IT 775
Database Technology

SQL-DML

Transaction Management

Transaction Support

<u>Transaction</u>

Action, or series of actions, carried out by user or application, which reads or updates contents of database.

- Logical unit of work on the database.
- Application program is series of transactions with non-database processing in between.
- Transforms database from one consistent state to another, although consistency may be violated during transaction.

Transaction Support

Can have one of two outcomes:

Success - transaction *commits* and database reaches a new consistent state.

Failure - transaction *aborts*, and database must be restored to consistent state before it started.

Such a transaction is *rolled back* or *undone*.

Committed transaction cannot be aborted.

Aborted transaction that is rolled back can be restarted later.

Lost Update Problem

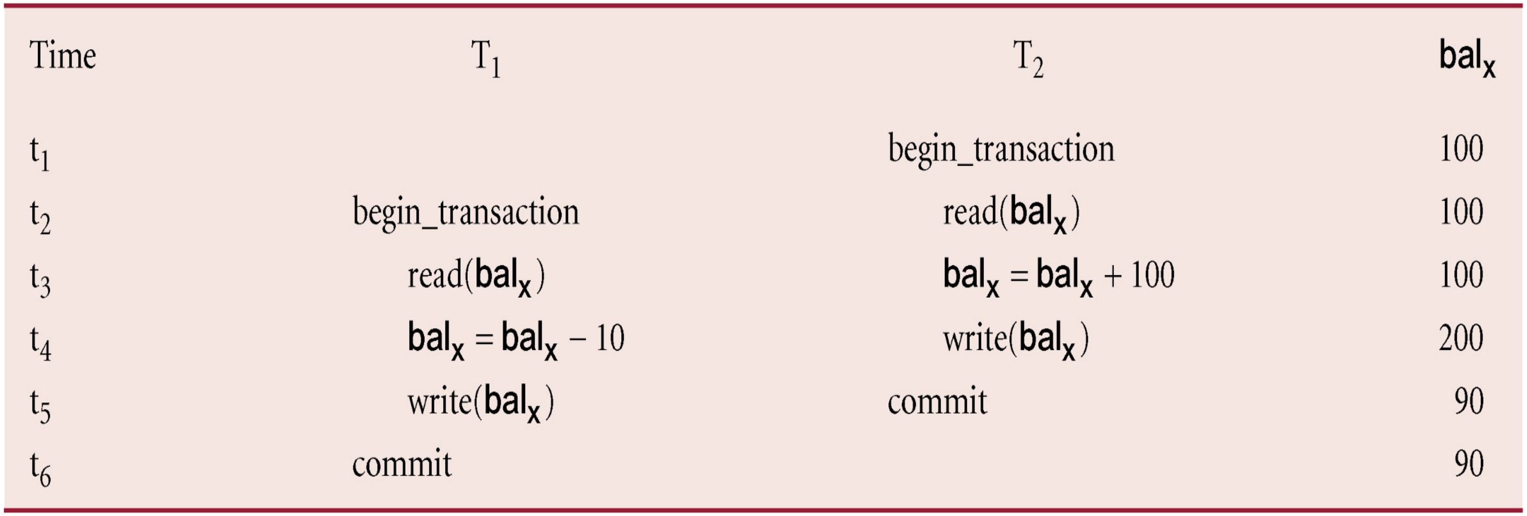
Successfully completed update is overridden by another user.

T_1 withdrawing \$10 from an account with bal_x, initially \$100.

T_2 depositing \$100 into same account.

Serially, final balance would be \$190.

Lost Update Problem



Time	T_1	T_2	bal_x
t_1		begin_transaction	100
t_2	begin_transaction	read(bal_x)	100
t_3	read(bal_x)	$bal_x = bal_x + 100$	100
t_4	$bal_x = bal_x - 10$	write(bal_x)	200
t_5	write(bal_x)	commit	90
t_6	commit		90

Loss of T_2's update avoided by preventing T_1 from reading bal_x until after update.

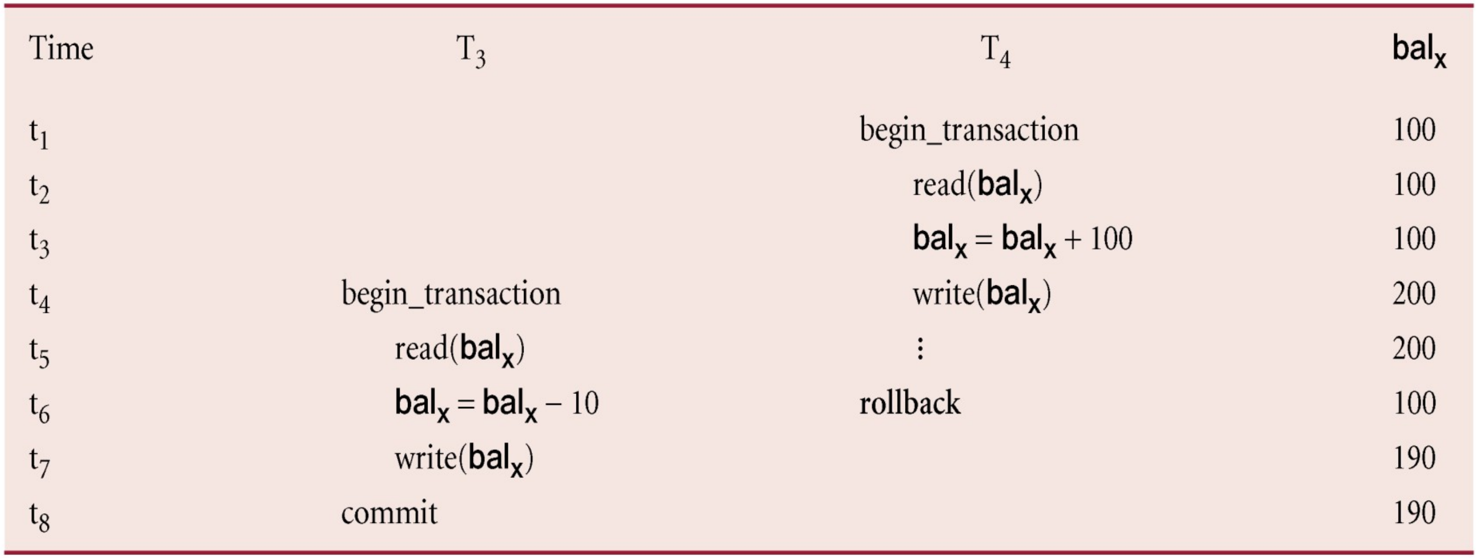
Uncommitted Dependency Problem

Occurs when one transaction can see intermediate results of another transaction before it has committed.

T_4 updates bal_x to \$200 but it aborts, so bal_x should be back at original value of \$100.

T_3 has read new value of bal_x (\$200) and uses value as basis of \$10 reduction, giving a new balance of \$190, instead of \$90.

Uncommitted Dependency Problem



Time	T_3	T_4	bal_x
t_1		begin_transaction	100
t_2		read(bal_x)	100
t_3		$\text{bal}_x = \text{bal}_x + 100$	100
t_4	begin_transaction	write(bal_x)	200
t_5	read(bal_x)	⋮	200
t_6	$\text{bal}_x = \text{bal}_x - 10$	rollback	100
t_7	write(bal_x)		190
t_8	commit		190

Problem avoided by preventing T_3 from reading bal_x until after T_4 commits or aborts.

Inconsistent Analysis Problem

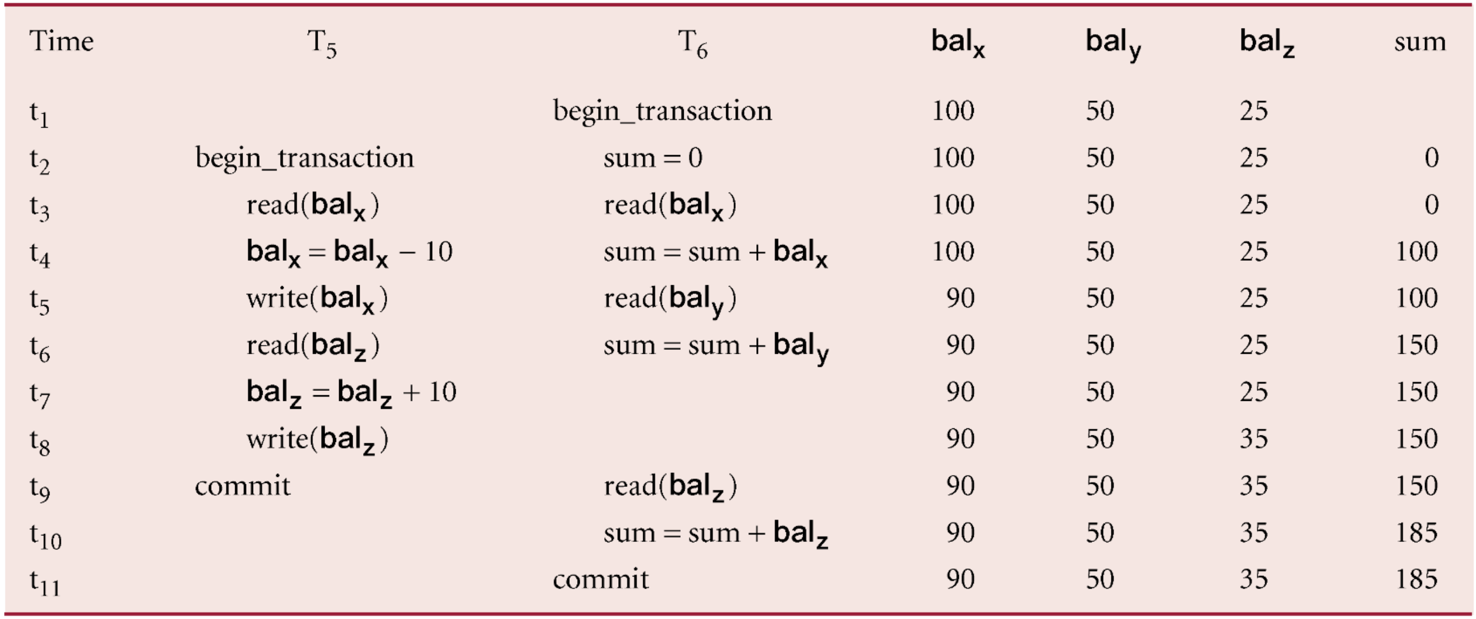
Occurs when transaction reads several values but second transaction updates some of them during execution of first.

Sometimes referred to as *dirty read* or *unrepeatable read*.

T_6 is totaling balances of account x ($100), account y ($50), and account z ($25).

Meantime, T_5 has transferred $10 from bal_x to bal_z, so T_6 now has wrong result ($10 too high).

Inconsistent Analysis Problem



Time	T_5	T_6	bal_x	bal_y	bal_z	sum
t_1		begin_transaction	100	50	25	
t_2	begin_transaction	sum = 0	100	50	25	0
t_3	read(bal_x)	read(bal_x)	100	50	25	0
t_4	$\text{bal}_x = \text{bal}_x - 10$	sum = sum + bal_x	100	50	25	100
t_5	write(bal_x)	read(bal_y)	90	50	25	100
t_6	read(bal_z)	sum = sum + bal_y	90	50	25	150
t_7	$\text{bal}_z = \text{bal}_z + 10$		90	50	25	150
t_8	write(bal_z)		90	50	35	150
t_9	commit	read(bal_z)	90	50	35	150
t_{10}		sum = sum + bal_z	90	50	35	185
t_{11}		commit	90	50	35	185

Problem avoided by preventing T_6 from reading bal_x and bal_z until after T_5 completed updates.