CS 348 - Homework 6: Transactions, concurrency, and graph databases (Neo4J).

(100 Points)

Spring 2022
Due on: **4/13/2022 at 11:59 pm**
This assignment is to be completed by individuals. You should only talk to the instructor, and the TAs about this assignment. You may also post questions (and not answers) to Campuswire.

There will be a 10% penalty if the homework is submitted 24 hours after the due date, a 20% penalty if the homework is submitted 48 hours after the due date, or a 30% penalty if the homework is submitted 72 hours after the due date. The homework will not be accepted after 72 hours, as a solution will be posted by then.

Submission Instruction: For questions 2 to 5, write your answers on a word or text document and generate a pdf file. **Upload the pdf file to Gradescope**.

For questions 6 to 13, write your answers (queries) in the HW6_Neo4j.py file. **Upload the file to Brightspace.**

Transactions and Concurrency.

Select the isolation level for each of the following application features. Each transaction should specify the appropriate isolation level that maximize concurrency and reduce overhead (locking overhead) but still provides the required level of consistency the application requires. Explain why the more restrictive isolation level should not be used. Also, explain why the less restrictive isolation level should not be used. Include a schedule or scenario where the less restrictive isolation level violates the level of consistency the application requires.

Question 1) (zero points, solution is provided)

Biker Performance Application: Consider a speed sensor attached to each bike in a bike race. The sensor sends data periodically (e.g., every 2 seconds) about the current speed of a biker to the application backend. The backend has transaction T1 that inserts the data into the following table: bikerStats(bikerID, raceID, timestamp, speed). Another transaction T2 runs every 10 seconds to update a dashboard with the approximate average speed of each biker since the beginning of the trip. Note that multiple instances of transaction T1 run simultaneously to

insert data for different bikers. On the other hand, there is only one instance of T2 that updates the dashboard of all bikers.

Solution:
Transactions:

T1:
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
Start Transaction;
Insert Into bikerStats(bikerID, tournamentID, timestamp, speed) VALUES (v1, v2, v3, v4);
// v1 to v4 are data sent from a speed sensor
Commit;

T2:
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
Start Transaction;
SELECT bikerID, AVG(Speed)
Group By BikerID;

// Send query results to the dashboard module/function
commit;

Why did not you select a more restrictive isolation level?

T1 and T2 accessing the same biker data can have interleaving operations. The most interesting schedule of operations is for T1 to insert a new row in bikeStats, T2 computes the average, then T1 commits/aborts. Since T2 does not require a read lock (read uncommitted), T2 reads the uncommited new row that T2 inserted. If T2 aborts for some reason (e.g., system crash), T2 will report an average speed that is not consistent with the data in bikerStats. However, since T2 needs only to report an approximate average, read uncommited is the appropriate level that provides an acceptable consistency for the application and reduces the overhead of obtaining a read lock(s) over the rows of a certain biker.

Why did not you select a less restrictive isolation level?

We choose read uncommitted, which is the least restrictive isolation level.

Question 2) (10 points)

Consider the transaction T1 that books a flight to a customer. T1 works on the table Flight(number, origin, destination, available_seats), where available_seats records the remaining seats in a given flight. Multiple instances of T1 work concurrently to serve different customers.

T1 code:

SET TRANSACTION ISOLATION LEVEL ????
Start transaction;
Select available_seats into AV From Flight Where number = N;
If AV > 0
    // add customer to tickets table
    Update Flight set available_seats = available_seats - 1;
    Print confirmation

Commit;

Why did not you select a more restrictive isolation level?

Why did not you select a less restrictive isolation level?

Question 3) (10 points)

A State's department of transportation operates sensors on highways to track traffic. Suppose we have the table sensor_table(highway, direction, station#, speed). Two transactions T1 and T2 use this table as follows. T1 gets the data from a sensor on a highway and inserts the data into the table. T2 on the other hand reads the table and display approximate statistics on a dashboard.

T1 Code:
SET TRANSACTION ISOLATION LEVEL ????
Start transaction;
Get data (d1, d2, d3, d4) from a sensor

Insert into sensor_table Values (d1, d2, d3, d4);
Commit;

T2 Code:
SET TRANSACTION ISOLATION LEVEL ????
Start transaction;

Select highway, station, AVG(speed)
From sensor_table
Group By highway, station

Display data on the dashboard

Commit;

Why did not you select a more restrictive isolation level?

Why did not you select a less restrictive isolation level?

Question 4) (10 points)

Ticket archiving: Consider an IT department ticket system. Tickets are created by employees for various technical issues. When an IT staff resolve those issues a transaction update the table: tickets(number, description, date, resolved), by changing resolved from false to true. Other transactions work on the table as well, such as the ones that create new tickets or cancel (delete) tickets. Consider the T1 transaction whose task is to archive old resolved tickets by copying them from the tickets table to the tickets_archive table and then removing the resolved tickets from tickets table. T1 must keep both tables consistent. For example, the tickets_archive table can be used in billing or budget purposes.

Transaction:

```
SET TRANSACTION ISOLATION LEVEL ????
Start transaction;
Insert into tickets_archive
        select * from tickets Where resolved = True;

Delete From tickets Where resolved = True;

commit;
```

Why did not you select a more restrictive isolation level?

Why did not you select a less restrictive isolation level?

Google Cloud Platform:

Q5 (5 points): Create a MySQL instance in Google Cloud Platform (GCP). Create a database and a table with a schema of your choice. Insert data or upload a data set (you can check the data.gov portal for free data sets in .csv format). Run two queries on your data. Include screenshots for the results of the two queries (take the screenshots while running the queries in the cloud shell).
Please remember to terminate your MySQL instance after you finish. Also, if you use

cloud storage to upload a data set, delete your data set file after you finish. If you choose to upload a data set, make sure the file size is not too large. Remove the headers from the csv file before loading the data. For details about cloud computing and using the GCP, please refer to the demo that will be posted in Brightspace in Week 13. Make sure you redeem the GCP credits (credits will be posted on Campuswire in Week 13).

Graph Databases (neo4J).

Question 6 to 13) (65 points)

Please check Questions5to13.txt file for the neo4j questions.