Transaction Management and Concurrency Control (Refresher)

Transactions

- A user's program may carry out many operations on the data retrieved from the database, but the DBMS is only concerned about what data is read/written from/to the database.
- A *transaction* is the DBMS's abstract view of a user program: a sequence of reads and writes.

T1: R(A); A=A+100; W(A); R(B); B=B-100; W(B); Commit

The ACID properties

- **A** tomicity: All actions in the Xact happen, or none happen.
- **C** onsistency: If each Xact is consistent, and the DB starts consistent, it ends up consistent.
- **I** solation: Execution of one Xact is isolated from that of other Xacts.
- **D** urability: If a Xact commits, its effects persist.

Concurrency in a DBMS

- Users submit transactions, and can think of each transaction as executing by itself.
- Concurrency is achieved by the DBMS, which interleaves actions (reads/writes of DB objects) of various transactions.
- Each transaction must leave the database in a consistent state if the DB is consistent when the transaction begins.

Example

- Consider two transactions (*Xacts*):

```
T1: A=A+100, B=B-100
T2: A=1.06*A,   B=1.06*B
```

- Intuitively, the first transaction is transferring $100 from B's account to A's account. The second is crediting both accounts with a 6% interest payment.
- There is no guarantee that T1 will execute before T2 or vice-versa, if both are submitted together. However, the net effect *must* be equivalent to these two transactions running serially in some order.

```
T1: A=A+100, B=B-100
T2:                          A=1.06*A,   B=1.06*B
```

```
T1:                             A=A+100, B=B-100
T2: A=1.06*A,   B=1.06*B
```


Example (Contd.)

- Consider a possible interleaving (*schedule*):

```
T1:     A=A+100,                    B=B-100
T2:                 A=1.06*A,                   B=1.06*B
```

- This is OK. But what about:

```
T1:     A=A+100,                               B=B-100
T2:                 A=1.06*A, B=1.06*B
```

- The DBMS's view of the second schedule:

```
T1:     R(A), W(A),                                     R(B), W(B)
T2:                     R(A), W(A), R(B), W(B)
```

Scheduling Transactions

- *Serial schedule:* Schedule that does not interleave the actions of different transactions.
- *Equivalent schedules:* For any database state, the effect (on the set of objects in the database) of executing the first schedule is identical to the effect of executing the second schedule.
- *Serializable schedule*: A schedule that is equivalent to some serial execution of the transactions.

(Note: If each transaction preserves consistency, every serializable schedule preserves consistency.)

Lock-Based Concurrency Control

- Each Xact must obtain a S (*shared*) lock on object before reading, and an X (*exclusive*) lock on object before writing.
- If an Xact holds an X lock on an object, no other Xact can get a lock (S or X) on that object.

T1: S(A), R(A), unlock(A)
T2: X(A), R(A), W(A), unlock(A)

Two-Phase Locking (2PL)

- Each Xact must obtain a S (*shared*) lock on object before reading, and an X (*exclusive*) lock on object before writing.
- A transaction can not request additional locks once it releases any locks.
- If an Xact holds an X lock on an object, no other Xact can get a lock (S or X) on that object.

Strict 2PL

- Each Xact must obtain a S (*shared*) lock on object before reading, and an X (*exclusive*) lock on object before writing.
- All locks held by a transaction are released when the transaction completes
- If an Xact holds an X lock on an object, no other Xact can get a lock (S or X) on that object.

Strict 2PL allows ***only serializable*** schedules

Deadlocks

- Deadlock: Cycle of transactions waiting for locks to be released by each other.

Deadlock Detection

- Create a waits-for graph:
  - Nodes are transactions
  - There is an edge from Ti to Tj if Ti is waiting for Tj to release a lock
- Periodically check for cycles in the waits-for graph

Deadlock Detection (Continued)

Example:

T1:	S(A), R(A),		S(B)				
T2:		X(B),W(B)			X(C)		
T3:				S(C), R(C)			X(A)
T4:						X(B)