Lab 08 - Concurrency

Due Apr 8 at 11:59pm **Points** 40 **Questions** 12
Available after Apr 7 at 9am **Time Limit** None **Allowed Attempts** Unlimited

Instructions

For this lab we are going to be working on various concurrency control and transaction management concepts.

Note, you may not resubmit answers to true-false or drop-down questions.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	1,533 minutes	38 out of 40

Correct answers are hidden.

Score for this attempt: **38** out of 40
Submitted Apr 8 at 10:43am
This attempt took 1,533 minutes.

Part 1: Transactional Errors

Making use of the SaleCo database structure and the provided tables, please write a series of SQL statements for two concurrent transactions that would create the following errors. Assume that there are no concurrency controls in place.

To answer these questions, please provide a table with two transactions and the SQL commands you are running to create the errors. Make sure that START TRANSACTION and COMMIT (or ROLLBACK) are present in the table.

For a table example:

Transaction 1	Transaction 2
START TRANSACTION	START TRANSACTION
SELECT cus_code from customer;	UPDATE customer set cus_code = 10100 WHERE cus_code = 10000;
COMMIT	COMMIT

Question 1

4 / 4 pts

Create a table of SQL statements that will cause a 'read on uncommitted data' error regarding a product's price. (Make sure to include a ROLLBACK operation.)

Your Answer:

Transaction 1	Transaction 2
START	START
SELECT P_DESCRIPT FROM product WHERE P_PRICE = 39.95;	
	UPDATE product SET P_PRICE = 42.22 WHERE P_PRICE = 39.95;
SELECT P_DESCRIPT FROM product WHERE P_PRICE = 39.95;	
	ROLLBACK
COMMIT	COMMIT

Question 2

4 / 4 pts

Create a table of SQL statements that will cause a 'lost update' error by incorrectly overwriting a customer's current balance.

(Note that you will need to store a value in a variable to allow for an overwrite. This means that you will need to DECLARE a variable and SELECT INTO it.)

Your Answer:

Transaction 1	Transaction 2
START	START
SELECT CUS_BALANCE INTO cusbalance FROM customer WHERE CUS_CODE = '10018';	
	SELECT CUS_BALANCE INTO cusbalance FROM customer WHERE CUS_CODE = '10018';
	UPDATE customer SET CUS_BALANCE = cusbalance + 50 WHERE CUS_CODE = '10018';
UPDATE cusbalance SET CUS_BALANCE = cusbalance + 100 WHERE CUS_CODE = '10018';	
	COMMIT
COMMIT	

Something gets read incorrectly.

one transaction hold onto a value. (Lost update) -> monday

Change data so its doing it on current balance on saleco.

4sql statements

declare a variable to store some value,

select into that variable,

Question 3

4 / 4 pts

Create a table of SQL statements that will cause a 'phantom read' error by incorrectly returning a list of all vendors.

(Please use "-" to indicate a running set of insertion statements. You do not need to write out the values for the INSERT commands, just what table they are inserting into.)

Your Answer:

Transaction 1	Transaction 2
START	START
SELECT * FROM vendors;	
	INSERT Into Vendor(V_CODE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE, V_STATE, V_ORDER) VALUES (26696, Pepsi, Blanchard, 805,754-9148, CA, Y);
	-
	-
	COMMIT;

SELECT * FROM vendors;	
COMMIT;	

- dash means its still running

Question 4

1 / 1 pts

What is the minimum transaction isolation level needed to avoid the phantom read error from question 4 above?

- Read Uncommitted
- Read Committed
- Repeatable Read
- [✓ Serializable]

Question 5

1 / 1 pts

What is the minimum transaction isolation level needed to avoid the inconsistent retrieval error from question 2 above?

- Read Uncommitted
- [✓ Read Committed]
- Repeatable Read
- Serializable

Partial

Question 6

2 / 4 pts

Part 2: Isolation Level Analysis

Please take a look at the following simple table, Pets(name, age) and the following transaction T:

```
START TRANSACTION;
(1) SELECT count(*) from Pets;
<other read statements>
(2) SELECT count(*) from Pets;
COMMIT;
```

If we assume that (1) and (2) run with complete atomicity, what is the minimum isolation level that will make sure (1) and (2) return the same count for each of the following conditions?

1. There is no information available about any other concurrent transactions. Serializable
2. Every other transaction running on the system is a read. Read Uncommitted
3. Every other transaction running is declared at a Serializable isolation level and may involve updates. Read Uncommitted
4. Every other transaction running is declared at a Serializable isolation level and may involve insertions Read Uncommitted

Answer 1:

Serializable

Answer 2:

Read Uncommitted

Answer 3:

Read Uncommitted

Answer 4:

Read Uncommitted

Question 7 **5 / 5 pts**

Part 3: Locking Protocols

Assuming a simple shared and exclusive locking scheme please fill in the following table to indicate if the lock manager (L) will grant (“G”) or deny (“D”) a transaction locking request. T1, T2 and T3 are concurrent transactions with aligned times (t).

Also assume that there is no deadlock prevention running and that locks cannot preempt each other. (i.e. A shared lock cannot cancel an exclusive lock and an exclusive lock cannot cancel a shared lock.)

S(x) indicates a request for a shared (read) lock.

E(x) indicates a request for an exclusive (write) lock.

t	T1	T2	T3	L
1	E(B)			G
2		E(A)		G
3			S(A)	D
4			S(C)	G
5		E(B)		D
6	S(C)			G
7		E(C)		D

Answer 1:

G

Answer 2:

G

Answer 3:

D

Answer 4:

G

Answer 5:

D

Answer 6:

G

Answer 7:

D

Question 8

5 / 5 pts

Assuming 2PL please fill in the following table to indicate if the lock manager will grant or deny a lock request. Use the same structure as the previous question with the following addition to the table:

U(x) indicates a lock release.

t	T1	T2	T3	L
1	E(B)			G
2		E(A)		G

3	U(B)			
4			S(C)	G
5		U(A)		
6	S(C)			D
7			U(C)	
8		E(C)		D

Answer 1:

G

Answer 2:

G

Answer 3:

G

Answer 4:

D

Answer 5:

D

Question 9

5 / 5 pts

Construct a waits-for graph for the following schedule assuming no lock is ever released.

Please upload a png of your graph image. (draw.io works well for this.)

t	T1	T2	T3
1	S(A)		
2		E(B)	
3			S(C)
4			S(A)
5	E(A)		
6		E(C)	
7			S(B)
8	S(C)		

[Lab8_9.png (https://cilearn.csuci.edu/files/2783559/download)](https://cilearn.csuci.edu/files/2783559/download)

Question 10 **5 / 5 pts**

Part 4: Serializability

For the following schedule, construct a precedence graph. (Draw.io is useful for constructing graph images.)

Please upload a png image of your completed graph.

t	T1	T2	T3
1		Read (A)	
2			Read(C)
3	Write (B)	Write (D)	
4		Write (E)	
5	Read (H)		Write (G)
6	Read (I)	Write (F)	Read (D)
7	Write (A)		Write (H)
8		Write (J)	Read (B)
9	Read (K)		Read (E)
10			Read (G)

[Lab8_10.png (https://cilearn.csuci.edu/files/2783555/download)](https://cilearn.csuci.edu/files/2783555/download)

Question 11

1 / 1 pts

The previous schedule is conflict serializable.

[✓ True]

False

Question 12

1 / 1 pts

Please explain your reasoning for the previous true / false answer regarding conflict serializability.

Your Answer:

The graph is acyclic. The schedule is conflict serializable because we cannot get to every edge and back again on the graph given.

Quiz Score: **38** out of 40