83/100

COMP518 Assignment 3

Peter Prescott (201442927)

Question 1

46/60

14/14

Create the following relational database schema…:

- *Employee(eid, ename, age)*
- *Department(did, dname, dtype, address)*
- *WorksIn(eid, did, since)*
- *Product(pid, pname, ptype, pcolor)*
- *Sells(did, pid, quantity)*

Entity Integrity is ensured by requiring primary keys to be NOT NULL.
Referential Integrity is ensured by the necessary FOREIGN KEY CONSTRAINTS.

```
CREATE TABLE Employee (
    eid INT NOT NULL AUTO_INCREMENT,
    ename VARCHAR(64),
    age TINYINT,
    PRIMARY KEY(eid)
);

CREATE TABLE Department(
    did INT NOT NULL AUTO_INCREMENT,
    dname VARCHAR(64),
    dtype VARCHAR(64),
    address VARCHAR(256),
    PRIMARY KEY(did)
);

CREATE TABLE WorksIn(
    eid INT NOT NULL,
    did INT NOT NULL,
    since DATE,

    PRIMARY KEY(eid, did),

    CONSTRAINT fk_eid
    FOREIGN KEY (eid)
    REFERENCES Employee(eid)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT fk_did
    FOREIGN KEY (did)
    REFERENCES Department(did)
        ON UPDATE CASCADE
        ON DELETE CASCADE

);


CREATE TABLE Product(
    pid INT NOT NULL AUTO_INCREMENT,
    pname VARCHAR(64),
    ptype VARCHAR(64),
    pcolor VARCHAR(16),
    PRIMARY KEY(pid)
);


CREATE TABLE Sells(
    did INT NOT NULL,
    pid INT NOT NULL,
    quantity TINYINT,

    PRIMARY KEY(did, pid),

    CONSTRAINT fk_did_sells
    FOREIGN KEY(did)
    REFERENCES Department(did)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

    CONSTRAINT fk_pid
    FOREIGN KEY(pid)
    REFERENCES Product(pid)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

1.2 *Provide MySQL queries for the following…*

3/3

```
PROMPT (A) Names of departments selling blue products

SELECT DISTINCT D.dname
FROM Department D, Product P, Sells S
WHERE D.did = S.did
  AND S.pid = P.pid
  AND P.pcolor = 'blue';
```

6/6

```
PROMPT (B) Name of departments selling products of type tool and
toy

SELECT D.dname
FROM Department D
WHERE D.did IN (SELECT S.did
  FROM Sells as S
  WHERE S.pid in (SELECT P.pid
    FROM Product as P
    WHERE P.ptype = 'tool'))
AND D.did IN (SELECT S.did
  FROM Sells as S
  WHERE S.pid in (SELECT P.pid
    FROM Product as P
    WHERE P.ptype = 'toy'));
```

3/8

```
PROMPT (C) Names of departments which sell blue products and do
not have any employee older than 40

SELECT DISTINCT D.dname
FROM Department D, Product P, Sells S
WHERE D.did = S.did
  AND S.pid = P.pid
  AND P.pcolor = 'blue'
AND D.did IN (SELECT D.did
  FROM Employee E, WorksIn W, Department D
  WHERE E.eid = W.eid AND W.did = D.did
  GROUP BY D.did
  HAVING MAX(E.age) < 41
);
```

This returns departments with at least one employee no older than 40.

6/9

```
PROMPT (D) For each department, id and age of oldest employee.

SELECT D.did, MAX(E.age)
FROM Employee E, WorksIn W, Department D
```

What if a department has no employee? It should be in the outcome.

```
WHERE E.eid = W.eid AND W.did = D.did
GROUP BY D.did;
```

9/9

```
PROMPT (E) Names of employees older than at least one employee
working in 'Central' dept.

SELECT E.ename
FROM Employee E
WHERE E.age > ANY(SELECT age FROM Employee
  WHERE eid IN (SELECT W.eid
    FROM WorksIn W
    WHERE W.did IN (SELECT D.did
    FROM Department D
    WHERE D.dname = 'Central'
      )
    )
  );


PROMPT (F) Names of employees working in depts which sell at
least five types of products
```

5/11

```
SELECT E.ename
FROM Employee E, WorksIn W, Department D
WHERE E.eid = W.eid AND W.did = D.did
AND 5 <= (SELECT COUNT(S.quantity)
          FROM Sells AS S
          WHERE D.did = S.did);
```

Your solution doesn't count how many different TYPES of products each department sells.

What if the quantity in Sells is zero? You should add the condition that quantity > 0 in the WHERE clause.

27/30

Question 2

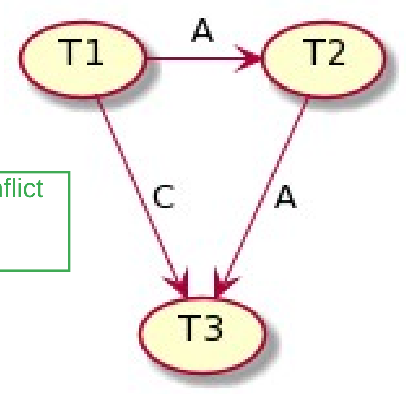
For each schedule, (i) create the precedence graph of the conflicts; (ii) show whether the schedule is conflict-serializable or not; (iii) explain whether this schedule can occur by use of two-phase locking.

First we note that Two-Phase locking involves a 'growing phase' in which the transaction requires locks, and a 'shrinking phase' in which the transaction releases locks. If every transaction in a schedule follows 2PL, it is conflict serializable, but the reverse does not hold. To show a schedule cannot occur by two-phase locking it is sufficient to find one operation that is impossible to reconcile with 2PL. To show that it can, we show when locking could occur such that each transaction has the necessary growing and shrinking phases. We will add the notation LJ(X) and UJ(X) to the suggested RJ(x) and WJ(X), to mean that transaction TJ respectively locks and unlocks data item X.

1. R1(A), R1(B), W1(A), R2(A), R1(C), W1(C), R3(C), W2(A), R3(B), W3(A)

2

(i) Precedence Graph:



T1 and T3 also both conflict on variable A

2

(ii) Since the precedence graph has no direct cycles it *is* conflict-serializable. The corresponding serial schedule is R1(A), R1(B), W1(A), R1(C), W1(C), R2(A), W2(A), R3(C), R3(B), W3(A)

4

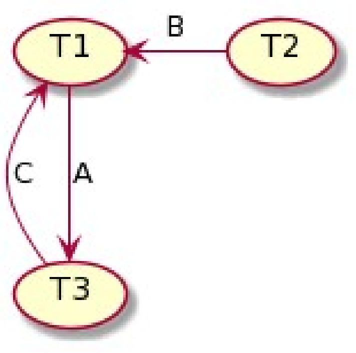
(iii) Yes. This schedule can occur by use of two-phase locking. (We add the notation LJ(X) and UJ(X) to the suggested RJ(X) and WJ(X), to mean that transaction TJ respectively locks and unlocks data item X.) L1(A), L1(B), L1(C), R1(A), R1(B), U1(B), W1(A), U1(A), L2(A), R2(A), R1(C), W1(C), U1(C), L3(C), R3(C), W2(A), U2(A), L3(B), R3(B), L3(A), W3(A), U3(A), U3(B), U3(C)

LJ(X) represents a write (exclusive) lock or a read (shared) lock?

2. R1(A), W1(A), R2(C), R2(B), R3(C), W3(C), R3(A), R1(B), W1(B), R1(C)

2

(i) Precedence Graph:



T2 and T3 also both conflict on variable C.

2

(ii) Since the precedence graph has a direct cycle it is *not* conflict-serializable. T1 writes A before T3 reads it, but T3 writes C before T1 reads it.

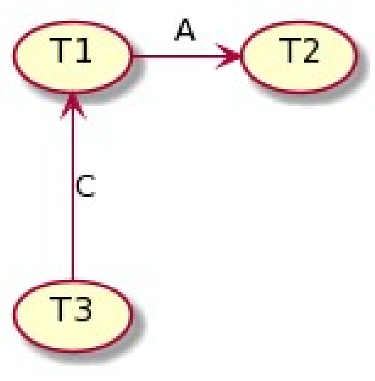
5

(iii) No. This schedule can not occur by use of 2PL. Proof by contradiction. If Schedule 2 could be achieved by 2PL it would be conflict-serializable. Since it is not conflict-serializable it cannot be achieved by 2PL.

3. R1(A), R1(B), W1(A), R2(A), W3(C), W1(C), W2(A)

3

(i) Precedence Graph:



2

(ii) Since the precedence graph has no direct cycles it is conflict-serializable. The corresponding serial schedule is W3(C), R1(A), R1(B), W1(A), W1(C), R2(A), W2(A)

5

(iii) No. This schedule can not occur by use of 2PL. This is because T1's growing phase cannot end until it has locked C, which it cannot do until after T3 has written to it as step 5, but it's shrinking phase must begin before it unlocks A, which must happen before T2 reads A at step 4. 2PL requires that the growing phase ends before the shrinking phase begins, which would mean step 5 needing to be before step 4, which it is not. ✓

$\boxed{10/10}$

Question 3

1. After time step 16, the database records for A and B are 2 and 6, and the product (local to Transaction 2) is 21. See Table.

$\boxed{8}$

Time	**Database(A)**	**Database(B)**	T1(A)	T1(B)	T2(A)	T2(B)	**T2(product)**
0	**3**	**5**	n/a	n/a	n/a	n/a	**n/a**
1	**3**	**5**	3	n/a	n/a	n/a	**n/a**
2	**3**	**5**	1	n/a	n/a	n/a	**n/a**
3	**3**	**5**	1	n/a	n/a	n/a	**1**
4	**3**	**5**	1	n/a	3	n/a	**1**
5	**1**	**5**	1	n/a	3	n/a	**1**
6	**1**	**5**	1	n/a	3	n/a	**3**
7	**1**	**5**	1	n/a	2	n/a	**3**
8	**1**	**5**	1	5	2	n/a	**3**
9	**1**	**5**	1	6	2	n/a	**3**
10	**6**	**6**	1	6	2	n/a	**3**
11	**6**	**6**	1	6	2	n/a	**3**
12	**2**	**6**	1	6	2	n/a	**3**
13	**2**	**6**	1	6	2	6	**3**
14	**2**	**6**	1	6	2	7	**3**
15	**2**	**6**	1	6	2	7	**21**
16	**2**	**6**	1	6	2	7	**21**

$\boxed{1}$

2. If we first execute T1, then T2, we get A=0, B=6, and product=7.

$\boxed{1}$

3. If we first execture T2, then T1, we get A=0, B=6, and product=18.