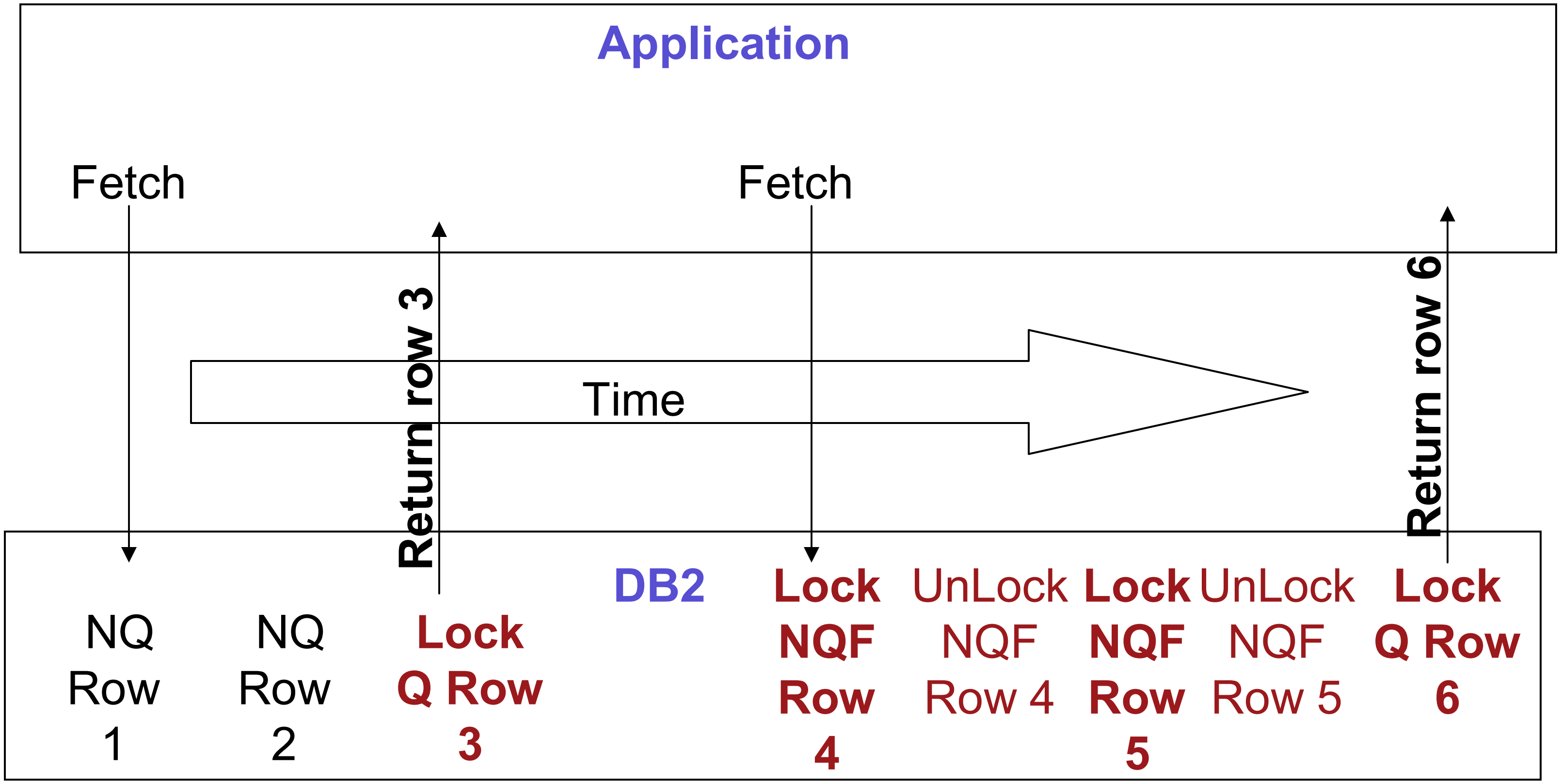


Q = stage 1 qualifying row
NQ – non-qualifying row

With Currentdata(no) may be able to avoid locking

Isolation RS

Locks are held until commit on all qualifying rows



Application

Fetch

Fetch

Return row 3

Time

Return row 6

DB2

NQ Row 1

NQ Row 2

Lock Q Row 3

Lock NQF Row 4

UnLock NQF Row 4

Lock NQF Row 5

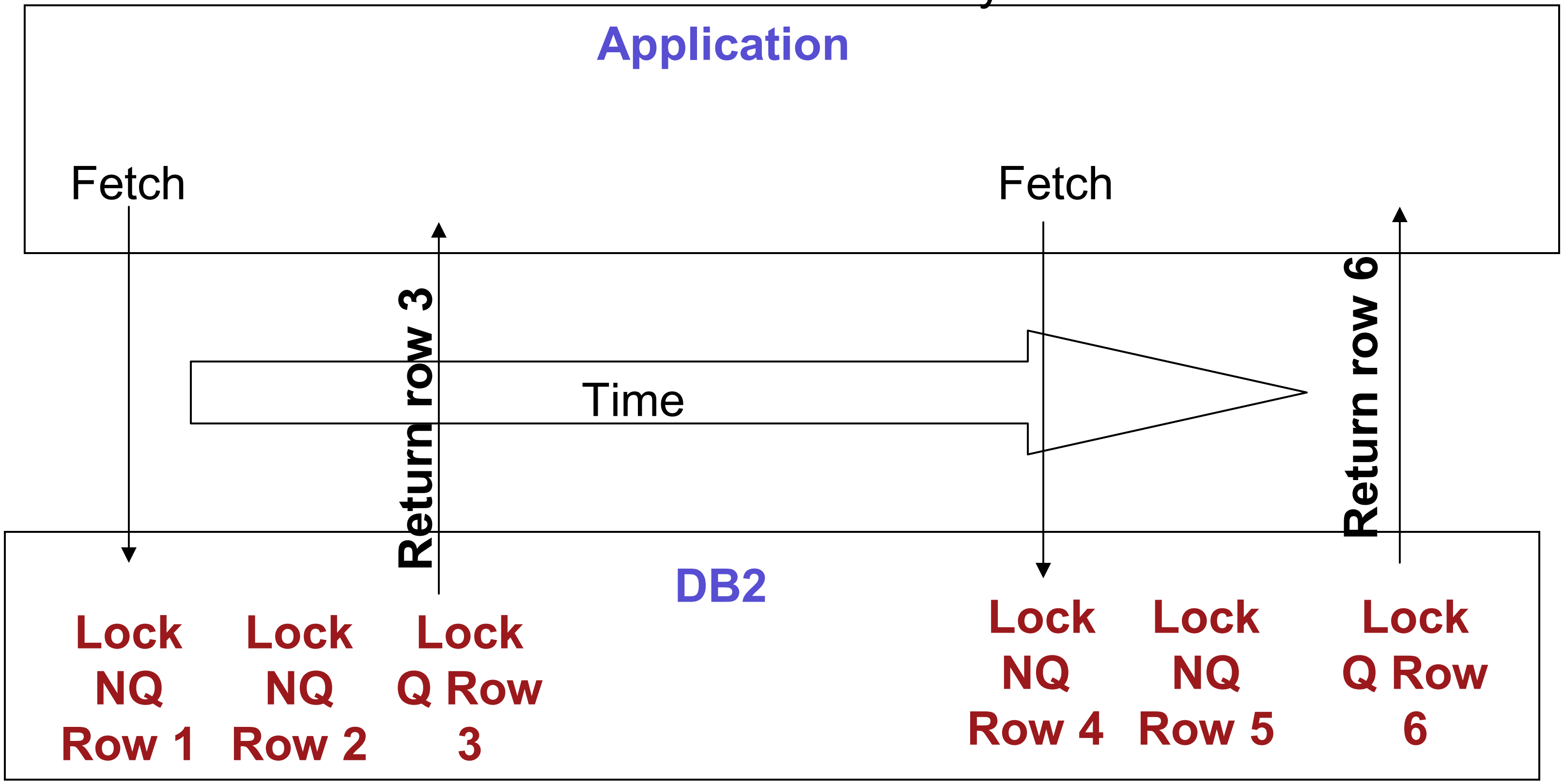
UnLock NQF Row 5

Lock Q Row 6

Q = stage 1 qualifying row
NQ – non-qualifying row
NQF – non-qualifying row & lock avoidance fails

Isolation RR

Locks are held until commit on every row that is read



Q = stage 1 qualifying row
NQ – non-qualifying row

Use and Keep Locks

- Specifies the lock state for the page or row lock
- Can specify SHARE, UPDATE or EXCLUSIVE
- Can be specified on the ISOLATION clause of a SELECT statement
- Only valid with RS and RR

SELECT * FROM T1 WITH RS
USE AND KEEP UPDATE LOCKS

Subsystem Parameters

- **NUMLKUS**: Locks per user
  - Specifies the maximum number of row, page, LOB and XML locks a single application can hold concurrently for all table spaces.
- **NUMLKTS**: Locks per table space
  - Specifies the maximum number of row, page, LOB and XML locks an application can hold at one time on a table or table space.
- **RRULOCK**: U-lock for RR/RS
  - YES indicates the U locks are used instead of S locks when using cursor to fetch rows for update
- **XLKUPDLT**: x-lock for searched updates/deletes
  - DB2 uses an X-lock on qualifying rows or pages during a searched update or delete
- **EVALUNC**: evaluate uncommitted
  - Indicates whether predicate evaluation is to be allowed on uncommitted data of other transactions
  - For isolation(cs) and isolation(rs) only
- **SKIPUNCI**: skip uncommitted inserts
  - whether statements are to ignore a row that was inserted by another transaction if the row has not been committed
  - For isolation(cs) or isolation(rs) and row level locking

Lock states used with isolations RS and RR

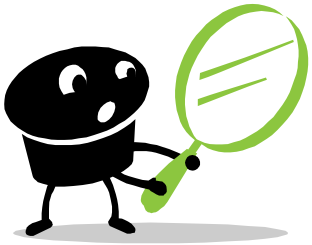
	Cursor SELECT with FOR UPDATE	Non-cursor SELECT	DELETE and UPDATE
RRULOCK	U		U
USE AND KEEP LOCKS	S,U,X	S,U,X	
XLKUPDLT			X

For USE AND KEEP LOCKS:

- S – SHARE
- U – UPDATE
- X - EXCLUSIVE

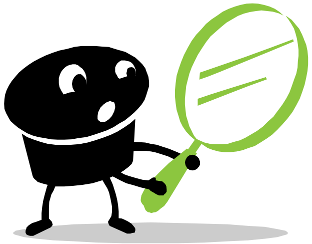
XLKUPDLT takes precedence over RRULOCK
USE AND KEEP takes precedence over RRULOCK

Monitoring Locking



- **Catalog**
  - LOCKRULE column of SYSTABLESPACE gives the lock size for the table space
  - LOCKMAX column of SYSTABLESPACE gives the maximum number of locks per user for the table or table space
  - ISOLATION column of SYSPLAN and SYSPACKAGE
  - RELEASE column of SYSPLAN and SYSPACKAGE
  - CURRENTLYCOMMITTED column of SYSPLAN and SYSPACKAGE
- **Display Database command**
  - With the LOCKS option, display which table spaces are locked and in what lock state and duration
- **Explain output**
  - TSLOCKMODE in PLAN_TABLE gives the table space lock to be used by the SQL statement

Monitoring Locking



- Traces
  - IFCID 20 – lock summary
  - IFCID 21 – lock detail
  - IFCID 172 – deadlock trace
  - IFCID 196 – timeout trace

What's new in locking in DB2 V10?

Remove ACQUIRE(ALLOCATE)

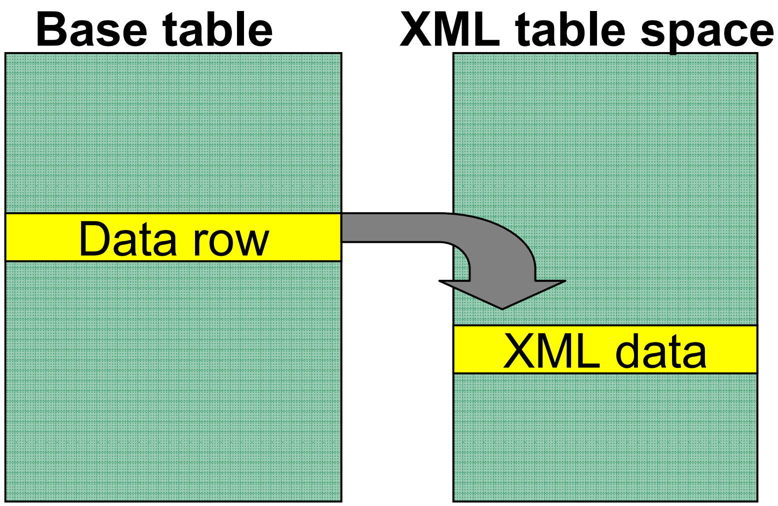
XML

Changes to zParms

Currently committed

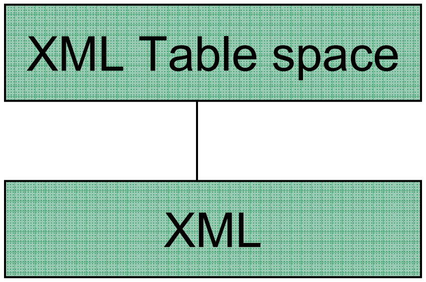
Row locks in the catalog

XML locking



- V9 – XML data type introduced
- V10 – XML versioning
- XML data is stored in a separate table space
- XML table space is locked separately from the base table space

Locking hierarchy



XML locking

	V9	V10 (XML versioning)
Insert/Update/Delete	X-lock XML Hold until commit	X-lock XML Hold until commit
Select	S-lock XML Release lock on next fetch	No XML lock for ISO(CS), ISO(RS), ISO(RR). S-lock for ISO(UR), if needed.

ACQUIRE(ALLOCATE) (V10)

- Deprecated in V10
- Goes hand-in-hand with the disallowing DBRMs bound into plans
- All plans and packages will be treated as ACQUIRE(USE)

BIND PLAN(PL147) PKLIST(PK01.D119746) ACQUIRE(ALLOCATE)

Subsystem Parameters

- **NUMLKTS**: Locks per table space
  - Default changes from 1000 to 2000 (V10)
- **RRULOCK**: U-lock for RR/RS
  - Default changes from NO to YES (V10)

Currently Committed (V10)

- Allows a query transaction to access the currently committed image of data if this query hits a row locked by any INSERT or DELETE
- Helps to avoid time-outs and waits for locks
- Universal Table space (UTS) only
- Isolation(CS) or isolation(RS)
- Page level or row level locking
- Cannot be used if updater holds a gross lock on the partition

Currently Committed and Uncommitted Insert

```
CREATE T1 (COL1 CHAR(1),
COL2 INT,
COL3 CHAR(1));
Transaction A:
INSERT INTO T1 VALUES ('D', 2, 'Y');  not committed
Transaction B:
SELECT * FROM T1 WHERE COL1 = 'D';
```

Transaction B finds that the row where COL1 = ‘D’ is locked. With Currently Committed, it skips the row (just like zParm SKIPUNCI).
No rows returned.
No waiting for a lock.

Currently Committed and Uncommitted Delete

CREATE T1 (COL1 CHAR(1),
COL2 INT,
COL3 CHAR(1));
Transaction A:
DELETE FROM T1 WHERE COL1=‘D’; **not committed**
Transaction B:
SELECT * FROM T1 WHERE COL1 = 'D';

Transaction B finds that the row where COL1 = ‘D’ is locked. With Currently Committed, it determines that the delete is not committed. Returns the row.

No waiting for a lock.

Reader must be ISO(CS) Currentdata(No)

Where to specify Currently Committed

- As an attribute of a PREPARE statement
  - USE CURRENTLY COMMITTED
  - WAIT FOR OUTCOME
- As a option on BIND & REBIND PLAN, BIND & REBIND PACKAGE, REBIND TRIGGER PACKAGE
  - CONCURRENTACCESSRESOLUTION
    - USECURRENTLYCOMMITTED
    - WAITFOROUTCOME
- As an option on CREATE & ALTER PROCEDURE, CREATE & ALTER FUNCTION
  - CONCURRENT ACCESS RESOLUTION
  - USE CURRENTLY COMMITTED
  - WAIT FOR OUTCOME

Currently Committed and Skip Uncommitted Insert

SKIPUNCI	CONCURRENTACCESSRESOLUTION	ACTION
YES	USECURRENTLYCOMMITTED	Skip uncommitted inserts
YES	WAITFOROUTCOME	Wait for COMMIT or ROLLBACK
YES	Not specified	Skip uncommitted inserts
NO	USECURRENTLYCOMMITTED	Skip uncommitted inserts
NO	WAITFOROUTCOME	Wait for COMMIT or ROLLBACK
NO	Not specified	Wait for COMMIT or ROLLBACK

Conclusion

- You need not do anything. DB2 will lock for you.
- If you have concurrency issues such as time-outs, deadlocks, lots of suspensions, DB2 provides various tuning options.