15.05.2015

Deadlock

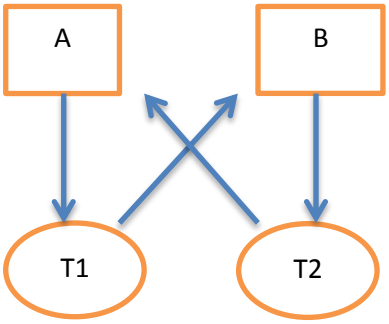
A&B= accounts

T1&T2=transactions

A is assigned to T1.

Lock A so that no one else can access it. exclusive lock

T1 needs to B proceed!



Loop= deadlock

Deadlock detection

Assume that we detect a deadlock!

Kill both transactions. abort

Undo all actions of both T1 and T2.

For undoing use the system log file which contains all actions of all transactions.

The second type of lock is sharable lock.

A resource locked with a sharable lock can be read by more than one xact at the same, but no xact can change it.

Deadlock Prevention

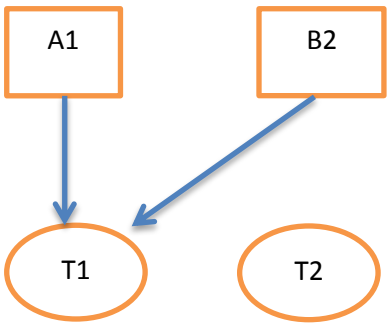
How to achieve deadlock prevention?

Assign everything needed by a transaction from the very beginning.

Problem: How to know everything?

Assign numbers to resources in ascending order.

A transaction can only get resources in ascending order.



When do we have problems during xact processing?

RR	/	WW	RW	WR
No Problem			Problem!	

Conflict Equivalent (for two schedules)

Read the notes (slide)!

S1

T_1 …
T_2 …

Conflicts are in the same order.

$R_1(A)\ W_2(A)\ \ldots\ W_1(A)\ W_2(A)$

S2

T_1 …
T_2 …

$R_1(A)\ W_2(A)\ \ldots\ W_1(A)\ W_2(A)$

Conflict Serializable (only for one schedule)

Conflict equivalent to some serial schedule.

Every serializable schedule is serializable.

Example: Schedule

C-Commit

T_1 …	$W_1(A)\ R_1(B)\ C$
T_2 …	$W_2(A)\ R_2(A)\ C$

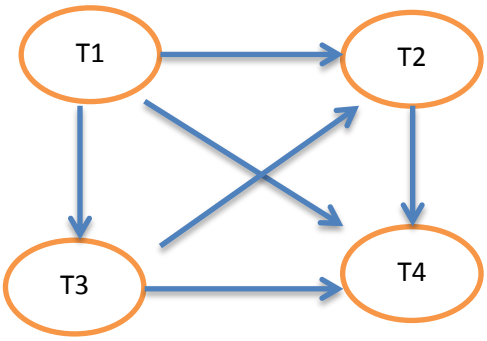
$W_1(A)\ R_1(B)\ C_1\ W_2(A)\ R_2(A)\ C_2$

Consider a schedule as

S: $W_1(X)\ R_3(X)\ W_2(X)\ W_4(X)\ R_4(Y)\ R_1(Y)$

$W_1(X)R_3(X)$, $W_1(X)W_2(X)$, $W_1(X)W_4(X)$, $R_3(X)W_2(X)$, $R_3(X)W_4(X)$, $W_2(X)W_4(X)$

Precedence Graph (Serializability Graph)



Check the graph and see if there is a cycle.

If no cycle it means that it is conflict serializable.

Topological sort T1, T3, T2, T4