Reflection 4

Concurrency Control and Database Recovery Techniques

Topics to Covered

- Introduction to Concurrency Control
- Deadlocks and Starvation
- Concurrency Control Protocols
  - Locking Protocols
  - Time Stamp Ordering Protocol

Concurrency Control

- Concurrency Control insures:
  - Serializability
  - Strict Recoverability

- Concurrency Control Protocols:
  - Locking Protocols
  - Time Stamp Ordering Protocols

Deadlock

In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

Example: A set of transactions {T_0, T_1, T_2, ...,T_n}.

T_0 needs a resource X to complete its task. Resource X is held by T_1, and T_1 is waiting for a resource Y, which is held by T_2. T_2 is waiting for resource Z, which is held by T_0.

- Time stamp ordering is used to prevents deadlocks. Two deadlock prevention protocols are:
  1. Wait-Die Protocol
  2. Wound-Wait Protocol

Starvation

If a transection or process wait for a resource forever or infinity time than this scenario is called starvation.

Example: A transaction T_2 has a shared lock on a data item, and another transaction T_1 requests an exclusive lock on the data item. Clearly, T_1 has to wait for T_2 to release the share lock. Meanwhile a transaction T_3 may request a shared lock on the same data item. The lock request is compatible with the lock granted to T_2 so T_3 may be granted the shared lock. At this point T_2 may release the lock, but still T_1 has to wait for T_3 to finish. But again there may be a new transaction T_4 that request a shared lock on the same data item and is granted the lock before T_3 releases it. In fact, it is possible that there is a sequence of transactions that each request a shared lock on the data item and each transaction release the lock a short while after it is granted, but T_1 never gets the exclusive lock on the data item. The transaction T_1 may never make progress and is said to be starved.

Locking Protocols

<u>Locks:</u> Locks are the variable used to identify the status of Data item.

Example:

Transection T

Lock(A) ← Grants Lock

Read(A)

Write(A)

Release(A) ← Release Lock

Types of Locks

- Shared Lock: Read only lock

  Example:

  Transection T
  Shared Lock(A) ⟵ Grants Lock
  Read(A)
  Write(A) ⟵ Not Allowed

- Exclusive Lock: Read/Write lock

  Example:

  Transection T
  Exclusive Lock(A) ⟵ Grants Lock
  Read(A)
  Write(A)

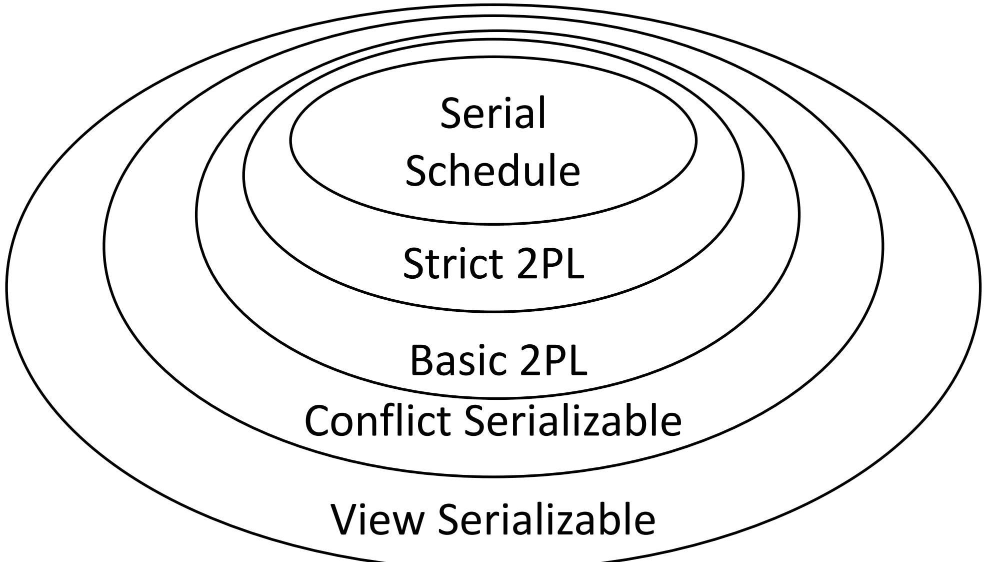
2 Phase Locking Protocol (2PL)

- Ensure Serializability.
- Equivalent Serial Schedule is based upon Lock Points.
- If any schedule is not conflict serializable schedule, it will also not allowed by 2PL.

Disadvantage of 2PL

- 2PL not free from Deadlock.
- 2PL not free from Starvation.
- 2PL not free from irrcoverbility, cascading rollback and last update problem (Strict 2PL is suggested to solve this problem).

  Strict 2PL: All exclusive locks of transection T should be hold until commit or rollback of T.



Time Stamp Ordering Protocol(TSOP)

- Time stamp ordering should be same as conflict pair precedence.

  Example:

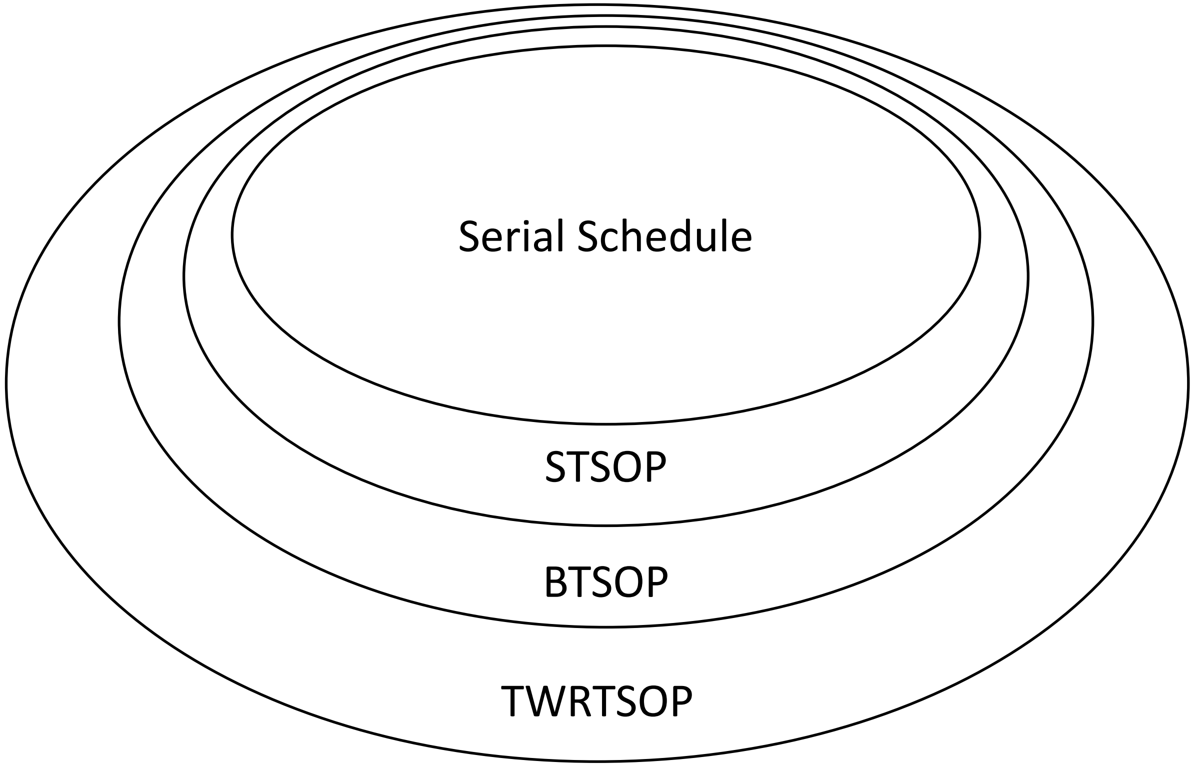
  T_i (10) $\longrightarrow$ T_j (20) Allowed

  T_i (10) $\longleftarrow$ T_j (20) Not Allowed

- If any schedule is not conflict serializable than it will also not allowed by Time stamp ordering protocol.
- Time stamp protocol is divided in three categories.
  - Basic Time Stamp Ordering Protocol(BTSOP)
  - Thomas Write Time Stamp Ordering Protocol(TWRTSOP)
  - Strict Time Stamp Ordering Protocol(STSOP)
- Dead lock is not possible in Time Stamp Ordering Protocol.

Disadvantage of Time Stamp Ordering Protocol

- Time Stamp Ordering Protocol is not free from starvation.
- Very less degree of concurrency.



Database Recovery

- To bring the database into the last consistent state, which existed prior to the failure.
- To preserve transaction properties (Atomicity, Consistency, Isolation and Durability).
- Transection logs are used to recover the database.
- As soon as the data is modified by any transection, it is reflected back to disk. There four types of data update are present.
    1. Immediate Update
    2. Deferred Update
    3. Shadow update
    4. In-place update

Write-Ahead Logging(WAL)

- For Undo Operations: Before a data item's AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
- For Redo Operations: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.

  Where:

  BFIM (BeFore Image) -> data values prior to modification

  AFIM (AFter Image) -> data values after modification

Flushing database cache to database disk and Recovery Handling

1. Steal: Cache can be flushed before transaction commits.
2. No-Steal: Cache cannot be flushed before transaction commit.
3. Force: Cache is immediately flushed (forced) to disk.
4. No-Force: Cache is deferred until transaction commits

Based on the above flushing techniques four way of Recovery Handling is possible:

1. Steal/No-Force (Undo/Redo)
2. Steal/Force (Undo/No-redo)
3. No-Steal/No-Force (Redo/No-undo)
4. No-Steal/Force (No-undo/No-redo)

Deferred Update (No Undo/Redo)

- A set of transactions records their updates in the log.
- At commit point under WAL scheme these updates are saved on database disk.
- After reboot from a failure the log is used to redo all the transactions affected by this failure. No undo is required because no AFIM is flushed to the disk before a transaction commits.

Immediate Update (Undo/No-redo Algorithm)

- In this algorithm AFIMs of a transaction are flushed to the database disk under WAL before it commits.
- For this reason the recovery manager undoes all transactions during recovery.
- No transaction is redone.
- It is possible that a transaction might have completed execution and ready to commit but this transaction is also undone.

Thank You