Assignment 10 Concurrency and Isolation Level

Name _____Christopher Holmes_____

Reading:

- Chapter 14 in murach textbook.
- Watch my videos posted on iLearn about transactions and locking.

Part 1

Based on your reading answer the following questions.

1. What is the purpose of concurrency control?

The purpose of concurrency control is to make sure that two processes do not try to update the same record at the same time that would cause integrity problems with the database.

2. Explain the trade-off that exist in concurrency control.

When locking tables, rows, databases, it can consume lots of resources and lead to improper utilization of the database. These tradeoffs need to be considered when designing a database and an application so that you can make it the most efficient and utilize the database properly.

3. Explain the difference between concurrent transactions and simultaneous transactions. How many CPUs are required for simultaneous transactions?

In a concurrent transaction, you can have two or more people accessing the database and using it. A concurrent transaction only requires a single CPU. A simultaneous transaction will require to have two or more cpu's to accomplish the same results.

4. Define the terms *dirty read, nonrepeatable read*, and *phantom read.* How are these terms related to isolation level?

Dirty read: Happens when a transaction is allowed to read data from a row that was modified by another process, but has yet to be committed. Nonrepeatable read: Happens when executing a transaction, a row is received twice and the values within the row differ between reads. Phantom read: Happens when executing a transaction and new rows are added by another process to the records that are being read

Assignment 10 Concurrency and Isolation Level

5. What is lock granularity?

Lock granularity tells what resources have been locked by a single lock from a process. Since a row, page, index, table, or entire database can be locked, it is important to be able to know what level is locked when trying to execute a transaction.

6. Explain the difference between exclusive lock and shared lock.

An exclusive lock makes it so that nobody can read the object until that exclusive lock has been released. A shared lock cannot be obtained when an exclusive lock is applied. A shared lock allows for multiple people to have a lock on the data so that they can read it, but the data cannot be changed with an exclusive lock while a shared lock is applied.

7. What is deadlock? How can it be avoided? How can it be resolved when it occurs?

A deadlock is when two different transactions are trying to access data that is already locked by another transaction. You can avoid a deadlock by having preventative measures in place to prevent a deadlock, or having ways to break the deadlock after it happens. To resolve, the DBMS will select the transaction to cancel and revert the entire transaction until the required resource is available.

8. Explain the difference between optimistic and pessimistic locking.

Pessimistic locking places a lock on a row once on of it's attributes is changed. Any subsequent transaction that tries to make changes to the row that has already been locked will be forced to wait until the first transaction is completed.

An optimistic lock is one where it can be assumed that multiple transactions can be completed at the same time without affecting each other. A lock is not acquired on the row until the changes have been posted with an optimistic lock.

9. Explain the meaning of the expression *ACID transaction*.

ACID consists of Atomicity, Consistency, Isolation, and Durability. When a transaction is able to pass all four of these properties, it is said to be an ACID transaction.

Atomicity: States that all transactions should complete at a time or none of them should

Consistency: Data must be present in a consistent state from the start of the transaction to the finish
Isolation: When a transaction is in progress and has yet to complete, it should isolate itself from any other transactions
Durability: When a transaction completes, the changes made by the transaction should not be reversed or incomplete when succesfful.

Two important application sql variables related to transactions.

AUTOCOMMIT: a single sql statement always has ACID properties. If you have a multiple row update statement, either the entire statement will be successful or nothing will occur. If there is an error during processing of the update, any modifications already performed will be rolled back. But if you want a transaction to span multiple sql statement, then auto commit must be turned OFF. In MySQL this is done with the statement

```
SET AUTOCOMMIT = 0;
```

You can change auto commit to ON by setting the value to 1. See https://dev.mysql.com/doc/refman/5.7/en/commit.html for more information.

TRANSACTION ISOLATION LEVEL:

You can set the value of transaction isolation to ***read uncommitted, read committed, repeatable read*** and ***serializable***. The values determine if records will be locked and whether the lock will be held temporarily or until transaction commit. See https://dev.mysql.com/doc/refman/5.7/en/set-transaction.html, and https://dev.mysql.com/doc/refman/5.7/en/innodb-transaction-isolation-levels.html for more details on MySQL.

Note: MySQL has two database engines MyISAM and INNODB. MyISAM does not support transactions and locking. MyISAM is used in the WORLD schema tables. INNODB is used by default when you create tables. So unless you specify MyISAM, you will have support for transactions.

Part 2

You can see the effects of transaction isolation level for yourself by doing the following steps. You must have TWO CONNECTIONS to the DMBS server that that you can do two concurrent transactions.

1. Start mysql workbench.

   Open and run **A11_setup.sql** script to create the cst438 schema and the cst438.city table data.

2. Connect and open a query tab. In the query tab window execute the statements:

   SET AUTOCOMMIT=0;

   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

3. **IMPORTANT**: **Now go back to the HOME TAB and do a second connection** and open a query tab. You now have 2 query tabs but each tab has a separate connection to the server. Enter the same command in query tab 2

SET AUTOCOMMIT=0;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

4. Use scripts A11_client1.sql and A11_client2.sql to help do the following exercise.

	Query tab connection 1	**Query tab connection 2**
1	## step 1 set autocommit=0; set transaction isolation level repeatable read; select @@autocommit, @@tx_isolation; ## pause here. go to step 2 at client 2	
2		## step 2 set autocommit=0; set transaction isolation level repeatable read; select @@autocommit, @@tx_isolation; select population from cst438.city where id=1820; ## pause here. go to step 3 at client 1
3	## step 3 ## the next update statement should hang ## wait several seconds, it will timeout. ## why? Client#2 has a lock on the record for id=1820. update cst438.city set population=population+1000 where id=1820;	
4		## step 4. change isolation level ## to Read Commited. commit; set transaction isolation level read committed; select population from cst438.city where id=1820; ## pause here. Go to step 5 at client 1.
5	## step 5. Change isolation to read committed. rollback; set transaction isolation level read committed; ## the update statement will not hang because of read committed. update cst438.city set population=population+1000 where id=1820; select population from cst438.city where id=1820; ## pause here. go to step 6 in client 2.	

6		## step 6. re-read the population data. ## you should see the same value even though ## client 1 has updated it. ## why? client 1 has not commited the change yet. select population from cst438.city where id=1820; ## pause here. go to step 7 in client 1
7	## step 7. Now commit the transaction. commit; ## go to step 8 in client 2	
8		## step 8. re read the population. This time you see the new value. select population from cst438.city where id=1820; commit;

Questions

10. Explain in your own words why step 3 failed and the update statement done by connection 1 timed out?

> As the version of MySQL handles transactions differently now, I did not get a timeout due to a lock being placed on the data. Per the forum post explaining that it uses versioning now, as the statement is written, it works. I was able to get a lock when I used a serializable isolation. The reason I get a lock and it times out with a serializable isolation is because this is a more restrictive isolation. Generally, you want to use a serializable isolation when it would be detrimental to your data if it were to change while you are accessing it.

11. What is the difference between using REPEATABLE READ and READ COMMITTED regarding reading and locking data? what about writing and locking data?

> When you use a read committed, you are guaranteeing that any data that you read was committed as well the moment that it's read. This will prevent you from getting dirty reads. There is no guarantee that data will not change between reads though.
> When you use a repeatable read, you are guaranteeing that not only are the conditions of a read committed satisfied, but that also the data is not going to change between reads.

Assignment 10 Concurrency and Isolation Level

Part 3. Design your own demo of concurrency and isolation level.

You have a table that keeps track of inventory of part numbers. Create a table that has id as primary key, partName and inventoryCount as other columns. Load the table with 3 rows of data for part ids 1, 2, and 3 and inventoryCount of 2 for each part.

Client 1 needs to use 2 of part 1 and 2 of part 2. Client one first reads the inventory count of parts 1 and 2 to verify that there is sufficient quantity of hand. Then updates the new count of part 1 and the new count of part 2.

Concurrently client 2 needs to use 2 of part 1 and 2 of part 3.

First run the scripts using autocommit=0 and read committed isolation level. Is the inventory count correct at the end of both client transactions?

The inventory count at the end of both transactions is correct.

Now rerun the scripts but use serializable isolation level. What is the difference in behavior? Do you get the correct inventory count at the end?

The difference in behavior with the serializable isolation is that both transactions cannot run at the same time. When transaction 1 runs, it places a lock on the data, and transaction 2 cannot run until the first is complete. You do get the correct inventory count at the end.

Is you rerun the scripts with repeatable read isolation level, do you get the correct behavior?

The inventory count is correct.

To make the scripts easier to work with you can use variables in the script

To read current inventory of parts 1 and 2 you can code

```
select count into @countp1 from inventory where id = 1;
select count into @countp2 from inventory where id = 2;
# pause here so that user can verify sufficient quantity
# and client 2 can run concurrently
update inventory set count=@countp1 - 2 where id = 1;
update inventory set count=@countp2 - 2 where id = 2;
```