ISOLATION LEVELS in MySQL

Kathleen Durant PhD

CS3200

Performance of Locking

- Locks force transactions to wait
  - Abort and restart due to deadlock wastes the work done by the aborted transaction
  - In practice, deadlocks are rare, e.g., due to lock downgrades approach
- Waiting for locks becomes bigger problem as more transactions execute concurrently
  - Allowing more concurrent transactions initially increases throughput, but at some point leads to thrashing
  - Need to limit maximum number of concurrent transactions to prevent thrashing
  - Minimize lock contention by reducing the time a transaction holds locks and by avoiding hotspots (objects frequently accessed)

Controlling Locking Overhead

- Declaring transaction as "READ ONLY" increases concurrency
- Isolation level: trade off concurrency against exposure of transaction to other transaction's uncommitted changes
  - Degrees of serializability

Isolation level	Dirty Read	Nonrepeatable Read	Phantom
READ UNCOMMITTED	Maybe	Maybe	Maybe
READ COMMITTED	No	Maybe	Maybe
REPEATABLE READ	No	No	Maybe
SERIALIZABLE	No	No	No

Isolation levels

- SERIALIZABLE: obtains locks on (sets of) accessed objects and holds them until the end
- REPEATABLE READ: same locks as for serializable transaction, but does not lock sets of objects at higher level
- READ COMMITTED: obtains X-locks before writing and holds them until the end; obtains S-locks before reading, but releases them immediately after reading
- READ UNCOMMITTED: does not obtain S-locks for reading; not allowed to perform any writes
  - Does not request any locks ever

Hierarchy of Granularity

- **Could represent granularity of locks in a hierarchical structure.**
- **Root node represents entire database, level 1s represent files, etc.**
- **When node is locked, all its descendants are also locked.**
- **DBMS should check hierarchical path before granting lock.**

Lock Modes: State Intent

	IS	IX	S	X
IS	✓	✓	✓	
IX	✓	✓		
S	✓		✓	
X				

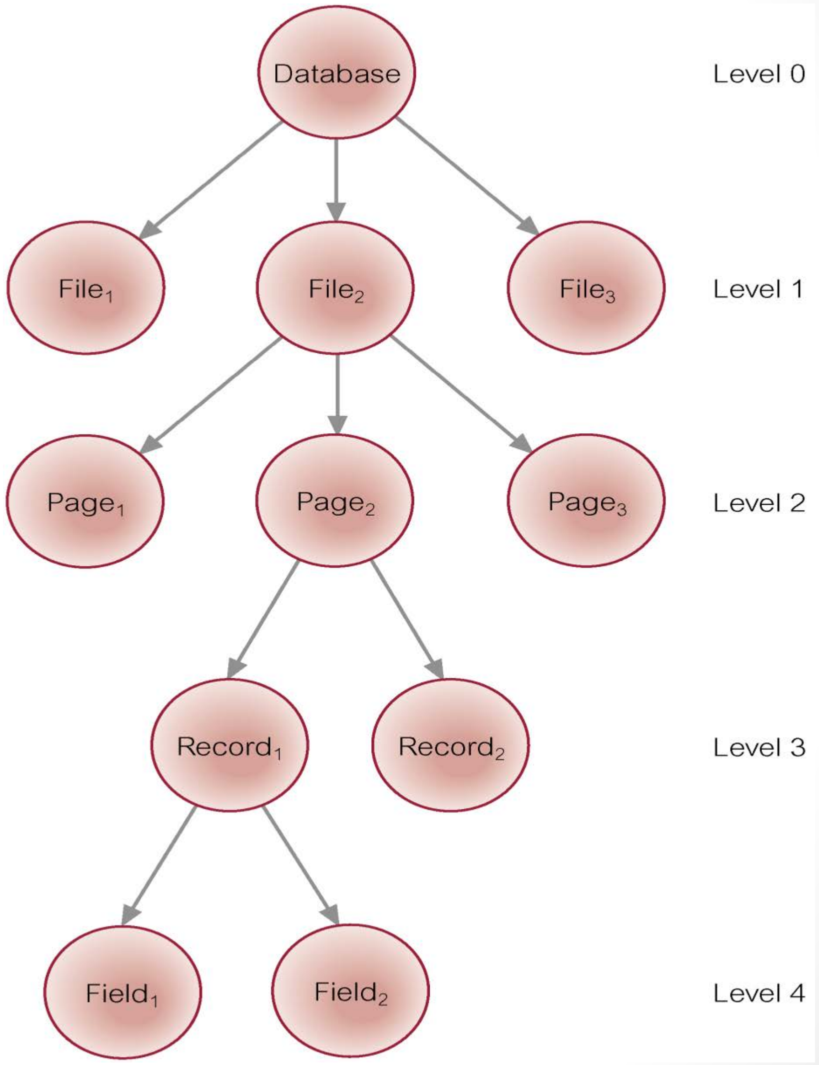
- Allows transactions to lock at each level but with a special protocol using new ‘intentions’ locks.
    - Can be read intent (intent share) or write intent (intent exclusive)
- Before viewing an item, transaction must set intention locks on all its ancestors (higher level containers)
- Locks are applied top-down, released bottom-up

Granularity of Data Items

- **Size of data items chosen as unit of protection by concurrency control protocol.**
- **Ranging from coarse to fine:**
  - **The entire database.**
  - **A file.**
  - **A table.**
  - **A page (or area or database spaced).**
  - **A record.**
  - **A field value of a record.**

Levels of locking

- Each transaction starts from the root of the hierarchy
- To get S or IS lock on a node, must hold IS or IX on parent node
- To get X or IX on a node, must hold IX on parent node
- Must release locks in bottom-up order
- Equivalent to directly setting locks at the leaf levels



Granularity of Data Items

- **Tradeoff:**
  - **coarser, the lower the degree of concurrency;**
  - **finer, more locking information that is needed to be stored.**
- **Best item size depends on the types of transactions.**

ISOLATION LEVEL: MYSQL

- **SET TRANSACTION** ISOLATION LEVEL l*evels*;
  - SERIALIZABLE
  - REPEATABLE READ
  - READ COMMITTED
  - READ UNCOMMITTED
- Default is that the command affects the next transaction
- Can also set the ISOLATION LEVEL for the current session and globally
  - SET [GLOBAL|SESSION] TRANSACTION ISOLATION LEVEL l*evels*;
  - **GLOBAL** applies globally for all subsequent sessions. Existing sessions are unaffected.
  - **SESSION** applies to all subsequent transactions performed within the current session
- Can also define the access method for the query
  - **SET TRANSACTION READ ONLY**
  - **SET TRANSACTION READ WRITE**

INNODB and Transactions

- All user activity occurs inside a transaction
- If autocommit mode is enabled, each SQL statement forms a single transaction on its own.
- Perform a multiple-statement transaction by starting it with an explicit START TRANSACTION
- autocommit mode is disabled within a session with SET autocommit = 0,
  - The session will have a transaction open until it is explicitly closed
  - Issue commit or rollback to close the transaction
- Default InnoDB Isolation level is REPEATABLE READ
- InnoDB performs row level locking
  - Only if two transactions try to modify the same row does one of the transactions wait for the other to complete

InnoDB and locks

- InnoDB implements standard row-level locking where there are two types of locks
  - (S) shared locks
    - permits the transaction that holds the lock to read a row.
  - (X) exclusive locks
    - permits the transaction that holds the lock to update or delete a row.
- InnoDB supports *multiple granularity locking* which permits coexistence of record locks and locks on entire tables.
  - Intention locks are table locks in InnoDB that indicate which type of lock a transaction will require later for a row in that table.
  - Intention shared (IS) Transaction T intends to set *S* locks on individual rows in table t. (SELECT … LOCK IN SHARE MODE)
  - Intention exclusive(IX) Transaction T intends to set *X* locks on individual rows in table t (SELECT … LOCK FOR UPDATE)

http://dev.mysql.com/doc/refman/5.7/en/innodb-locking-reads.html

Granting locks

- A lock is granted to a requesting transaction if it is compatible with existing locks
- A transaction waits until the conflicting existing lock is released
- If a lock request conflicts with an existing lock and cannot be granted because it would cause deadlock, an error occurs
- Main purpose of *IX* and *IS* locks is to show that someone is locking a row, or going to lock a row in the table.
- SHOW ENGINE INNODB STATUS;
  - To report on any transactions and deadlock conditions.

Summary

- InnoDB supports transactions
- MySQL allows a user/system administrator to determine the level of Isolation for transactions
- MySQL implements intent lock at the table level
- MySQL provides commands that allow you to list the transactions, locks currently active in the system