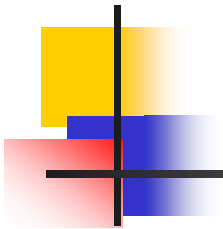
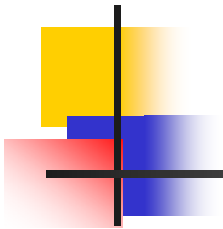
Concurrency Control and Recovery



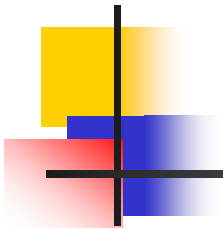
Objectives

- Introduction to Concurrency Control +
- Concurrency Control Techniques +
- Introduction to Recovery +



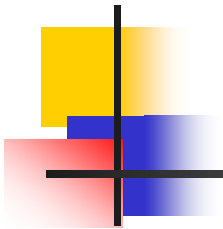
- Introduction to Concurrency Control ...

- The objective of a concurrency control is to schedule transactions in such a way as to avoid any interference.
- We could run transactions serially, but this limits degree of concurrency or parallelism in system.
- A schedule is a sequence of reads/writes by set of concurrent transactions.
- A serial schedule is a schedule where operations of each transaction are executed consecutively without any interleaved operations from other transactions.
- A nonserial schedule where operations from set of concurrent transactions are interleaved.



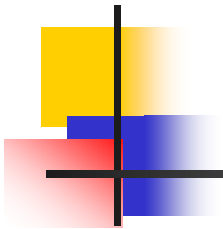
... - Introduction to Concurrency Control ...

- The objective of serializability is to find nonserial schedule that allow transactions to execute concurrently without interfering with one another.
- In other words, we want to find nonserial schedules that are equivalent to some serial schedule. Such schedule are called serializable.
- In serializability, ordering of read/write is important:
  - If two transactions only read a data item, they do not conflict and their order is not important.
  - If two transactions either read or write completely separate data items, they do not conflict and their order is not important.
  - If one transaction writes a data item and another one reads or writes the same data item, then a conflict occurs and the order of execution becomes important.
  - To fulfill the srializability requirements the schedule should be a conflict free. Therefore, conflicts must be resolved as will be explained later.



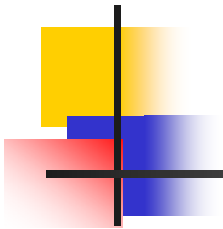
- Concurrency Control Techniques ...

- The two major techniques for concurrency control are:
  - Locking +
  - Time stamping +



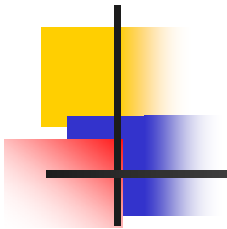
-- Locking Technique ...

- This is the most widely used technique to ensure serializability.
- The technique is based on using locks on data items by transactions.
- A **lock** is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied on it.
- Generally, there is one lock for each data item in the DB.
- A transaction must claim a lock on a data item before read or write.



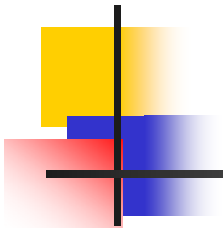
... -- Locking Technique

- There are two types of locks:
  - Binary
  - Shared/Exclusive
- A binary lock can have two states: locked, unlocked.
- A shared/exclusive lock can have multiple states: read_locked, write_locked and unlocked.
- With binary locks, a data item X can be used by only one transaction at a time, whether the transaction reads or writes X.
- With shared/exclusive locks, a data item X can be used by more than one transaction at a time, if all of them read X, and only one transaction if it writes X.



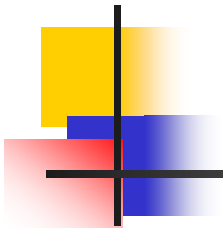
-- Timestamp Technique

- A timestamp (t) is a unique identifier created by DBMS that indicates relative starting time of transaction T. A timestamp can be generated by using a system clock, or by incrementing a logical counter every time a new transaction starts.
- In this technique transactions are ordered globally so that older transactions Those with the smaller timestamp) get priority in the event of conflict.
- A conflict is resolved by rolling back the new transactions and restarting it later.
- A read/write operation performed by transaction T_1 proceeds only if the last update on that data item was carried out by an older transaction T_2. Otherwise, T_1 which is requesting read/Write is restarted and give a new timestamp.



- Introduction to Recovery ...

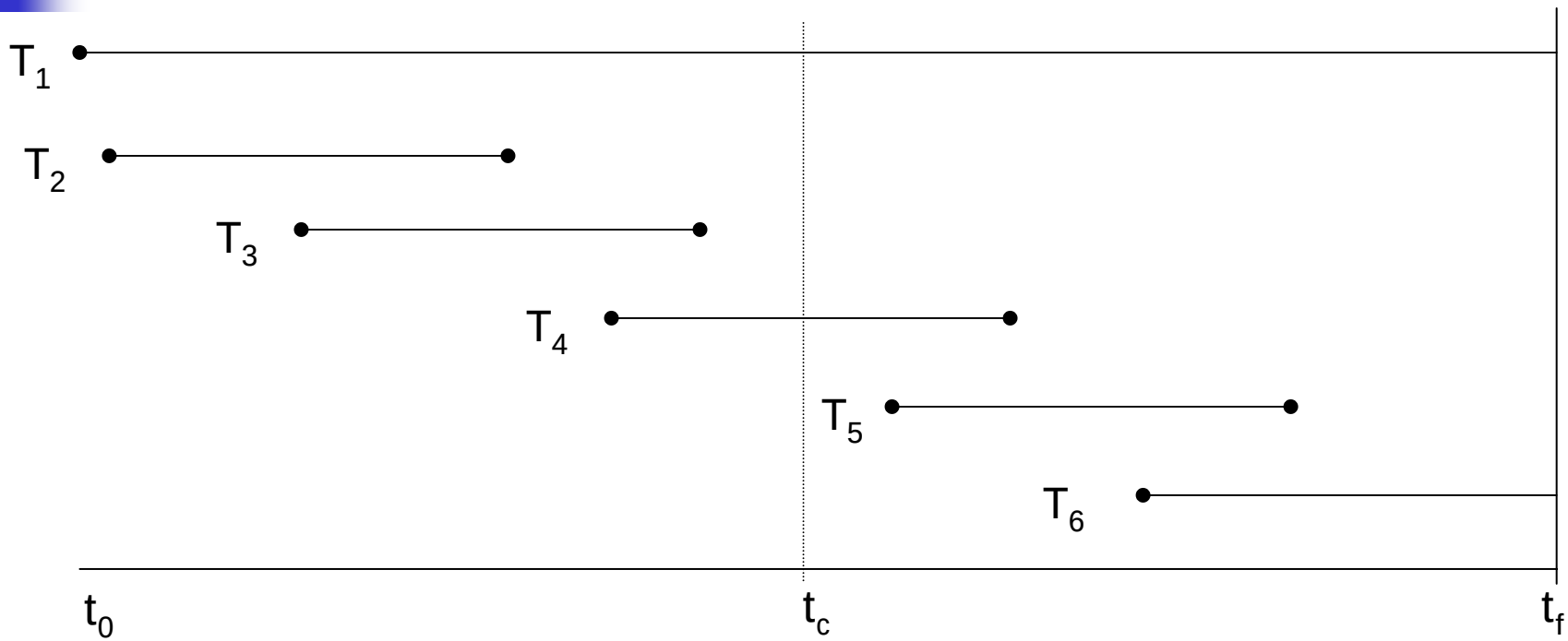
- Types of failures:
  1. System crash (hardware, software, network) results in loss of main memory contents.
  2. Transaction/System Errors, e.g. integer overflow, division by zero, erroneous parameter value or logical error.
  3. Local errors or exception conditions detected by the transaction, e.g. data not found, or invalid action.
  4. Concurrency control enforcement: abort transaction due to serialization problem.
  5. Disk failure
  6. Physical problems and disasters



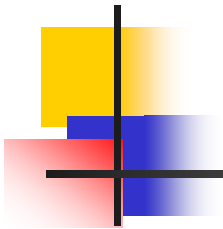
- Transactions and Recovery

- Transaction represent basic units of recovery.
- Recovery manager responsible for atomicity and durability.
- If failure occurs between commit and DB buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction's updates.
- If transaction had not committed at a failure, recovery manager has to undo (rollback) any effects of that transaction for atomicity.
- Partial undo – only one transaction has to be undone.
- Global undo – all transactions have to be undone.

-- Example

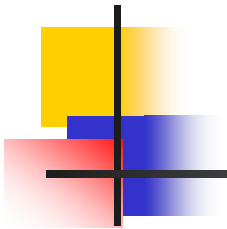


- DBMS started at time t_0 but failed at time t_f. T_c is a check point.
- Transactions T_2 and T_3 have recorded their updates to secondary storage.
- T_1 and T_6 have to be undone and restarted later.
- In absence of any other information, recovery manager has to redo T_4 and T_5



- Recovery Facilities ...

- DBMS should provide the following facilities to assist with recovery:
  - Backup mechanism, which makes periodic backup copies of DB.
  - Logging facilities, which keeps track of current state of transactions and DB changes.
  - Checkpoint facility, which enables updates to DB in progress to be made permanent.
  - Recovery manager, which allows DBMS to restore the DB to a consistent state following failure.



... - Recovery Facilities: Log File ...

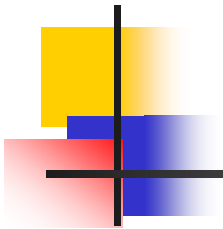
- Contains information about all updates to DB:
  - Transaction records
  - Checkpoint records
- Often used for other purposes (for example, auditing).
- Transaction records contain:
  - Transaction identifier
  - Type of log record, (transaction start, insert, update, delete, abort, commit).
  - Identifier of data item affected by DB action (insert, delete, and update operations).
  - Before-image of data item.
  - After-image of data item
  - Log management information
- Log file may be duplexed or triplexed.
- Log file sometimes split into two separate random-access files.
- Potential bottleneck; critical in determining overall performance

-- Sample Log File

Tid	Time	Operation	Object	B-image	A-Image	PPtr	NPtr
T1	10:12	START				0	2
T1	10:13	UPDATE	STAFF SL21	(old value)	(new value)	1	8
T2	10:14	START				0	4
T2	10:16	INSERT	STAFF SG37		(new value)	3	5
T2	10:17	DELETE	STAFF SA9	(old value)		4	6
T2	10:17	UPDATE	PROPERTY PG16	(old value)	(new value)	5	9
T3	10:18	START				0	11
T1	10:18	COMMIT				2	0
	10:19	CHECKPOINT	T2, T3				
T2	10:19	COMMIT				6	0
T3	10:20	INSERT	PROPERTY PG4		(new value)	7	12
T3	10:21	COMMIT				11	0

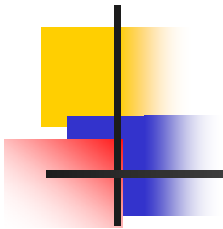
- Recovery Facilities: Checkpoint

- A checkpoint is a point of synchronization between DB and log file.
- Checkpoint record is created containing identifiers of all active transactions
- When a failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
- In previous example, with checkpoint at t_c changes made by T_2 and T_3 have been written to secondary storage. Thus:
  - Only redo T_4 and T_5
  - Undo transactions T_1 and T_6



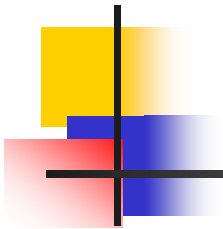
- Recovery techniques ...

- If the DB has been damaged:
  - Need to restore last backup copy of DB and reapply updates of committed transactions using log file.
- If the DB is only inconsistent:
  - Need to undo changes that caused inconsistency. May also need to redo some transactions to ensure updates reach secondary storage.
  - Do not need backup, but can restore using before and after images in the log file.



... - Recovery techniques ...

- Differed Update Technique:
  - Updates are not written to the DB until after a transaction has reached its commit point.
  - If a transaction fails before commit, it will not have modified DB and so no undoing of changes required.
  - May be necessary to redo updates of committed transactions as their effect may not have reached DB.



... - Recovery techniques

- Immediate Update Technique:
  - Updates are applied to DB as they occur.
  - No need to redo updates of committed transactions following a failure.
  - May need to undo effects of transactions that had not being committed at the time of failure.
  - Essential that log records are written before write to DB – write ahead log protocol.
  - If no “transaction commit” record in log, then that transaction was active at failure and must be undone.
  - Undo operations are performed in reverse order in which they were written to log.