CSE 344 Section 7 Worksheet

Solutions

(a) Consider a concurrency control manager that uses strict two phase locking that schedules three transactions:

- *T1 : R1(A), R1(B), W1(A), W1(B), Co1*
- *T2 : R2(B), W2(B), R2(C), W2(C), Co2*
- *T3 : R3(C), W3(C), R3(A), W3(A), Co3*

Each transaction begins with its first read operation and commits with the *Co* statement. Answer the following questions for each of the schedules below:

- Is this schedule possible under a strict 2PL protocol?
- If strict 2PL does not allow this schedule because it denies a read or write request, is the system in a deadlock at the time when the request is denied?

Note:
A common question is if conflict serializability implies the schedule is possible under a strict 2PL protocol, and the answer is Conflict Serializable -/-> strict 2PL but strict 2PL → Conflict Serializable

i. Schedule 1:

R2(B), W2(B), R3(C), W3(C), R3(A), W3(A), Co3, R2(C), W2(C), Co2, R1(A), R1(B), W1(A), W1(B), Co1

a) Is the schedule conflict-serializable? If yes, indicate a serialization order.

Yes: 3,2,1

b) Is it possible under strict 2PL?

Yes

c) Does strict 2PL lead to a deadlock?

No

ii. Schedule 2:

R2(B), W2(B), R3(C), W3(C), R1(A), R1(B), W1(A), W1(B), Co1, R2(C), W2(C), Co2, R3(A), W3(A), Co3

a) Is the schedule conflict-serializable? If yes, indicate a serialization order.

No

a) Is it possible under strict 2PL?

No: T2 holds L(B) thus R1(B) is not possible before Co2.

b) Does strict 2PL lead to a deadlock?

Yes: T1 holds L(A), T2 holds L(B), T3 holds L(C) and none can make progress.

(b) Given the following three transactions:

T1: R(A), W(B), I(D), R(C)
T2: R(B), R(D), W(C)
T3: R(D), R(C), R(D), W(A)

Assume that R(X) reads all tuples in table X, W(X) updates all tuples in X, and I(X) inserts one new tuple in X.

Does there exist a schedule of the above transactions that would result in a deadlock if executed under strict 2PL with **only exclusive table locks**? If so, write such a schedule with lock and unlock operations and indicate why the transactions are deadlocked. Otherwise write "No". Use L1(A) to refer to T1 locking table A, and U1(A) for unlocking.

`L1(A); R1(A); L2(B); R2(B); L3(D); R3(D); L3(C); R3(C);` <deadlock>
T1 holds lock on A and is waiting for lock on B, which is held by T2
T2 holds lock on B and is waiting for lock on D, which is held by T3
T3 holds lock on D and is waiting for lock on A, which is held by T1

To get full points, students will need to show both a schedule, explain which transaction holds which lock, and how that leads to a deadlock.

(c) What indexes could we make on Users?

CREATE TABLE Users (
id INT PRIMARY KEY,
age INT,
score INT, ...);

Expecting 1000 exec/day
SELECT *
FROM Users, Assets
WHERE Users.id = Assets.uid

Expecting 1000 exec/day

SELECT *
FROM Users
WHERE Users.score > 95

Expecting 10 exec/day
SELECT *
FROM Users
WHERE Users.age > 21

Sol:

- **IDs are unique so create an unclustered index on ID.**
- **Create a covering index primarily keyed on score.**
- **Create a covering index primarily keyed on age.**