HW 06
Junseok Oh
CS348
04-10-22

Question 2)
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.
SET TRANSACTION ISOLATION LEVEL Repeatable Read
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?

- I chose this isolation level because Repeatable Read locks only individual reads not whole table, while Serializable does lock the entire statements. Therefore, I chose Repeatable Read not Serializable.

Why did not you select a less restrictive isolation level?

- If this is less restrictive than the one I chose, when two different people are trying to reserve a same available seat in each flight, its object has to be locked and we should not let others to change the row. If we choose Read Committed, two people might read a data for the same seat, and book at the same time, which should not happen.

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 Read Uncommitted
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 Read Uncommitted
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?

- T1 is reading it from a sensor on a highway and since T1 does not require a read lock (read uncommitted), T1 reads the uncommited new row that T1 inserted.
- Since T2 is displaying approximate statistics, 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 highways traffic.

Why did not you select a less restrictive isolation level?

- For T1, we choose read uncommitted, which is the least restrictive isolation level.
- For T2, we choose read uncommitted, which is the least 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 Serializable
Start transaction;
Insert into tickets_archive
    select * from tickets Where resolved = True;
Delete From tickets Where resolved = True;
commit;

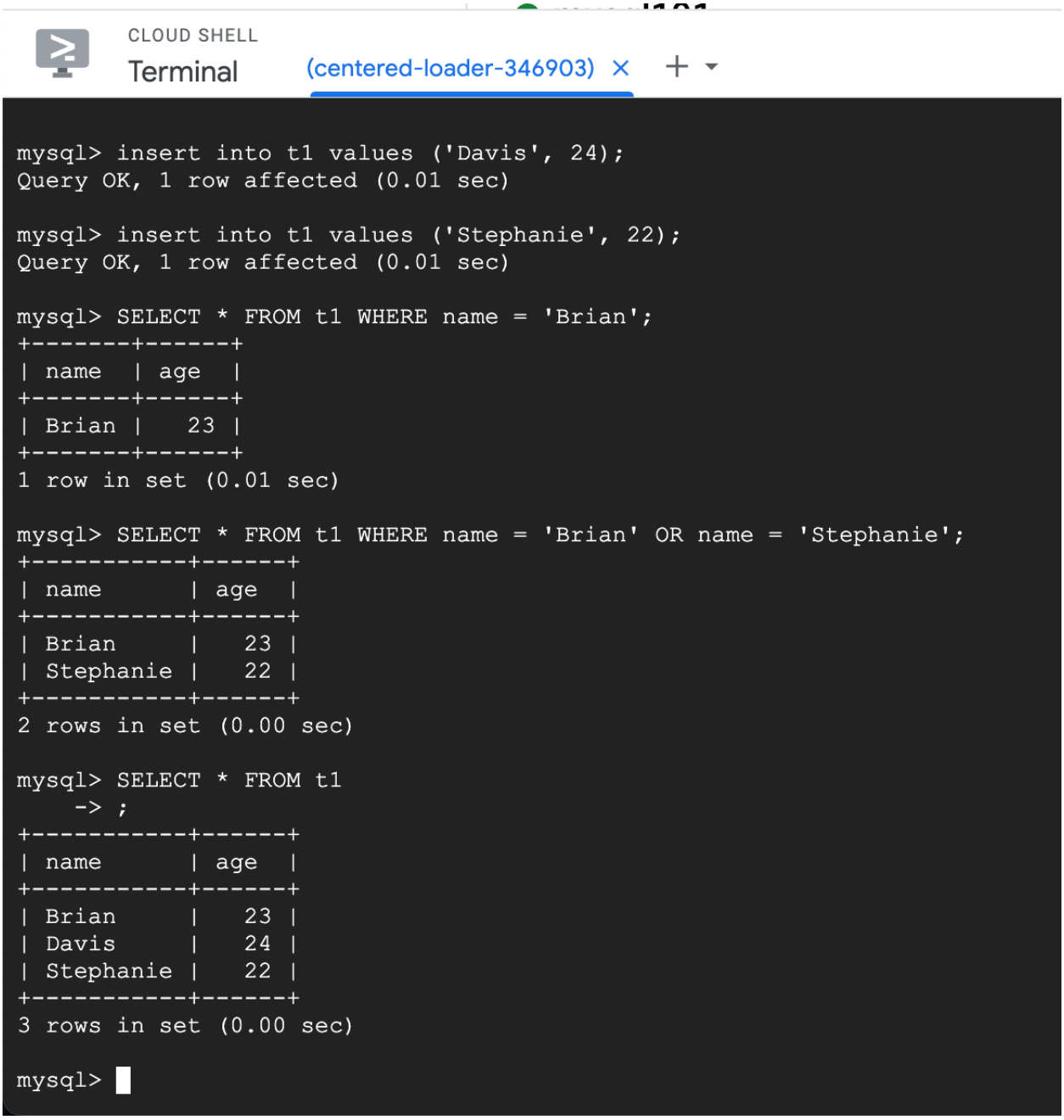
Why did not you select a more restrictive isolation level?

- For this table, I choose serializable, which is the most restrictive isolation level.

Why did not you select a less restrictive isolation level?

- Because this table must keep both tables consistent, it is not allowed to have phantom reads. To avoid phantom reads, we must make the table itself to be an execution of operations in which concurrently executing transactions appears to be serially executing. Therefore, we must choose isolation level as serializable, whose effect on any consistent database instance is guaranteed to be identical.

Question 5)


CLOUD SHELL
Terminal
(centered-loader-346903)
mysql> insert into t1 values ('Davis', 24);
Query OK, 1 row affected (0.01 sec)
mysql> insert into t1 values ('Stephanie', 22);
Query OK, 1 row affected (0.01 sec)
mysql> SELECT * FROM t1 WHERE name = 'Brian';
+-------+------+
| name | age |
+-------+------+
| Brian | 23 |
+-------+------+
1 row in set (0.01 sec)
mysql> SELECT * FROM t1 WHERE name = 'Brian' OR name = 'Stephanie';
+-----------+------+
| name | age |
+-----------+------+
| Brian | 23 |
| Stephanie | 22 |
+-----------+------+
2 rows in set (0.00 sec)
mysql> SELECT * FROM t1
-> ;
+-----------+------+
| name | age |
+-----------+------+
Brian	23
Davis	24
Stephanie	22
+-----------+------+
3 rows in set (0.00 sec)
mysql>