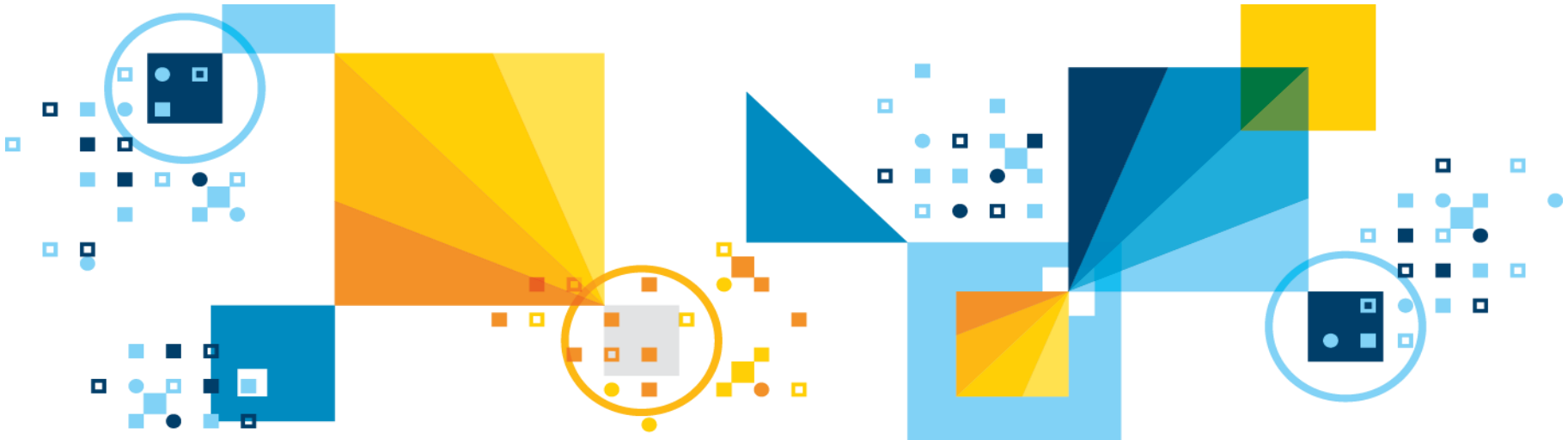


# DB2 Data Concurrency

**Module ID** | 10106

**Length** | 45 minutes



For questions about this presentation contact [askdata@ca.ibm.com](mailto:askdata@ca.ibm.com)

January 29, 2015

## Disclaimer

**© Copyright IBM Corporation 2015. All rights reserved.**

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

Other company, product, or service names may be trademarks or service marks of others.

## Module Information

- You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:
  - DB2 Fundamentals
  
- After completing this module, you should be able to:
  - Describe DB2 Concurrency Control
  - Explain the concepts of:
    - Concurrency
    - Isolation Levels

## Module Content

- Concurrency
- Concurrency Issues
- Concurrency Control
  - Isolation Levels
  - Locking
  - Specifying Isolation Levels



# Concurrency

- Concurrency is the sharing of resources by multiple interactive users or application programs at the same time
  - Provides increased application throughput
  - Increased responsiveness across the system
  - Better resource utilization within the system
- Need to be able to control the degree of concurrency:
  - With proper amount of data stability
  - Without loss of performance
- Having multiple interactive users can lead to:
  - Lost Update
  - Uncommitted Read
  - Non-repeatable Read
  - Phantom Read

# Terminology in Concurrent Applications

- Transaction
  - Sequence of one or more SQL operations, grouped together as a single unit
  - Also known as a unit of work
- Committed Data
  - Using the COMMIT statement commits any changes made during the transaction to the database
- Uncommitted Data
  - Changes during the transaction before the COMMIT statement is executed



## Concurrency Issues

- Lost Update

- Occurs when two transactions read and then attempt to **update** the **same data**, the **second update** will **overwrite** the **first update** before it is committed

- 1) Two applications, A and B, both read the same row and calculate new values for one of the columns based on the data that these applications read
- 2) A updates the row
- 3) Then B also updates the row
- 4) A's update lost

## Concurrency Issues

- Uncommitted Read
  - Occurs when **uncommitted data** is **read** during a transaction
  - Also known as a Dirty Read

- 1) Application A updates a value
- 2) Application B reads that value before it is committed
- 3) A backs out of that update
- 4) Calculations performed by B are based on the uncommitted data



## Concurrency Issues

- Non-repeatable Read
  - Occurs when a transaction **reads** the **same row of data twice** and returns **different data values** with each read

- 1) Application A reads a row before processing other requests
- 2) Application B modifies or deletes the row and commits the change
- 3) A attempts to read the original row again
- 4) A sees the modified row or discovers that the original row has been deleted

## Concurrency Issues

- Phantom Read
  - Occurs when a **search** based on **some criterion** returns **additional rows** after **consecutive searches** during a transaction

1) Application A executes a query that reads a set of rows based on some search criterion

2) Application B inserts new data that would satisfy application A's query

3) Application A executes its query again, within the same unit of work, and some additional phantom values are returned

# Concurrency Control

- Isolation Levels
  - determine **how data is locked** or isolated from other concurrently executing processes while the data is being accessed
  - are in **effect** while the **transaction is in progress**
- There are four levels of isolation in DB2:
  - **Repeatable read** (RR)
  - **Read stability** (RS)
  - **Cursor Stability** (CS)
    - Currently Committed semantics introduced in DB2 9.7
  - **Uncommitted read** (UR)

## Locking in DB2

- Isolation levels are **enforced by locks**
  - Locks **limit or prevent** data **access** by **concurrent users** or applications
  - Before read/write data, transactions need to acquire the lock on the data
- Locking Attributes
  - objects which can be **explicitly** locked are **databases, tables & tablespaces**
  - objects which can be **implicitly** locked are **rows, index keys, and tables**
  - implicit locks are acquired by DB2 according to isolation level and processing situations
  - **object** being locked represents **granularity** of lock
  - **length of time** a lock is held is **called lock count** and is affected by isolation level
- Database Configuration Parameters
  - **LOCKLIST**: amount of memory allocated to the lock list
  - **MAXLOCKS**: percentage of the lock list held by an application that must be filled before the database manager performs lock escalation
  - Both can be automatically managed by DB2's Self-Tuning Memory Manager.

## Types of Locks

- DB2 for LUW
  - Locks are acquired for all operations to control how other applications access the same resource.
- Factors that affect locking:
  - The type of processing that the application performs
  - The data access method
  - The values of various configuration parameters
- Examples of Types of Locks in DB2
  - **Share (S)**
    - Owner and concurrent transactions are limited to **read-only**
  - **Update (U)**
    - Owner can read/write, but concurrent transactions are limited to **read-only** operations
  - **Exclusive (X)**
    - Owner can read/write. Concurrent transactions **cannot read/write**. UR application can still read the data.



# Deadlock


- Deadlock Detector

- It monitors information about agents that are waiting on locks to discover **deadlock cycles**
- **Randomly selects** one of the transactions involved to **roll back** and **terminate**
  - An SQL error code is sent to the chosen transaction
  - Every lock it had acquired is released
- deadlock detector awakens at a frequency controlled by **dlchktime**, a database configuration parameter
- Set the value of the **diaglevel** dbm configuration parameter to 4, for more logging on deadlocks
- DB2DETAILDEADLOCK event monitor is deprecated but enabled by default. It should be disabled if the new locking event monitors are used to detect deadlocks (recommended)
  - db cfg parameters for new locking monitors are MON\_LOCKTIMEOUT, MON\_DEADLOCK, MON\_LOCKWAIT, MON\_LW\_THRESH

## Isolation Level – Repeatable Read

- Highest level of isolation
  - No dirty reads, non-repeatable reads or phantom reads
- Locks the entire rows scanned for a query
  - Provides **minimum concurrency**
- When to use Repeatable Read:
  - **Changes** to the result set are **unacceptable**
  - **Data stability** is more important than performance

```
SELECT *  
FROM employee  
WHERE id > 4
```




| ID | LASTNAME  | MANAGER | DEPT |
|----|-----------|---------|------|
| 1  | Smith     | Y       | A01  |
| 2  | Martinez  | N       | A01  |
| 3  | Chen      | Y       | E05  |
| 4  | Rousseau  | N       | B15  |
| 5  | Kumar     | N       | A10  |
| 6  | Ivanov    | N       | B15  |
| 7  | Tanaka    | Y       | B15  |
| 8  | Assaf     | N       | C70  |
| 9  | Schneider | Y       | C70  |
| 10 | Rosenberg | N       | E09  |

Employee table

## Isolation Level – Read Stability

- Similar to Repeatable Read but not as strict
  - No dirty reads or non-repeatable reads
  - Phantom reads can occur
- Locks only the retrieved or modified rows in a table or view
- When to use Read Stability:
  - Application needs to operate in a **concurrent** environment
  - Qualifying rows must remain stable for the duration of a transaction
  - If an application does not issue the same query more than once during UOW and does not require the same result set when a query is issued more than once during UOW

```
SELECT *  
FROM employee  
WHERE id > 4
```



| ID | LASTNAME  | MANAGER | DEPT |
|----|-----------|---------|------|
| 1  | Smith     | Y       | A01  |
| 2  | Martinez  | N       | A01  |
| 3  | Chen      | Y       | E05  |
| 4  | Rousseau  | N       | B15  |
| 5  | Kumar     | N       | A10  |
| 6  | Ivanov    | N       | B15  |
| 7  | Tanaka    | Y       | B15  |
| 8  | Assaf     | N       | C70  |
| 9  | Schneider | Y       | C70  |
| 10 | Rosenberg | N       | E09  |

Employee table



## Isolation Level – Cursor Stability

- Default isolation level
  - No dirty reads
  - Non-repeatable reads and phantom reads can occur
- Locks only the row currently referenced by the cursor
- When to use Cursor Stability:
  - Want maximum concurrency while seeing **only committed data**

```
SELECT *  
FROM employee  
WHERE id > 4
```



| ID | LASTNAME  | MANAGER | DEPT |
|----|-----------|---------|------|
| 1  | Smith     | Y       | A01  |
| 2  | Martinez  | N       | A01  |
| 3  | Chen      | Y       | E05  |
| 4  | Rousseau  | N       | B15  |
| 5  | Kumar     | N       | A10  |
| 6  | Ivanov    | N       | B15  |
| 7  | Tanaka    | Y       | B15  |
| 8  | Assaf     | N       | C70  |
| 9  | Schneider | Y       | C70  |
| 10 | Rosenberg | N       | E09  |

Employee table

## Isolation Level – Uncommitted Read

- Lowest level of isolation
  - Dirty reads, non-repeatable reads and phantom reads can occur
- Locks only rows being modified in a transaction involving `DROP` or `ALTER TABLE`
  - Provides **maximum concurrency**
- When to use Uncommitted Read:
  - Querying **read-only** tables
  - Using only **SELECT** statements
  - Retrieving **uncommitted data is acceptable**

```
SELECT *  
FROM employee  
WHERE id > 4
```

| ID | LASTNAME  | MANAGER | DEPT |
|----|-----------|---------|------|
| 1  | Smith     | Y       | A01  |
| 2  | Martinez  | N       | A01  |
| 3  | Chen      | Y       | E05  |
| 4  | Rousseau  | N       | B15  |
| 5  | Kumar     | N       | A10  |
| 6  | Ivanov    | N       | B15  |
| 7  | Tanaka    | Y       | B15  |
| 8  | Assaf     | N       | C70  |
| 9  | Schneider | Y       | C70  |
| 10 | Rosenberg | N       | E09  |

**Employee table**

## DB2 Isolation Levels

| Isolation Level                | Dirty Read | Non-repeatable Read | Phantom Read |
|--------------------------------|------------|---------------------|--------------|
| Repeatable Read ( <b>RR</b> )  | -          | -                   | -            |
| Read Stability ( <b>RS</b> )   | -          | -                   | Possible     |
| Cursor Stability ( <b>CS</b> ) | -          | Possible            | Possible     |
| Uncommitted read ( <b>UR</b> ) | Possible   | Possible            | Possible     |

| Application Type        | High data stability required                                  | High data stability NOT required |
|-------------------------|---|----------------------------------|
| Read-write transactions | Read Stability ( <b>RS</b> )                                  | Cursor Stability ( <b>CS</b> )   |
| Read-only transactions  | Repeatable Read ( <b>RR</b> ) or Read Stability ( <b>RS</b> ) | Uncommitted Read ( <b>UR</b> )   |

## Isolation Level – Currently Committed

- Currently Committed is a variation on Cursor Stability
  - Avoids timeouts and deadlocks
  - Log based:
    - No management overhead

### Cursor Stability

| Situation            | Result |
|----------------------|--------|
| Reader blocks Reader | No     |
| Reader blocks Writer | Maybe  |
| Writer blocks Reader | Yes    |
| Writer blocks Writer | Yes    |



### Currently Committed

| Situation            | Result |
|----------------------|--------|
| Reader blocks Reader | No     |
| Reader blocks Writer | No     |
| Writer blocks Reader | No     |
| Writer blocks Writer | Yes    |

# Isolation Level – Currently Committed

- **Up to DB2 9.5**
  - **Cursor Stability** semantics where writers block readers, is the **default** isolation level
- **Starting DB2 9.7 and in DB2 10**
  - **Currently Committed** semantics are **default** for **NEW** databases
  - Currently Committed is disabled for upgraded databases prior to v9.7, i.e., Cursor Stability semantics are used instead
- **Applications that depend on the old behavior (writers blocking readers) will need to update their logic or disable the Currently Committed semantics**



## Currently Committed – How to Use It

- **cur\_commit** – database configuration parameter
  - **ON**: default for new databases – CC semantics in place
  - **DISABLED**: default value for existing databases prior to DB2 9.7 – old CS semantics in place
- **PRECOMPILE / BIND**
  - **ConcurrentAccessResolution**: Specifies the concurrent access resolution to use for statements in the package.
    - **USE CURRENTLY COMMITTED**
    - **WAIT FOR OUTCOME**

## Specifying Isolation Levels

- Precompile / Bind
  - **ISOLATION** option of **PREP** or **BIND** command
  - Can determine isolation level of a package by executing the following query

```
SELECT ISOLATION FROM syscat.packages
WHERE pkgname = 'pkgname'
AND pkgschema = 'pkgschema'
```

- Statement Level
  - Use the **WITH** {**RR**, **RS**, **CS**, **UR**} clause
  - The **WITH UR** option applies only to read-only operations
    - ensure that a result table is read-only by specifying **FOR READ ONLY** in the SQL statement
  - Overrides the isolation level specified for the package

```
SELECT * FROM tb1 WITH RR
```

## Specifying Isolation Levels

- Dynamic SQL within the current session
  - **SET CURRENT ISOLATION**
  - For all subsequent dynamic SQL statements within the same session
- JDBC or SQLJ at run time
  - SQLJ profile customizer (db2sqljcustomize command)
- CLI or ODBC at run time
  - **CHANGE ISOLATION LEVEL** command changes the isolation level for the current connection

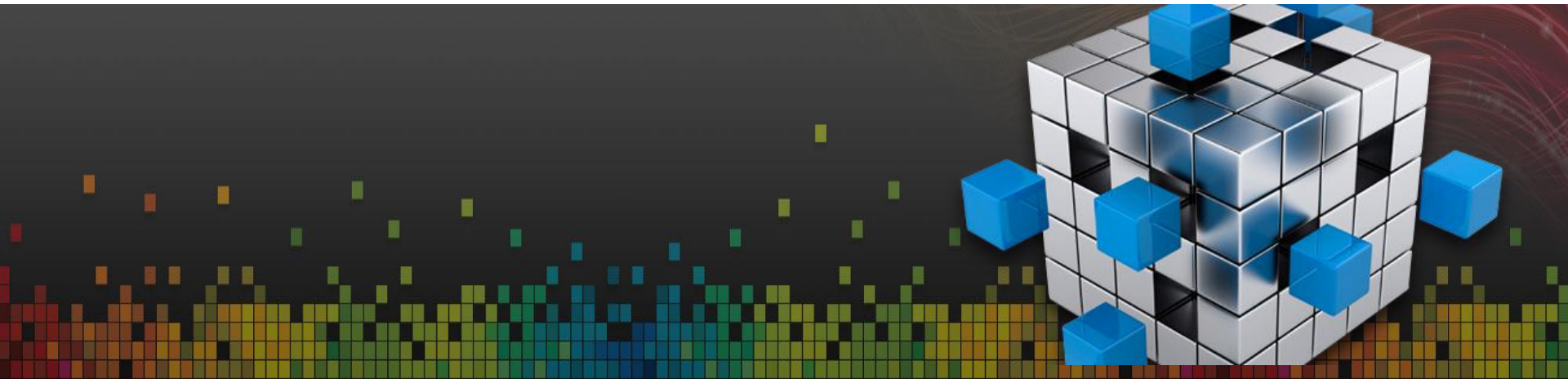
**CHANGE ISOLATION LEVEL TO RR**



## Summary

- Concurrency control in DB2 is based on locking
- DB2 offers 4 isolation levels:
  - Repeatable Read
  - Read Stability
  - Cursor Stability with Currently Committed (default in DB2 10)
    - Cursor Stability (default prior to DB2 9.7)
  - Uncommitted Read
- In DB2 10, with Currently Committed semantics, readers always received the last committed version of the data and don't wait for writers.
- Isolation levels can be modified during the pre-compilation phase or at runtime. It can be changed for the remaining of a session, or for a specific SQL statement only.

# The next steps...



## The Next Steps...

- Complete the online quiz for this module
  - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
  - Find the module and select the quiz
- Provide feedback on the module
  - Log onto SKI, go to “My Learning” page
  - Find the module and select the “Leave Feedback” button to leave your comments



Questions?  
[askdata@ca.ibm.com](mailto:askdata@ca.ibm.com)

