Xiang Xu 3/10/2017
605.741.31
Homework for Module 7

---

1. Concurrency Control.

a) Define *conflict equivalence*.

Two schedules are conflict equivalent if they have same operations and every pair of conflicting operations is ordered the same way, and at the end schedules leave the same effect on the database.

b) Define *conflict serializable.*

A schedule is conflict serializable if and only if it is conflict equivalent to a serial schedule.

c) Describe how these properties form the basis of all the concurrency control algorithms that we studied and relate them to the appropriate ACID properties.

Two Phase Locking, Distributed Two Phase Locking, Timestamp Ordering and Multi-version Timestamp Ordering, they are different implementation to ensure database follows ACID properties.
All four algorithms ensured Atomicity and Durability, once we do a roll back, everything will back to the original point. Once we commit, it will be the permanent change.
They use different approaches to handle Consistency and Isolation. Locking Algorithms use locks to ensure the serializability of the transactions, in the other hand, timestamp algorithms use timestamps to ensure the serializability of the transactions. Locking Algorithms we will have to deal with the dead lock problems, and the multiversion timestamp algorithm has an additional overhead of version maintenance.

2. Given the following two transactions:

T_1:	$R_1(x)$	T_2:	$R_2(x)$
	$R_1(y)$		$R_2(z)$
	$x = x+y$		$x = x*z$
	$W_1(x)$		$W_2(x)$

a) Write two serial schedules for the two transactions above. Label the first S_a and the second S_b.

S_a = { R_1(x) , R_1(y), W_1(x) , R_2(x), R_2(z), W_2(x) }
S_b = { R_2(x), R_2(z), W_2(x) , R_1(x) , R_1(y), W_1(x) }

b) **For each of the following schedules S_1, S_2, and S_3, indicate if it is conflict equivalent to S_a, S_b, or neither S_a nor S_b. Explain your answers.**

S_1 = { $R_2(z)$, $R_1(y)$, $R_2(x)$, $W_2(x)$, $R_1(x)$, $W_1(x)$ }
S_2 = { $R_1(x)$, $R_2(x)$, $R_1(y)$, $R_2(z)$, $W_1(x)$, $W_2(x)$ }
S_3 = { $R_1(x)$, $R_1(y)$, $R_2(z)$, $W_1(x)$, $R_2(x)$, $W_2(x)$ }

Note: for S_a, S_b, S_1, S_2, S_3, we will not care about the order of operations on y and operations on z, beacuase We only have one operation for y and one operation for z. so order does not matter here.

S_a:
Ordering of Operations on x: R_1(x) , W_1(x) , R_2(x), W_2(x)

S_b:
Ordering of Operation on x: R_2(x), W_2(x) , R_1(x) , W_1(x)

S_1
Ordering of Operations on x: $R_2(x)$, $W_2(x)$, $R_1(x)$, $W_1(x)$
S_1 is conflict equivalent to S_b, because their orders of conflict operations are the same, so they leave the same effect on the database.

S_2
Ordering of Operations on x: $R_1(x)$, $R_2(x)$, $W_1(x)$, $W_2(x)$
S_2 is neither conflict equivalent to S_a nor S_b, because their orders of conflict operations are the different, so they might NOT be leave the same effect on the database.

S_3
Ordering of Operations on x: $R_1(x)$, $W_1(x)$, $R_2(x)$, $W_2(x)$
S_1 is conflict equivalent to S_a, because their orders of conflict operations are the same, so they leave the same effect on the database