Nicholas Donahue

CS 348

Homework 3

1. Nesting & Un-nesting
   1. SELECT ename, C.child, Address.city as addr\_city, Address.state as addr\_state, Address.zipcode as addr\_zip, S.skill

FROM emp as E, unnest(E.childrenSet) as C (child), unnest(E.skillsSet) as S (skill);

* 1. SELECT ename, collect(child) as childrenSet, Address (addr\_city, addr\_state, addr\_zip), skill

FROM flat-emp

GROUP BY ename, skill, Address;

* 1. SELECT ename, child, Address (addr\_city, addr\_state, addr\_zip) as Address, collect(skill) as skillsSet

FROM flat-emp

GROUP BY ename, child, Address;

* 1. SELECT ename, collect(child) as childrenSet, Address (addr\_city, addr\_state, addr\_zip) as Address, collect(skill) as skillsSet

FROM flat-emp

GROUP BY ename, Address;



S1 =R1(X); R2(Y); W1(Y); W2(Z); R3(Z); W3(X)

2. **Yes**, it is serializable.
3. The serial schedule is: 2 – 1 – 3

S2 =R1(A); R3(B); W1(A); R2(C); R2(A); W3(B); W2(A); R3(A); R1(B); W2(C); W1(B)

2. **No**, it is not serializable.
3. This is not serializable because there are multiple conflicts that lead to a cycle in the precedence graph.
5. There is a **dirty read problem** in this schedule. In T1 the value of A is initially read, modified (+100) and written. In T2 immediately after, A is also read modified (-50) and then written. However, since there is a failure and rollback at the end of T1 it is never commit to the database whilst T2 updated and used the now invalidated value in variable A.
6. There is a **lost update problem** in this schedule. In T1 A is read initially and then there is 100 added to it, however the write statement is not until T=5. In T=3 and T=4, there is a read and update in T2 that changes the original value of A to A – 50. This causes the modifications of T2 to be saved to the database, overwriting the changes T1 wanted to do initially.

i.

1. 2PL: Yes, this is allowed.

Strict 2PL: No, there is no proper unlocking of Y before the end.

1. 2PL: Yes, this is allowed.

Strict 2PL: Yes, there is proper unlocking of transaction and timestamps.

1. 2PL: Yes, this is allowed.

Strict 2PL: Yes, there is proper unlocking of transaction and timestamps.

ii.

1. 2PL: Yes to all.

Strict 2PL: Yes to all. Strict locking ensures we obtain locks before writing. The hold and no shares provide recoverability and avoidance of rollback.

1. 2PL: Yes to serializability, no to recoverability and avoidance

Strict 2PL: Yes to serializability, no to recoverability and avoidance

1. 2PL: Yes to all.

Strict 2PL: Yes to all.