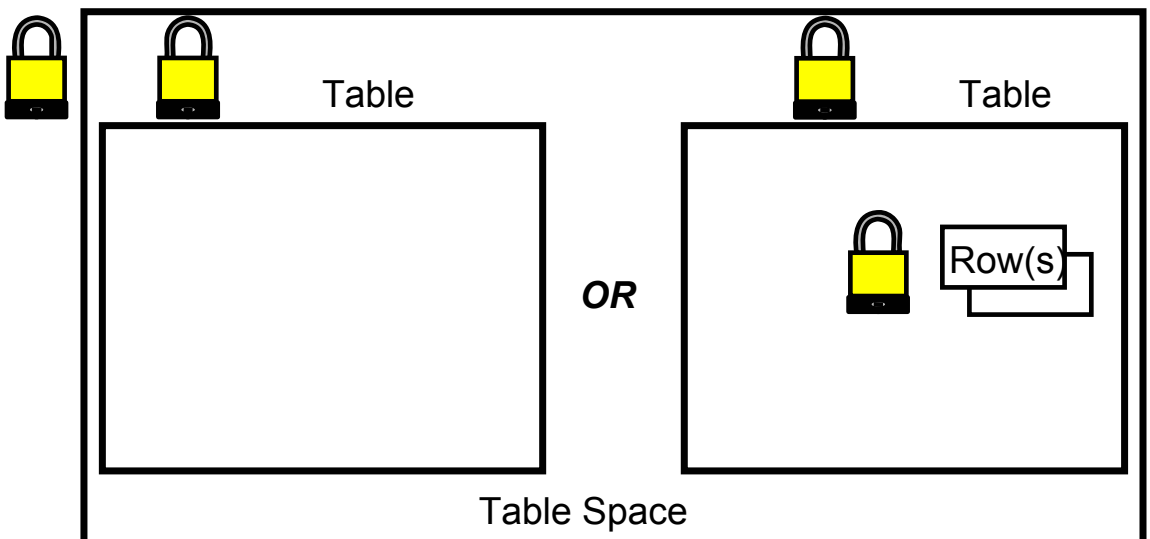
IBM Information Management software

Locks and concurrency

Why are locks needed?

- Ensure data integrity while permitting multiple applications to access data
- Prohibit applications from accessing uncommitted data written by other applications (*unless* Uncommitted Read isolation level is used)
- Control undesirable effects
  - Lost data due to concurrent updates
  - Unrepeatable reads
  - Phantom read phenomenon

Locking strategies



DB2 employs *either* strict table locking ***OR***
table locking in conjunction with row locking
for typical application processing

Table lock modes

IN	Intent None
IS	Intent Share
IX	Intent eXclusive
SIX	Share with Intent eXclusive
S	Share
U	Update
X	eXclusive
Z	superexclusive

Row Locking also used

Strict Table Locking

(See next page)

Row lock modes

Row Lock		Minimum* Supporting Table Lock
S	Share	IS
U	Update	IX
X	eXclusive	IX
W	Weak exclusive	IX
NS	Next key Share	IS
NW	Next key Weak exclusive	IX

An application does not acquire **Row locks**

if it is using Table Locks of **S, U, X, or Z**

Lock mode compatibility

MODE OF LOCK A	MODE OF LOCK B							
	IN	IS	S	IX	SIX	U	X	Z
IN	YES	YES	YES	YES	YES	YES	YES	NO
IS	YES	YES	YES	YES	YES	YES	NO	NO
S	YES	YES	YES	NO	NO	YES	NO	NO
IX	YES	YES	NO	YES	NO	NO	NO	NO
SIX	YES	YES	NO	NO	NO	NO	NO	NO
U	YES	YES	YES	NO	NO	NO	NO	NO
X	YES	NO	NO	NO	NO	NO	NO	NO
Z	NO	NO	NO	NO	NO	NO	NO	NO

Table Locks

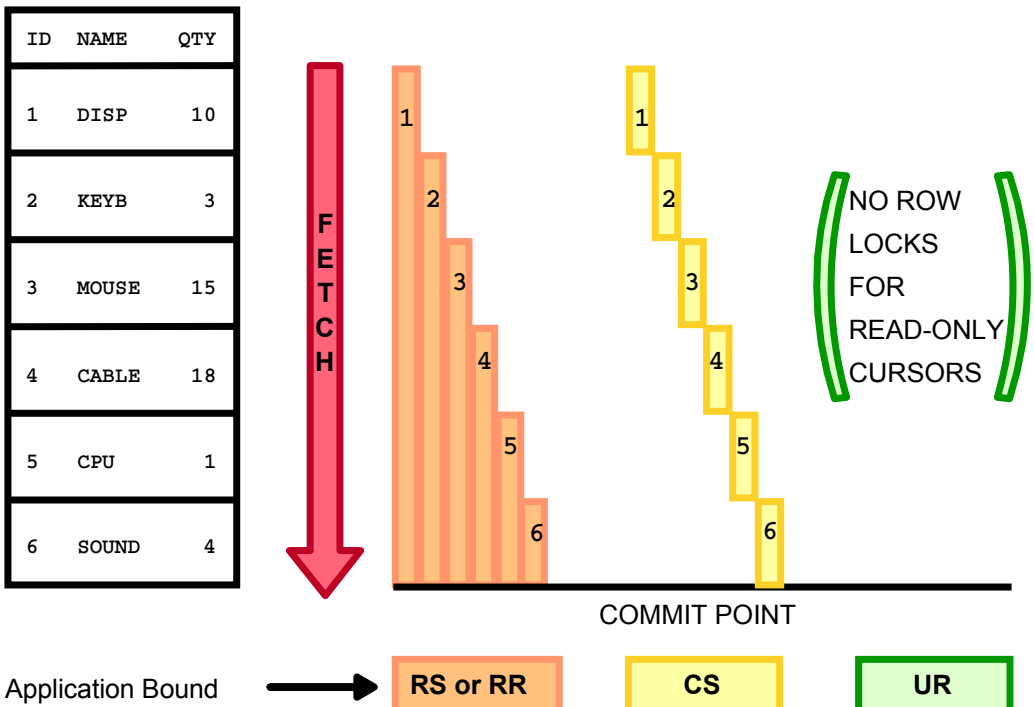
Row Locks

LOCK A MODE	MODE OF LOCK B					
	S	U	X	W	NS	NW
S	YES	YES	NO	NO	YES	NO
U	YES	NO	NO	NO	YES	NO
X	NO	NO	NO	NO	NO	NO
W	NO	NO	NO	NO	NO	YES
NS	YES	YES	NO	NO	YES	YES
NW	NO	NO	NO	YES	YES	NO

Introduction to isolation levels

- DB2 provides different levels of protection to isolate data:
  - Uncommitted read (UR)
  - Cursor stability (CS) - default
  - Read stability (RS)
  - Repeatable read (RR)
- Cursor stability is the default isolation level
- Isolation level can be specified for a session, a client connection, or an application before a database connection
  - For embedded SQL, the level is set at bind time
  - For dynamic SQL, the level is set at run time

Isolation levels



DB2 isolation levels

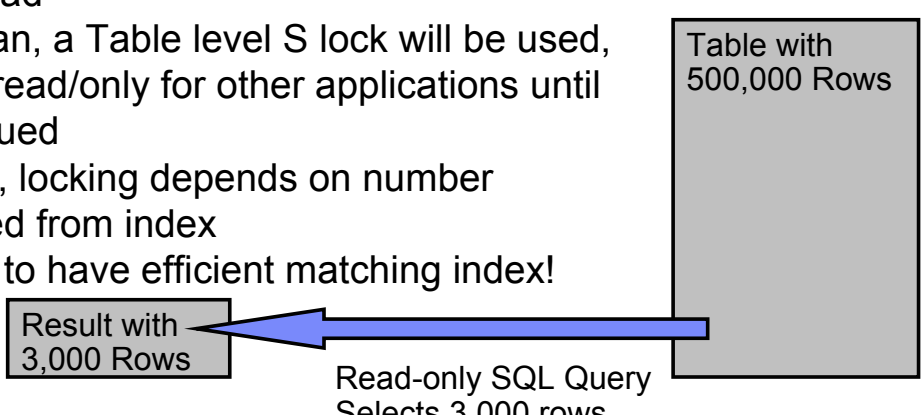
1. UR - Uncommitted Read
   - For read only queries, no record locking
   - Will see uncommitted changes by other transactions
   - Good for accessing read only tables
   - Statements in UR which modify data are upgraded internally to CS
2. CS - Cursor Stability
   - Default isolation level
   - Locks and unlocks each row, 1 at a time (never has 2 locks at once)
   - Guaranteed to only return data which was committed at the time of the read
3. RS - Read Stability
   - Will keep all qualifying rows locked until the transaction is completed
   - Does release locks on rows that do not satisfy query predicates
   - Use for result set stability or when future actions on returned rows may be taken
4. RR - Repeatable Read
   - Will lock all rows visited and keep locks until the transaction is completed
   - Use only when consistent results are required

DB2 and ANSI isolation levels anomalies allowed and disallowed

DB2 Isolation	ANSI Isolation	Dirty Write	Dirty Read	Fuzzy Read	Phantom Read
Uncommitted Read (UR)	Read Uncommitted (Level 0)	✗	✓	✓	✓
Cursor Stability (CS)	Read Committed (Level 1)	✗	✗	✓	✓
Read Stability (RS)	**Repeatable Read (Level 2)**	✗	✗	✗	✓
Repeatable Read (RR)	Serializable (Level 3)	✗	✗	✗	✗

Isolation levels and Read locks for Query results

- Uncommitted Read – 1 lock needed IN Table lock, No Row locks
- Cursor Stability – 2 Locks needed
  IS Table lock, 1 NS row lock
- Read Stability – 3001 locks needed
  IS Table lock, 3000 NS Row Locks
- Repeatable Read
  For a Table scan, a Table level S lock will be used, so the table is read/only for other applications until a commit is issued
  For Index scan, locking depends on number of rows matched from index
  Very important to have efficient matching index!



SET CURRENT ISOLATION

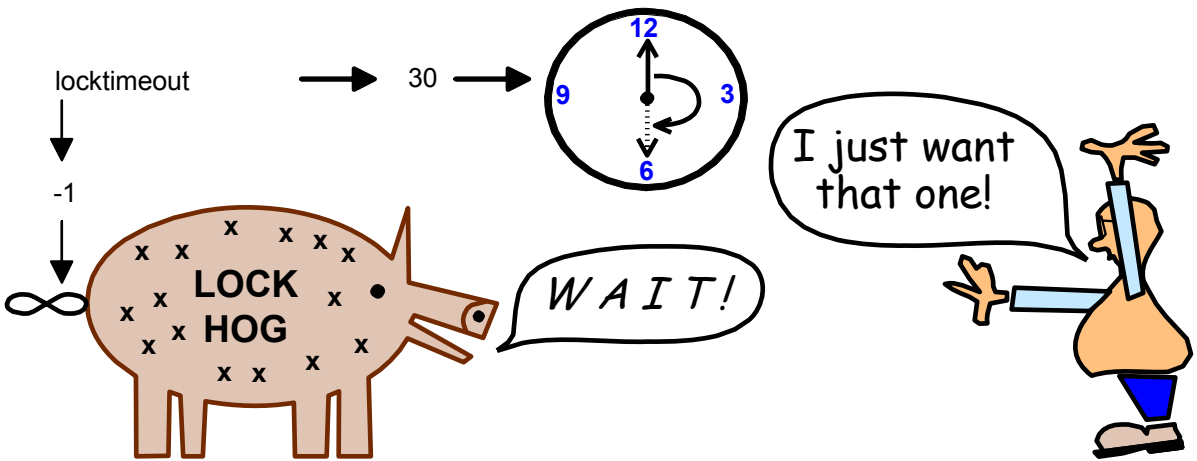
- Assigns a value to the CURRENT ISOLATION special register

```
SET [CURRENT] ISOLATION [=] {UR | CS | RR | RS | RESET}
```

- CURRENT ISOLATION can be set anytime while connected to a database (special registers are attributes of the connection session)
- Applies to the current session only
- The register setting is used immediately for subsequent SQL statements *until value is set again*, thus allowing different isolation to be applied to different SELECT (and other) statements even within one UOW/transaction
- You can read the current register setting with:

```
VALUES CURRENT ISOLATION
```

Lock wait and timeout

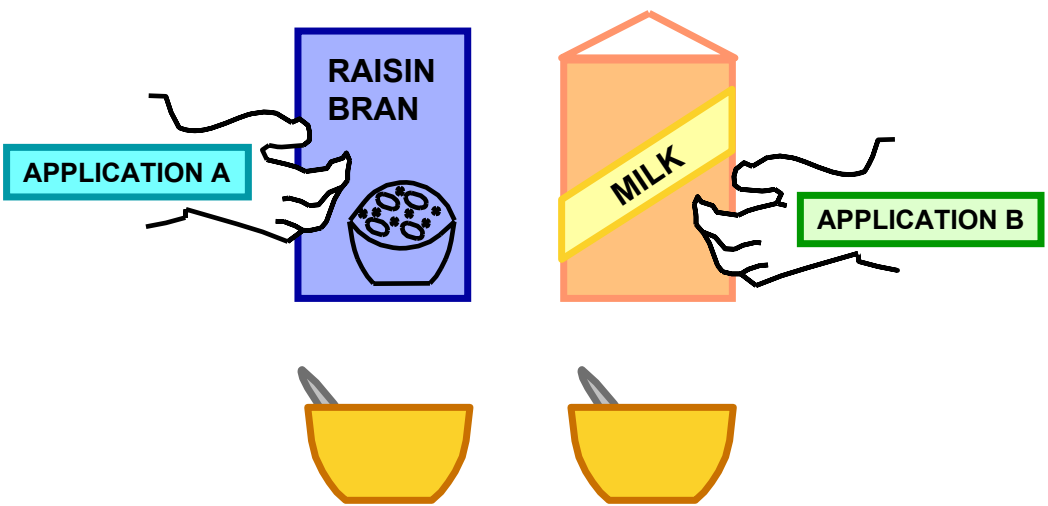


If the application *hogging* the locks doesn't COMMIT or ROLLBACK other applications wait until lock is available or timeout exceeded

Deadlock causes and detection

UNIT OF WORK —

DELETE SOME CEREAL AND MILK



dlchktime 10000 milliseconds

Deadlocks

- Deadlock detector wakes up every DLCHKTIME
  - Each database partition has its own deadlock detector (*db2dlock* process)
  - Catalog database partition (DPF) has the global deadlock detector (*db2glock*)
- A deadlock occurs when there is a cycle between 2 or more transactions which are in lock wait on one another - and the deadlock detector is able to wake up and recognize this before any lock timeout occurs
- DB2 itself will determine which transaction among those involved in the deadlock cycle will be chosen to be the deadlock victim and rolled back
  - Some utilities are executed with an internal flag, to *not* be chosen as the victim
  - Preferential treatment is given to transactions holding Z locks
  - There is currently no way to *help* DB2 choose or prefer/save a victim
- The victim will rollback and return SQL error SQL0911 or SQL0913
- A Deadlock Event monitor can be used to capture information when deadlocks are detected.
- The Health Monitor tracks deadlocks per hour and can generate an alert if the deadlock rate is unusually high.

Snapshot Monitor — taking snapshots