DBMS UNIT IV.pptx

Janagi Raman S

TRANSACTION

Basic database access operations:

OPERATIONS	DESCRIPTIONS
Retrieve	To retrieve data stored in a database.
Insert	To store new data in database.
Delete	To delete existing data from database.
Update	To modify existing data in database.
Commit	To save the work done permanently.
Rollback	To undo the work done.

DBMS UNIT IV.pptx

Janagi Raman S

Transaction Execution with SQL

- Transaction support is provided by two SQL statements namely COMMIT and ROLLBACK.
- Transaction sequence is initiated by a user or an application program.
- It must continue through all succeeding SQL statements until one of the following four events occur:

  1) A **COMMIT statement** is reached, in which case all changes are permanently recorded within the database.

  The COMMIT statement automatically ends the SQL transaction.

  The COMMIT operations indicates successful end-of-transaction.

DBMS UNIT IV.pptx

Janagi Raman S

Transaction Execution with SQL (Cont...)

2) A **ROLLBACK statement** is reached, in which case all the changes are aborted and the database is rolled back to its previous consistent state.

   The ROLLBACK operation indicates unsuccessful end-of-transaction.

3) The end of a program is successfully reached, in which case all changes are permanently recorded within the database. This action is equivalent to COMMIT.

4) The program is abnormally terminated, in which case the changes made in the database are aborted and the database is rolled back to its previous consistent state. This action is equivalent to ROLLBACK.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

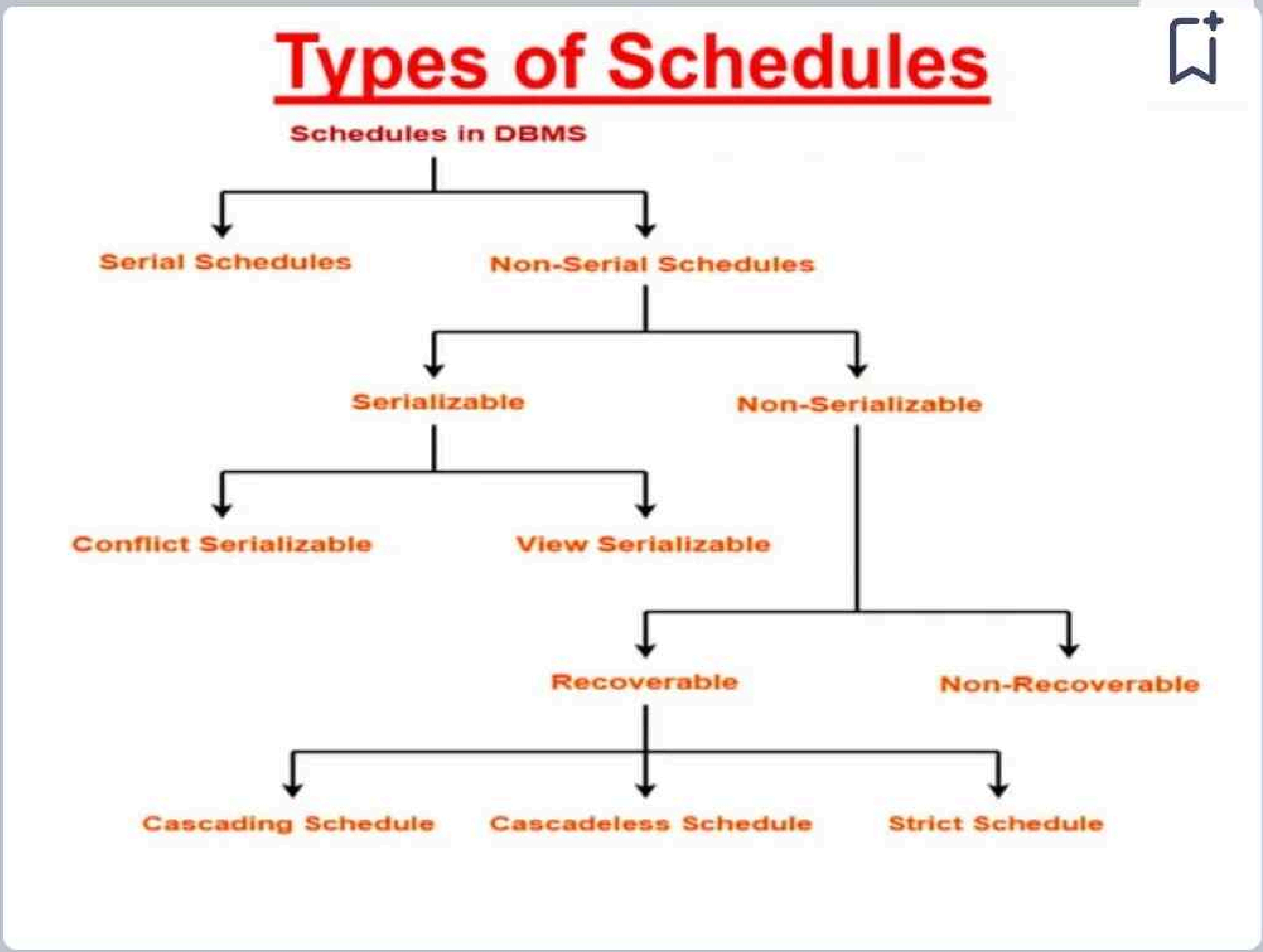
Need for Concurrency Control:

❖ When transactions are executed in an uncontrolled manner, several problems can occur.

1. Lost updates.
2. Dirty read (or uncommitted data).
3. Unrepeatable read (or inconsistent retrievals).

DBMS UNIT IV.pptx

Janagi Raman S



DBMS UNIT IV.pptx

SERIAL SCHEDULES

Serial Schedules-

- In serial schedules, All the transactions execute serially one after the other.
- When one transaction executes, no other transaction is allowed to execute.

Characteristics-

Serial schedules are always-

1. Consistent
2. Recoverable
3. Cascadeless
4. Strict

DBMS UNIT IV.pptx

Janagi Raman S

SERIAL SCHEDULES

Serial Schedules-

Transaction T1	Transaction T2
R (A)	
W (A)	
R (B)	
W (B)	
Commit	
	R (A)
	W (B)
	Commit

In this schedule,

- There are two transactions T1 and T2 executing serially one after the other.
- Transaction T1 executes first.
- After T1 completes its execution, transaction T2 executes.
- So, this schedule is an example of a **Serial Schedule.**

DBMS UNIT IV.pptx

Janagi Raman S

SERIAL SCHEDULES

Serial Schedules-

Transaction T1	Transaction T2
	R (A)
	W (B)
	Commit
R (A)	
W (A)	
R (B)	
W (B)	
Commit	

In this schedule,

❖There are two transactions T1 and T2 executing serially one after the other.

❖Transaction T2 executes first.

❖After T2 completes its execution, transaction T1 executes.

❖So, this schedule is an example of a **Serial Schedule**.

DBMS UNIT IV.pptx

Janagi Raman S

NON SERIAL SCHEDULES

In non-serial schedules,

- Multiple transactions execute concurrently.
- Operations of all the transactions are inter leaved or mixed with each other.

Characteristics-

Non-serial schedules are **NOT** always-

1. Consistent
2. Recoverable
3. Cascadeless
4. Strict

NON SERIAL SCHEDULES

Transaction T1	Transaction T2
R (A)	
W (B)	
	R (A)
R (B)	
W (B)	
Commit	
	R (B)
	Commit

In this schedule,

- There are two transactions T1 and T2 executing concurrently.
- The operations of T1 and T2 are interleaved.
- So, this schedule is an example of a **Non-Serial Schedule**.

DBMS UNIT IV.pptx

Janagi Raman S

NON SERIAL SCHEDULES

Transaction T1	Transaction T2
	R (A)
R (A)	
W (B)	
	R (B)
	Commit
R (B)	
W (B)	
Commit	

In this schedule,

- There are two transactions T1 and T2 executing concurrently.
- The operations of T1 and T2 are interleaved.
- So, this schedule is an example of a **Non-Serial Schedule.**

DBMS UNIT IV.pptx

Janagi Raman S

Serializability

- Some Non-Serial schedules may lead to inconsistency of the database.
- **Serializability** is a concept that helps to identify which Non-Serial schedules are correct and will maintain the consistency of the database.

DBMS UNIT IV.pptx

Janagi Raman S

Serializable Schedules

- Non-Serial schedule of **'n'** transactions is equivalent to some serial schedule of 'n' transactions, then it is called as a **serializable schedule**.

Characteristics-

Serializable schedules behave exactly same as serial schedules.

Thus, serializable schedules are always-

1. **Consistent**
2. **Recoverable**
3. **Cascadeless**
4. **Strict**

DBMS UNIT IV.pptx

Janagi Raman S

Serial Schedules Vs Serializable Schedules

Serial Schedules	Serializable Schedules
No concurrency is allowed. Thus, all the transactions necessarily execute serially one after the other.	Concurrency is allowed. Thus, multiple transactions can execute concurrently.
Serial schedules lead to less resource utilization and CPU throughput.	Serializable schedules improve both resource utilization and CPU throughput.
Serial Schedules are less efficient as compared to serializable schedules. (due to above reason)	Serializable Schedules are always better than serial schedules. (due to above reason)

DBMS UNIT IV.pptx

Janagi Raman S

Conflict Serializability

❖ **Definition:**

If a given Non-Serial schedule can be converted into a serial schedule by **swapping** its Non-Conflicting operations, then it is called as a **Conflict Serializable Schedule**.

DBMS UNIT IV.pptx

Janagi Raman S

Conflict Serializability

Conflict Operations:

Two operations are called as conflicting operations if all the following conditions hold true for them-

1. **Both the operations belong to different transactions.**
2. **Both the operations are on the same data item.**
3. **At least one of the two operations is a write operation.**

DBMS UNIT IV.pptx

Janagi Raman S

Conflict Serializability

Conflict Operations:

Transaction T1	Transaction T2
R1 (A)	
W1 (A)	
	R2 (A)
R1 (B)	

In this schedule,

- W1(A) and R2(A) are called as conflicting operations.
- This is because all the above conditions hold true for them.

DBMS UNIT IV.pptx

TESTING FOR CONFLICT SERIALIZABIL

- **Precedence Graph** can be used to test a schedule for Conflict Serializability.
- Precedence Graph is directed graph **G= (N, E)** that consist of a set of nodes **N = {T_1, T_2, ...T_n}** and a directed edges **E = {e_1, e_2,...e_3}**.
- Each edge **e_i** in the graph is of the form ($T_j \rightarrow T_k$), where **T_j** is the **starting node** of the edge e_i and of the edge e_i. **T_k** is the **ending node**
- An edge is created if one of the operations in T_j appears in the schedule before some conflicting operation in T_k.

DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

1. For each transaction T_i participating in schedule S, create a node labeled Ti in the precedence graph.
2. For each case in S where T_j executes a **Read(X) after** T_i executes a **Write(X)**, create an edge (**$T_i \rightarrow T_j$**) in the precedence graph.
3. For each case in S where T_j executes a **Write(X) after** T_i executes a **Read(X)**, create an edge (**$T_i \rightarrow T_j$**) in the precedence graph.
4. For each case in S where T_j executes a **Write(X) after** T_i executes a **Write(X)**, create an edge (**$T_i \rightarrow T_j$**) in the precedence graph.
5. The schedule **S is serializable** if and only if the precedence graph **has no cycles**.

DBMS UNIT IV.pptx

TESTING FOR CONFLICT SERIALIZABIL

SCHEDULE C

T1	T2
Read(X)	
X:=X-N	
	Read(X)
	X:=X+M
Write(X)	
Read(Y)	
	Write(X)
Y:=Y+N	
Write(Y)	

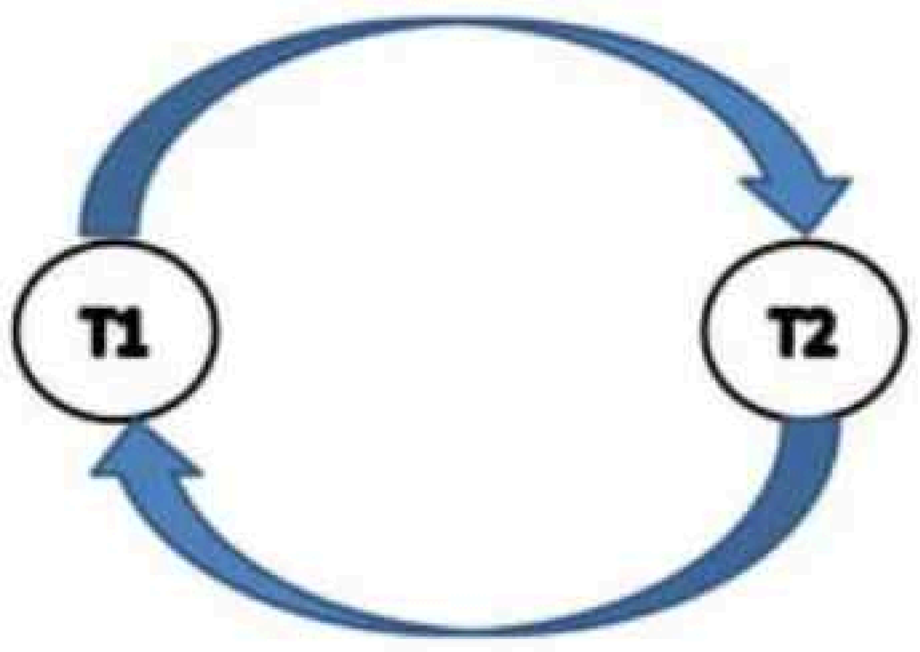
DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

SCHEDULE C

T1	T2
Read(X)	
X:=X-N	
	Read(X)
	X:=X+M
Write(X)	
Read(Y)	
	Write(X)
Y:=Y+N	
Write(Y)	



Graph has Cycle. So Schedule C is Not Serializable Schedule.

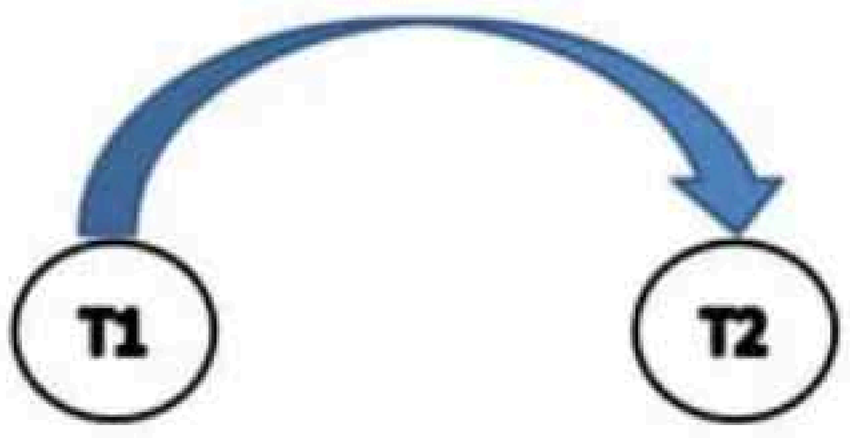
DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

SCHEDULE D

T1	T2
Read(X)	
X:=X-N	
Write(X)	
	Read(X)
	X:=X+M
	Write(X)
Read(Y)	
Y:=Y+N	
Write(Y)	



Graph has no Cycle. So Schedule D is Serializable Schedule.

DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

PRACTICE PROBLEMS BASED ON CONFLICT SERIALIZABILITY-

Problem-01: Check whether the given schedule S is conflict serializable or not-

S : $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

PRACTICE PROBLEMS BASED ON CONFLICT SERIALIZABILITY-

Problem-01: Check whether the given schedule S is conflict serializable or not-

S : $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

T1	T2	T3
R1(A)		
	R2(A)	
R1(B)		
	R2(B)	
		R3(B)
W1(A)		
	W3(B)	

DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

T1	T2	T3
R1(A)		
	R2(A)	
R1(B)		
	R2(B)	
		R3(B)
W1(A)		
	W3(B)	

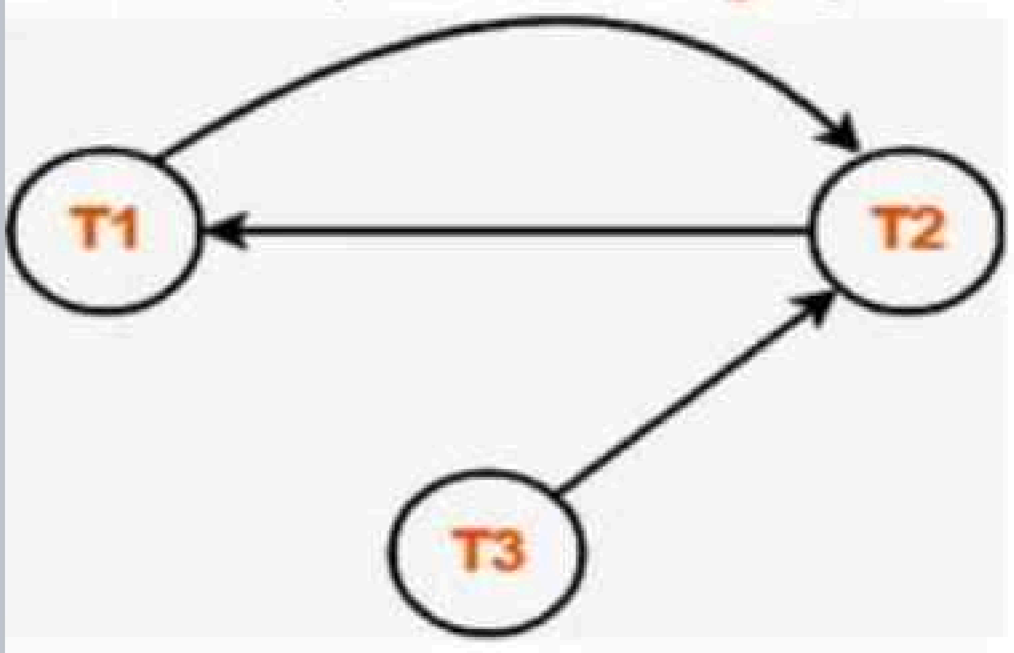
List all the conflicting operations

R2(A) , W1(A) (T2 → T1)

R1(B) , W2(B) (T1 → T2)

R3(B) , W2(B) (T3 → T2)

Draw the precedence graph-



- Clearly, there exists a **cycle** in the precedence graph.
- Therefore, the given schedule **S is not conflict serializable.**

DBMS UNIT IV.pptx

Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

PRACTICE PROBLEMS BASED ON CONFLICT SERIALIZABILITY-

Problem-02: Check whether the given schedule S is conflict serializable and recoverable or not-

T1	T2	T3	T4
	R(X)		
		W(X)	
		Commit	
W(X)			
Commit			
	W(Y)		
	R(Z)		
	Commit		
			R(X)
			R(Y)
			Commit

DBMS UNIT IV.pptx

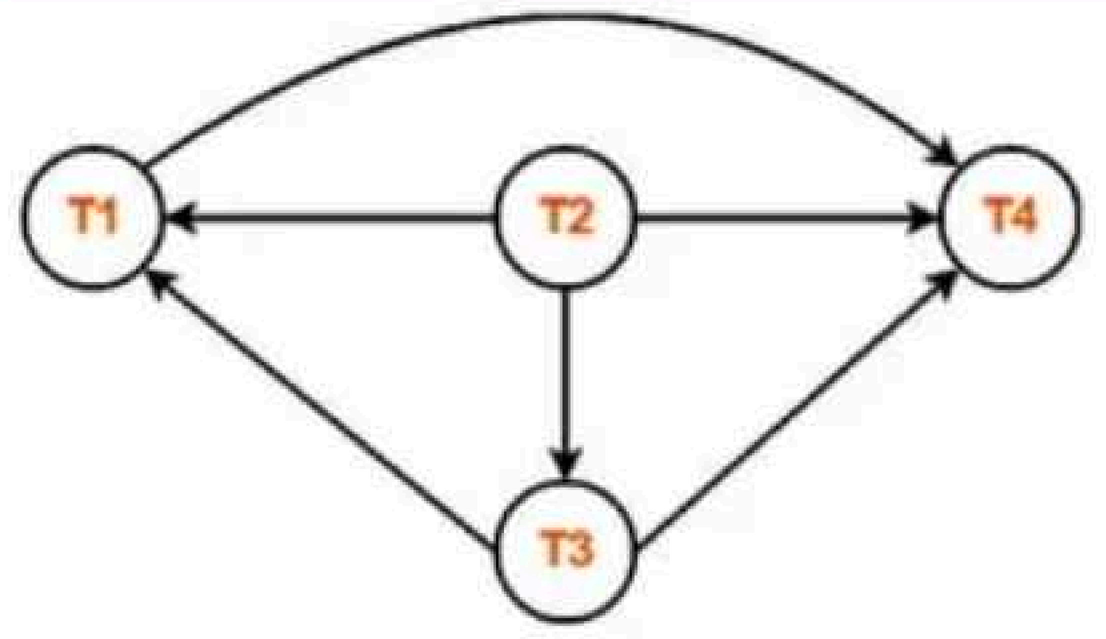
Janagi Raman S

TESTING FOR CONFLICT SERIALIZABIL

List all the conflicting operations and determine the dependency between the transactions-

$R_2(X)$, $W_3(X)$	$(T_2 \rightarrow T_3)$
$R_2(X)$, $W_1(X)$	$(T_2 \rightarrow T_1)$
$W_3(X)$, $W_1(X)$	$(T_3 \rightarrow T_1)$
$W_3(X)$, $R_4(X)$	$(T_3 \rightarrow T_4)$
$W_1(X)$, $R_4(X)$	$(T_1 \rightarrow T_4)$
$W_2(Y)$, $R_4(Y)$	$(T_2 \rightarrow T_4)$

Draw the precedence graph-



- Clearly, there exists **no cycle** in the precedence graph.
- Therefore, the given schedule **S is conflict serializable**.

DBMS UNIT IV.pptx

Janagi Raman S

CONFLICT SERIALIZABILITY

CONFLICT EQUIVALENCE:

Schedule A

T1	T2
Read(X)	
X:=X-N	
Write(X)	
Read(Y)	
Y:=Y+N	
Write(Y)	
	Read(X)
	X:=X+M
	Write(X)

Schedule C

T1	T2
Read(X)	
X:=X-N	
	Read(X)
	X:=X+M
Write(X)	
Read(Y)	
	Write(X)
Y:=Y+N	
Write(Y)	

Re-order Schedule C

T1	T2
Read(X)	
X:=X-N	
Write(X)	
Read(Y)	
Y:=Y+N	
Write(Y)	
	Read(X)
	X:=X+M
	Write(X)

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

Definition:

If a given schedule is found to be view equivalent to some serial schedule, then it is called as a view serializable schedule.

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

View Equivalent Schedules-

- Schedules S1 and S2 are called view equivalent if the following three conditions hold true for them-

Condition-01:

For each data item X, if transaction T_i reads X from the database initially in schedule S1, then in schedule S2 also, T_i must perform the initial read of X from the database.

Condition-02:

If transaction T_i reads a data item that has been updated by the transaction T_j in schedule S1, then in schedule S2 also, transaction T_i must read the same data item that has been updated by the transaction T_j.

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

View Equivalent Schedules-

Condition-03:

For each data item X, if X has been updated at last by transaction T_i in schedule S1, then in schedule S2 also, X must be updated at last by transaction T_i.

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

Checking Whether a Schedule is View Serializable Or Not-

Method-01:

1. Check whether the given schedule is conflict serializable or not.
2. If the given schedule is conflict serializable, then it is surely view serializable. Stop and report your answer.
3. If the given schedule is not conflict serializable, then it may or may not be view serializable.

- All conflict serializable schedules are view serializable.
- All view serializable schedules may or may not be conflict serializable.

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

Checking Whether a Schedule is View Serializable Or Not-

Method-02:

1. Check if there exists any blind write operation.

   (Writing without reading is called as a blind write).

1. If there does not exist any blind write, then the schedule is surely not view serializable. Stop and report your answer.
2. If there exists any blind write, then the schedule may or may not be view serializable.

❖No blind write means not a view serializable schedule.

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

Checking Whether a Schedule is View Serializable Or Not-

Method-03:

1. In this method, try finding a view equivalent serial schedule.
2. Then, draw a graph using those dependencies.
3. If there exists no cycle in the graph, then the schedule is view serializable otherwise not.

DBMS UNIT IV.pptx

Janagi Raman S

View Serializability

Checking Whether a Schedule is View Serializable Or Not-

Problem-01:

T1	T2	T3	T4
R (A)			
	R (A)		
		R (A)	
			R (A)
W (B)			
	W (B)		
		W (B)	
			W (B)

DBMS UNIT IV.pptx

Janagi Raman S

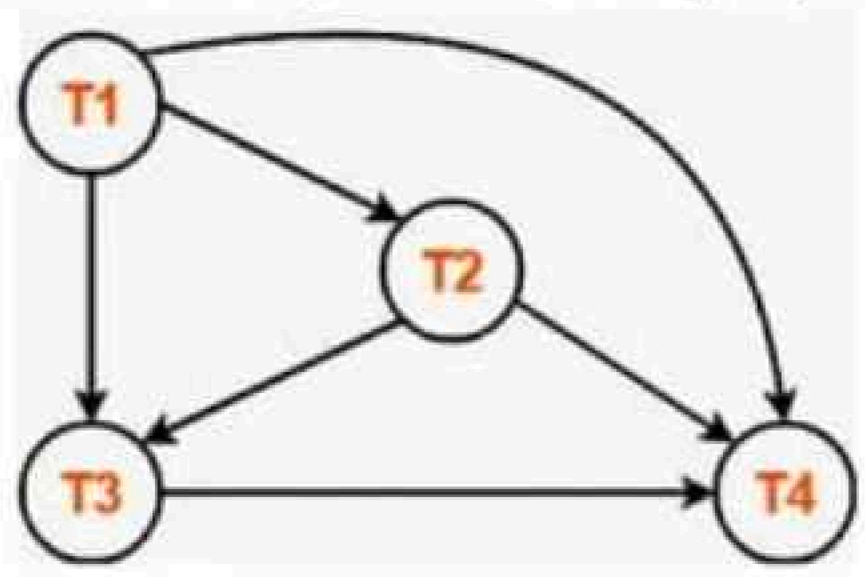
View Serializability

Checking Whether a Schedule is View Serializable Or Not

- If a schedule is conflict serializable, then it is surely view serializable.
- List all the conflicting operations and determine the dependency between the transactions-

W1(B) , W2(B)	(T1 → T2)
W1(B) , W3(B)	(T1 → T3)
W1(B) , W4(B)	(T1 → T4)
W2(B) , W3(B)	(T2 → T3)
W2(B) , W4(B)	(T2 → T4)
W3(B) , W4(B)	(T3 → T4)

Draw the precedence graph-



- Clearly, there exists no cycle in the precedence graph.
- Therefore, the given schedule S is conflict serializable.
- Thus, we conclude that the given schedule is also view serializable.

DBMS UNIT IV.pptx

Janagi Raman S

Recoverability in DBMS

- A schedule is the order in which the operations of multiple transactions appear for execution.
- Non-serial schedules may be serializable or non-serializable.

Non-Serializable Schedules

- A non-serial schedule which is not serializable is called as a non-serializable schedule.

Characteristics-

- may or may not be consistent
- may or may not be recoverable

DBMS UNIT IV.pptx

Janagi Raman S

Recoverable Schedule

If in a schedule,

1. A transaction performs a dirty read operation from an uncommitted transaction.
2. And its commit operation is delayed till the uncommitted transaction either commits or roll backs

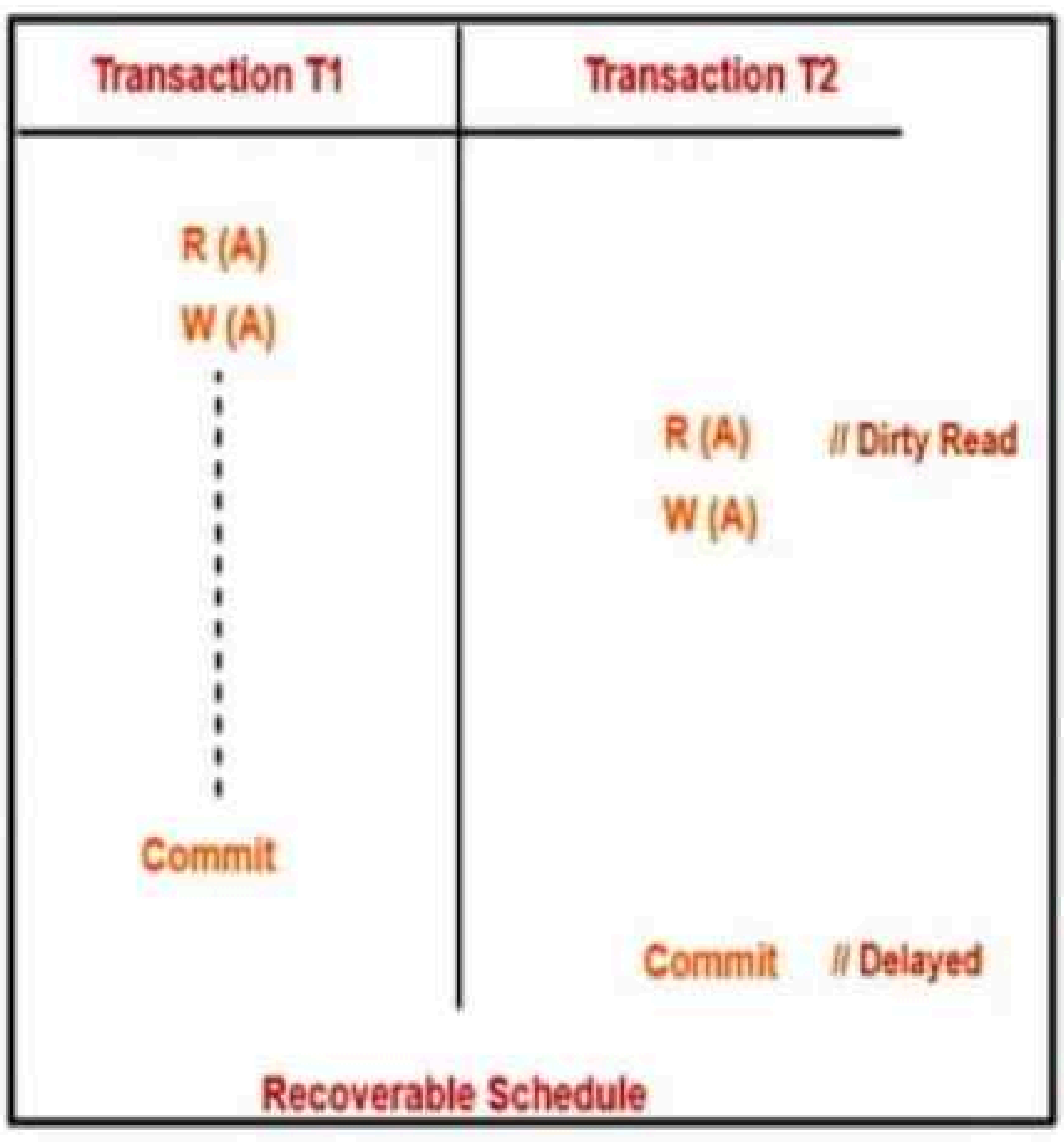
Then such a schedule is known as a **Recoverable Schedule**.

DBMS UNIT IV.pptx

Janagi Raman S

Oct 27, 2022 • 0 likes • 136 views

Recoverable Schedule



Here,

- T2 performs a dirty read operation.
- The commit operation of T2 is delayed till T1 commits or roll backs.
- T1 commits later.
- T2 is now allowed to commit.
- In case, T1 would have failed, T2 has a chance to recover by rolling back.

DBMS UNIT IV.pptx

Janagi Raman S

Non -Recoverable Schedul

If in a schedule,

- A transaction performs a dirty read operation from an uncommitted transaction
- And commits before the transaction from which it has read the value then such a schedule is known as an **Non - Recoverable Schedule**.

DBMS UNIT IV.pptx

Janagi Raman S

Non -Recoverable Schedul

Transaction T1	Transaction T2
R (A)	
W (A)	
	R (A) // Dirty Read
	W (A)
	Commit
Rollback	

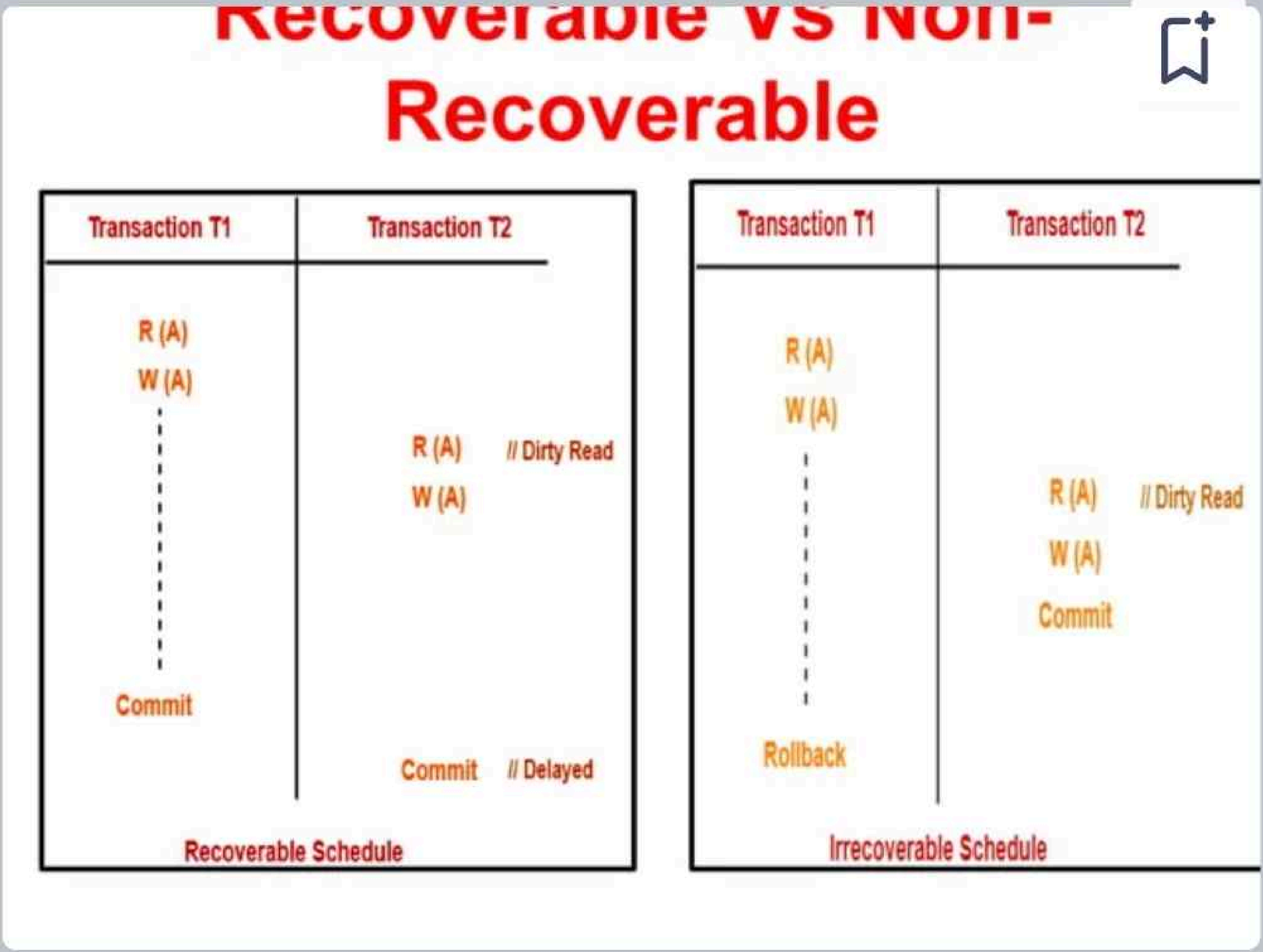
Irrecoverable Schedule

Here,

- T2 performs a dirty read operation.
- T2 commits before T1.
- T1 fails later and roll backs.
- The value that T2 read now stands to be incorrect.
- T2 can not recover since it has already committed.

DBMS UNIT IV.pptx

Janagi Raman S



DBMS UNIT IV.pptx

Janagi Raman S

Recoverable Problems

Checking Whether a Schedule is Recoverable or Irrecoverable-

Method-01:

- Check whether the given schedule is conflict serializable or not.
- If the given schedule is conflict serializable, then it is surely recoverable. Stop and report your answer.
- If the given schedule is not conflict serializable, then it may or may not be recoverable. Go and check using other methods.

- All conflict serializable schedules are recoverable.
- All recoverable schedules may or may not be conflict serializable.

DBMS UNIT IV.pptx

Recoverable Problems

Checking Whether a Schedule is Recoverable or Irrecoverable-

Method-02:

- Check if there exists any dirty read operation.

(Reading from an uncommitted transaction is called as a dirty read)

- If there does not exist any dirty read operation, then the schedule is surely recoverable. Stop and report your answer.
- If there exists any dirty read operation, then the schedule may or may not be recoverable.

DBMS UNIT IV.pptx

Janagi Raman S

SCHEDULES BASED ON RECOVER

SCHEDULE B

T1	T2
Read(X)	
Write(X)	
	Read(X)
Read(Y)	
	Write(X)
	Commit
Abort	

- The Schedule B is Not Recoverable.
- T2 reads item X from T1.
- T2 commits before T1 commits.
- T1 Aborts after T2 Commits.
- The value of X that T2 read is no longer valid.
- T2 must be Aborted after its Committed.
- Leading to schedule that is not recoverable.

Note: For the schedule to be recoverable, the T2 Commit operation in Schedule B must be postponed until T1 Commits.

DBMS UNIT IV.pptx

Janagi Raman S

SCHEDULES BASED ON RECOVER

SCHEDULE C

T1	T2
Read(X)	
Write(X)	
	Read(X)
Read(Y)	
	Write(X)
Write(Y)	
Commit	
	Commit

- The Schedule C is Recoverable.
- T2 reads item X from T1.
- T2 commits after T1 commits.

DBMS UNIT IV.pptx

SCHEDULES BASED ON RECOVER

SCHEDULE D

T1	T2
Read(X)	
Write(X)	
	Read(X)
Read(Y)	
	Write(X)
Write(Y)	
Abort	
	Abort

- The Schedule D is Recoverable.
- T2 reads item X from T1.
- T1 Aborts instead of Committing, Then T2 Should also Abort.
- Because the value of X it read is no longer valid

DBMS UNIT IV.pptx

RECOVERABLE SCHEDULES

If in a schedule,

- A transaction performs a dirty read operation from an uncommitted transaction
- And its commit operation is delayed till the uncommitted transaction either commits or roll backs
- then such a schedule is called as a **Recoverable Schedule**.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Recoverable Schedules

I. Cascading Schedule-

If in a schedule,

- failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a **Cascading Schedule** or **Cascading Rollback** or **Cascading Abort**.
- It simply leads to the wastage of CPU time.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Recoverable Schedules

I. Cascading Schedule-

T1	T2	T3	T4
R (A)			
W (A)			
	R (A)		
	W (A)		
		R (A)	
		W (A)	
			R (A)
			W (A)
Failure			

Cascading Recoverable Schedule

Hère,

- Transaction T2 dépends on transaction T1.
- Transaction T3 dépends on transaction T2.
- Transaction T4 dépends on transaction T3.

In this schedule,

- The failure of transaction T1 causes the transaction T2 to rollback.
- The rollback of transaction T2 causes the transaction T3 to rollback.
- The rollback of transaction T3 causes the transaction T4 to rollback.
- Such a rollback is called as a **Cascading Rollback.**

DBMS UNIT IV.pptx

Janagi Raman S

Types of Recoverable Schedules

II. Cascadeless Schedule-

If in a schedule,

- "A transaction is not allowed to read a data item until the last transaction that has written it is committed or aborted, then such a schedule is called As a **Cascadeless Schedule**."
- In other words,

  "Cascadeless schedule allows only **committed read** operations."
- Therefore, it avoids cascading roll back and thus saves CPU time.

DBMS UNIT IV.pptx

Janagi Raman S

Oct 27, 2022 • 0 likes • 136 views

Types of Recoverable Schedules

II. Cascadeless Schedule-

T1	T2	T3
R (A)		
W (A)		
Commit		
	R (A)	
	W (A)	
	Commit	
		R (A)
		W (A)
		Commit

Cascadeless Schedule

- Cascadeless schedule allows only committed read operations.
- However, it allows uncommitted write operations.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Recoverable Schedules

III. Strict Schedule-

If in a schedule,

- "A transaction is neither allowed to read nor write a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a **Strict Schedule**."

In other words,

- "Strict schedule allows **only committed read and write** operations."
- Strict schedule implements more restrictions than Cascadeless schedule.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Recoverable Schedules

III. Strict Schedule-

T1	T2
W (A)	
Commit / Rollback	
	R (A) / W (A)

Strict Schedule

Here,

- Strict schedules are more strict than Cascadeless schedules.
- All strict schedules are Cascadeless schedules.
- All Cascadeless schedules are not strict schedules.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

- Concurrency control techniques that are used to ensure the non-interference or Isolation property of concurrently executing transactions.
- There are **three methods for concurrency control**. They are as Follows:
  1. Locking Methods
  2. Time-stamp Methods
  3. Optimistic Methods

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

1. **Locking Methods of Concurrency Control:**
   - "A lock is a variable, associated with the data item, which controls the access of that data item."
   - Locking is the most widely used form of the concurrency control mechanism.
   - Locks are further divided into three fields:
     1. Lock Granularity
     2. Lock Types
     3. Deadlocks

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

I. **Lock Granularity**:

- A database is a collection of named data items.
- The size of the data item chosen as the unit of protection by a concurrency control program is called **GRANULARITY**.
- Locking can take place at the following level :
  1. Database level locking.
  2. Table level locking.
  3. Page level locking.
  4. Row (Tuple) level locking.
  5. Attributes (fields) level locking.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

II. Lock Types:

The DBMS mainly uses following types of locking techniques.

a) Binary Locking

b) Shared / Exclusive Locking

c) Two - Phase Locking (2PL)

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) **BINARY LOCKS**:

- A binary lock can have two states or values:
  1) Locked state - (1)
  2) Unlocked state - (0)
- If the value of the lock on X is 1, then the item X cannot be accessed by a other operation.
- If the value of the lock on X is 0, the item can be accessed by other operation.
- The current state of the lock associated with item X is **LOCK(X)**.
- Two operations, lock_item and unlock_item, are used with binary locking.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) **BINARY LOCKS (Cont…):**

Lock_item(X):

- A transaction requests an item.
- If LOCK(X) = 1, then transaction has to **wait**.
- If LOCK(X) = 0, it is set to 1 and the transaction is allowed to access item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) **BINARY LOCKS (Cont…):**

Unlock_item (X):

- Sets LOCK(X) to 0 (unlocks) so that X may be accessed by other transactions. (LOCK(X)←0)
- A binary lock enforces mutual exclusion on the data item ; i.e., at a time only one transaction can hold a lock.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) BINARY LOCKS RULES (Cont…):

1. A transaction T must issue the operation lock_item(X) before any read_item(X) or write_item(X) operations are performed in T.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) **BINARY LOCKS RULES (Cont…):**

2. A transaction T must issue the operation unlock_item(X) after all read_item(X) and write_item(X) operations are completed in T.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) BINARY LOCKS RULES (Cont…):

3. A transaction T will not issue a lock_item(X) operation if it already holds the lock on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) BINARY LOCKS RULES (Cont…):

4. A transaction T will not issue an unlock_item(X) operation unless it already holds the lock on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

a) **BINARY LOCKS RULES (Cont…):**

1. A transaction T must issue the operation lock_item(X) before any read_item(X) or write_item(X) operations are performed in T.
2. A transaction T must issue the operation unlock_item(X) after all read_item(X) and write_item(X) operations are completed in T.
3. A transaction T will not issue a lock_item(X) operation if it already holds the lock on item X.
4. A transaction T will not issue an unlock_item(X) operation unless it already holds the lock on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Locking:

- Binary locking scheme allows at most one transaction can hold a lock on a given time.
- We should allow several transaction to access the same data item X if they all access X for reading purposes only.
- If a transaction is to write a data item X, it must have exclusive access to X.
- For this purpose, a different type of lock called multiple-mode lock is used.
- This scheme is called shared/exclusive lock.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Locking:

- There are three locking operations:

  1. Read_lock(X) 2. Write_lock(X) 3.Unlock(X)

- A lock associated with an data item X, **LOCK(X)**
- Has three states – 1) read-locked state

  2) write-locked state

  3) unlocked state.

DBMS UNIT IV.pptx

CONCURRENCY CONTROL

b) Shared / Exclusive Locking:

- A read-locked item is also called share-locked because other transactions are allowed to read the item.
- A write-locked item is called exclusive –locked because a single transaction exclusively holds the lock on the item.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Locking:

Shared Lock:

- These locks are referred as read locks, and denoted by **'S'.**
- If a transaction T has obtained Shared-lock on data item X, then T can read X, but cannot write X.
- Multiple Shared lock can be placed simultaneously on a data item.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Locking (Cont…):

Exclusive lock:

- These Locks are referred as Write locks, and denoted by **'X'.**
- If a transaction T has obtained Exclusive lock on data item X, then **T can be read as well as write X.**
- Only one Exclusive lock can be placed on a data item at a time.
- This means multiple transactions does not modify the same data simultaneously.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

1. A transaction T must issue the operation read_lock(X) or write_lock(X) before any read_item(X) opeartion is performed in T.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

2. A transaction T must issue the operation write_lock(X) before any write_item(X) operation is performed in T.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

3. A transaction T must issue the operation unlock(X) after all read_item(X) and write_item(X) operations are completed in T.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

4. A transaction will not issue a read_lock(X) operation if it already holds a read_lock(shared) or a write_lock (exclusive) on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

5. A transaction will not issue a write_lock(X) operation if it already holds a read_lock(shared) or a write_lock (exclusive) on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

6. A transaction will not issue a unlock(X) operation unless it already holds a read_lock(shared) or a write_lock (exclusive) on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Locking (Cont…):

T1:

Lock-X(B)
Read(B)
B:=B-50
Write(B)
Unlock(B)
Lock-X(A)
Read(A)
A:=A+50
Write(A)
Unlock(A)

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Locking (Cont...):

T2:

Lock-S(A)
Read(A)
Unlock(A)
Lock-S(B)
Read(B)
Unlock(B)
Display(A+B)

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

b) Shared / Exclusive Rules:

1. A transaction T must issue the operation read_lock(X) or write_lock(X) before any read_item(X) opeartion is performed in T.
2. A transaction T must issue the operation write_lock(X) before any write_item(X) opearation is performed in T.
3. A transaction T must issue the operation unlock(X) after all read_item(X) and write_item(X) operations are completed in T.
4. A transaction will not issue a read_lock(X) operation if it already holds a read (shared) lock or a write (exclusive) lock on item X.
5. A transaction will not issue a write_lock(X) operation if it already holds a read (shared) lock or a write (exclusive) lock on item X.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL):

- Two-phase locking (**2PL**) is a method of controlling concurrent processing in which all locking operations precede the first unlocking operation.
- Thus, a transaction is said to follow the two-phase locking protocol if all locking operations (read_Lock, write_Lock) precede the first unlock operation in the transaction.
- Two-phase locking is the standard protocol used to defines how transactions acquire and hand over locks.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL): (Cont…)

- This protocol requires that each transaction issue lock and unlock requests in two phases.
  1. **Growing Phase**: In this phase, a transaction may obtain locks, but may not release any lock.
  2. **Shrinking Phase**: In this phase, a transaction may release locks, but may not obtain any new locks.
- Initially, a transaction is in the **growing phase**. The transaction acquires locks as needed.
- Once the transaction releases a lock, it enters in the shrinking phase, and it cannot issue more lock request.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL): (Cont...)

T3
Lock-X(B)
Read(B)
B:=B-50
Write(B)
Lock-X(A)
Read(A)
A:=A+50
Write(A)
Unlock(B)
Unlock(A)

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL): (Cont...)

T1	T2
Raed_lock(Y)	Read_lock(X)
Read_item(Y)	Read_item(X)
Unlock(Y)	**Unlock(X)**
Write_lock(X)	**Write_lock(Y)**
Read_item(X)	Read_item(Y)
X:X+Y	Y:=X+Y
Write_item(X)	Write_item(Y)
Unlock(X)	Unlock(Y)

Not obey two phase locking

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL): (Cont...)

T1	T2
Raed_lock(Y)	Read_lock(X)
Read_item(Y)	Read_item(X)
Write_lock(X)	**Write_lock(Y)**
Unlock(Y)	**Unlock(X)**
Read_item(X)	Read_item(Y)
X:X+Y	Y:=X+Y
Write_item(X)	Write_item(Y)
Unlock(X)	Unlock(Y)

Obey two phase locking

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL): (Cont...)

Time	Transaction	Remarks
t0	Lock - X (A)	Acquière Exclusive lock on A.
t1	Read A	read original value of A
t2	A = A - 100	subtract 100 from A
t3	Write A	write new value of A
t4	Lock - X (B)	Acquière Exclusive lock on B.
t5	Read B	read original value of B
t6	B = B + 100	add 100 to B
t7	Write B	write new value of B
t8	Unlock (A)	release lock on A
t9	Unock (B)	release lock on B

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

c) Two-Phase Locking (2PL): (Cont…)

Advantages:

1. The two-phase locking ensures conflict serializability.

Disadvantages:

1. Two-phase locking does not ensure freedom from deadlock.
2. Cascading Roll-backing may occur under two-phase locking.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Strict Two-Phase Locking (2PL):

- Strict two-phase locking method used in concurrent systems.
- Strict Two-Phase is most used in real-world implementation.
- **Rules of Strict 2PL:**
  1. If a transaction T wants to read/write an object, it must request a shared/exclusive lock on the object.
  2. All exclusive locks held by transaction T are released when T commits.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Strict Two-Phase Locking (2PL):

1. A Transaction T does not release any of its Exclusive Lock until it Commits or Aborts.
2. Hence, no other transaction can read or write an item that is written by T unless T has committed.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Strict Two-Phase Locking (2PL):

T1	T2
Lock-S(A)	
Read(A)	
	Lock-S(A)
	Read(A)
	Loc-X(B)
	Read(B)
	Write(B)
	Commit
Lock-X(C)	
Read(C)	
Write(C)	
Commit	

- Strict Two-phase locking prevents transaction from reading **uncommitted data**, **overwriting uncommitted data** and **unrepeatable reads**.
- Strict 2PL prevents Cascading Rollbacks.
- Strict 2PL does not guarantee a Deadlock Free Schedule.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Rigorous Two-Phase Locking (2PL):

- This protocol that all locks be held until the transactions commits.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

3. DEADLOCKS :

Definition:

"A System is in deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set."

"There exists a set of transactions {T0, T1, T2,....Tn} such that T0 is waiting for a data item that T2 holds and T2 waiting for a data item that T3 holds.... Tn is waiting for data item that T0 holds. In such situation, none of the transaction can make progress"

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

3. DEADLOCKS :

T1	T2
Lock-X(B)	
Read(B)	
B:=B-50	
Write(B)	
	Lock-X(A)
	Read(A)
	Lock-S(B)
Lock-X(A)	

- Here T1 is holding an Exclusive lock on B and t2 is requesting a Shared lock on B, T2 is waiting for T1 to unlock B.
- Similarly, T2 is holding Shared lock on A, T1 is requesting an Exclusive lock on A, T1 is waiting for T2 to unlock A.
- In this situation neither of transaction can proceed with normal execution.
- This situation is called DEADLOCK.

CONCURRENCY CONTROL

3. DEADLOCKS :

There are two principle methods for dealing with deadlock problem

1. **Deadlock Prevention**: This approach ensures that the system will never enter in deadlock state.
2. **Deadlock Detection and Recovery**: This approach tries to recover from Deadlock if system enters in deadlock state.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

DEADLOCK DETECTION & PREVENTION:

Deadlock Prevention:

- Deadlock prevention technique avoids the conditions that lead to deadlocking.
- It requires that every transaction lock all data items it needs in advance.
- In other words, a transaction requesting a new lock is aborted if there is the possibility that a deadlock can occur.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Deadlock Prevention:

There are two approaches for deadlock prevention:

Approach:1

- This ensures that no cyclic wait can occur by ordering the request for locks.
- It requires that every transaction lock all data items it needs in advance.
- It is required that, either **all data items should be locked** in one step, or **none should be locked.**
- **Disadvantages of this approach**:

1. It is **hard to predict** before the transaction begins, What data items need to be locked.
2. Data item **utilization may be very low**, because many of the data items may be locked but unused for a long time.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Deadlock Prevention:

Approach:2

- Deadlock prevention is to use **preemption** and **transaction rollbacks**.
- In preemption when a transaction T2 request a lock that transaction T1 holds.
- The lock granted to T1 may be preempted by rolling back T1, and granting of lock to T2.
- To control preemption, a unique timestamp is assigned to each transaction.
- The system uses timestamp to decide whether a transaction wait or roll back.

CONCURRENCY CONTROL

Deadlock Prevention:

Two different deadlock prevention schemes using timestamp are:

1. Wait Die.
2. Wound Wait.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Deadlock Prevention:

1. Wait Die:

- The Wait-Die scheme is non-preemption technique.
- When Transaction T_i request a data item held by T_j, T_i is allowed to wait only if it has a timestamp smaller than T_j.
- Otherwise, T_i is rolled back (Die)

EXAMPLE:

- Consider three transactions T1, T2, T3 with timestamps 5, 10, 15 respectively.
- If T1 request a data item held by T2, Then T1 will wait.
- If T3 request data item held by T2, then T2 will be Rolled Back.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

Deadlock Prevention:

2. Wound Wait:

- The Wound Wait scheme is preemption technique.
- When Transaction T_i request a data item held by T_j, T_i is allowed to wait only if it has a timestamp greater than T_j.
- Otherwise, T_i is rolled back (Die)

EXAMPLE:

- Consider three transactions T1, T2, T3 with timestamps 5, 10, 15 respectively.
- If T1 request a data item held by T2, Then T2 will be Rolled Back.
- If T3 request data item held by T2, then T3 will wait.

DBMS UNIT IV.pptx

Janagi Raman S

CONCURRENCY CONTROL

DEADLOCK DETECTION & PREVENTION:

Deadlock detection:

- This technique allows deadlock to occur, but then, it detects it and solves it.
- Here, a database is periodically checked for deadlocks.
- If a deadlock is detected, one of the transactions, involved in deadlock cycle, is aborted. other transaction continue their execution.
- An aborted transaction is rolled back and restarted.

DBMS UNIT IV.pptx

Janagi Raman S

DATABASE RECOVERY TECHNIQUES

Database Failure:

- We cannot guarantee a failure-free transaction every time. There are several types of failures that affect database processing.
- Some failures may affect the disk storage, while some may only affect the data items of the database residing in the main memory.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

- Failures are generally classified as Transaction, System, and Media Failures.
- There are several possible reasons for a transaction to fail in the middle of execution.

1. **A Computer Failure** (System Crash):
   - A hardware, software, or network error occurs in the computer system during transaction execution.
   - Hardware crashes are usually Media Failure
   - Example: Main Memory Failure.

DBMS UNIT IV.pptx

Types of Failure

2. **A Transaction or System error**:

- Some operation in the transaction may cause it to fail, such as **integer overflow** or **divide by zero**.
- Transaction Failure may also occur because of **erroneous parameter values** or because of a **logical programming error**.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

3. Disk Failure:

- Some disk blocks may lose their data because of a read or write malfunction or because of a read/write head crash.
- This may happen during a read or a write operation of the transaction.

DBMS UNIT IV.pptx

Types of Failure

4. **Physical Problem and Catastrophes:**
   - This refers to an endless list of problems that includes power failure or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

DBMS UNIT IV.pptx

Types of Failure

- Failure of types 1 and 2 are more common.
- Whenever a failure of type 1 and 2 occurs, the system must keep sufficient information to recover from the failure.
- Disk Failure and Catastrophic Failure are do not happen frequently.
- If they occur, recovery is a major task.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

- Recovery from transaction failure means that the database is restored to the most recent consistent state just before the time of failure.
- Wide portion of the database due to catastrophic failure, the recovery method **restores past copy** of the database and **reconstructs** a more current state by **redoing** the operations **up to the time of failure**.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

- When the database is not physically damaged, the recovery system is to reverse any changes that caused the inconsistency by undoing some operations.
- It may also redo some operation in order to restore a consistent state of the database.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

There are many different approaches to recover a database: Manual reprocessing, a variety of automated recovery techniques.

1. Manual Reprocessing

- The database is periodically backed up and all transactions applied since the last save are recorded.
- If the system crashes, the latest database save is restored and all of the transactions are re-applied (by users) to bring the database back up to the point just before the crash.

Limitations: 1) **Time required** to reapply transactions.

2)Transactions **might have other** (physical) potential **failures**.

3) Reapplying concurrent transactions is **not straight forward**.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. Automated Recovery:

There are several types of automated recovery techniques including: deferred update and immediate update, shadow paging, etc.

A transaction can be in one of the following states:

1. **Active** - When the transaction just begins
2. **Partially Committed** - After the last operation has completed (but before the commit point is reached)
3. **Failed** - Normal operation is prevented.
4. **Aborted** - Transaction is rolled back.
5. **Committed** - Transaction completes all operations and moved the database to the next consistent state

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. Automated Recovery: (Cont…)

- Each transaction writes the following information to the log:
- Start(T) - the fact that transaction T has started.
- Write(T, X, old_value, new_value) - The Transaction T has written to item X with the new_value. old_value is also maintained.
- Read(T, X) - The Transaction T has read data item X
- Either
  - Commit(T) - Transaction T committed, or
  - Abort(T) - transaction T was aborted

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. Automated Recovery: (Cont…)

Checkpoint:

- A recovery point in the logs where all current transactions have terminated and all updates have been written to disk. *Consists of 4 steps*:
  1. Cease accepting new transactions
  2. Allow all unfinished transactions to complete (commit or abort)
  3. Write all pending changes to disk and to logs
  4. Resume accepting new transactions
- In many environments, it is possible to take checkpoints each 15 minutes or half hour, etc.
- Recovery must then only be done from the time of the last checkpoint.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

1. DEFERRED UPDATE RECOVERY:

- Also called **NO-UNDO/REDO**
- During a transaction, only record the changes to data items in the log.
- When the transaction commits, update the data items on disk.
- Two main rules:
  1. A transaction cannot change any items in the database until it commits.
  2. A transaction may not commit until the entire write operations are successfully recorded in the log.

     This means that we must check the log is actually written to disk.

22:32 ··· 91

DBMS UNIT IV.pptx
slideshare.net

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

1. DEFERRED UPDATE RECOVERY: (Cont...)

```
T1:   Ra Rd Wd C
T2:   Rb Wb Rd Wd C
T3:   Ra Wa Rc Wc C
T4:   Rb Wb Ra Wa C

Log file:
Start(T1)
Write(T1, d, old, new)
Commit(T1)
checkpoint
Start(T4)
Write(T4, b, old, new)
Write(T4, a, old. new)
Commit(T4)
Start(T2)
Write(T2, b, old, new)
Start(T3)
Write(T3, a, old, new)
    s y s t e m   c r a s h
```

- Since T1 and T4 committed, their changes were written to disk.
- However, T2 and T3 did not commit, hence their changes were not written to disk.
- T2 and T3 are ignored because they did not reach their commit points.
- T4 is redone because its commit point is after the last system checkpoint.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

1. DEFERRED UPDATE RECOVERY: (Cont…)

Advantages:

1. **Recovery is made easier**:

   Any transaction that reached the commit point (from the log) has its writes applied to the database. All other transactions are ignored.

1. **Cascading rollback does not occur** because no other transaction sees the work of another until it is committed.

Disadvantages:

1. **Concurrency is limited:** Must employ Strict 2PL which limits concurrency.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. IMMEDIATE UPDATE RECOVERY

- A Transaction issues an update command, the database can be updated immediately, without any need to wait for the transaction to reach its commit point.
- An update operation must still be *recorded in the log* (*on disk*) before it is applied to the database – using the ***write ahead logging protocol***.
- So that we can recover in case of failure.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. IMMEDIATE UPDATE RECOVERY

- Immediate Update allows the write operations to the database as the transaction is executing.
- Writes are still saved in the log before being applied to the database - a ***Write-Ahead Log*** (WAL)
- Maintain two logs:
  - **REDO log**: A record of each new data item in the database.
  - **UNDO** log: A record of each updated data item (old values).

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. IMMEDIATE UPDATE RECOVERY: (Cont…)

Two rules:

1. Transaction T may not update the database until all UNDO entries have been written to the UNDO log.
2. Transaction T is not allowed to commit until all REDO and UNDO log entries are written (*forced-written* to disk).

DBMS UNIT IV.pptx

Types of Failure

2. IMMEDIATE UPDATE RECOVERY: (Cont…)

To Recover:

1. Begin at the end of the log and read backwards to the last checkpoint Create two lists:
   - **C** - transactions that have committed (Committed Transaction)
   - **NC** - transactions that did not commit (Active Transaction)
2. Undo all the write operations of the Active transactions from the log, using UNDO procedure.
3. Redo the write operations of the Committed Transactions from the log using REDO procedure.

DBMS UNIT IV.pptx

Janagi Raman S

2. IMMEDIATE UPDATE RECOVERY: (Cont…)

```
T1:   Ra Rd Wd C
T2:   Rb Wb Rd Wd C
T3:   Ra Wa Rc Wc C
T4:   Rb Wb Ra Wa C

Log file:
Start(T1)
Write(T1, d, old, new)
Commit(T1)
checkpoint
Start(T4)
Write(T4, b, old, new)
Write(T4, a, old. new)
Commit(T4)
Start(T2)
Write(T2, b, old, new)
Start(T3)
Write(T3, a, old, new)
    s y s t e m   c r a s h
```

- Since T1, T2, T3 and T4 changes were written to disk.
- However, T1 and T4 are committed.
- T2 and T3 are not committed.
- T2 and T3 are in Active Transaction List and T4 is in Committed Transaction.
- T2 and T3 are Undone from the log using UNDO procedure.
- T4 is redone because its commit point is after the last system checkpoint using REDO procedure.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

2. IMMEDIATE UPDATE RECOVERY: (Cont…)

Advantages:

Immediate update **allows higher concurrency** because transactions write continuously to the database rather than waiting until the commit point.

Disadvantages:

Can lead to cascading rollbacks - time consuming and may be problematic.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

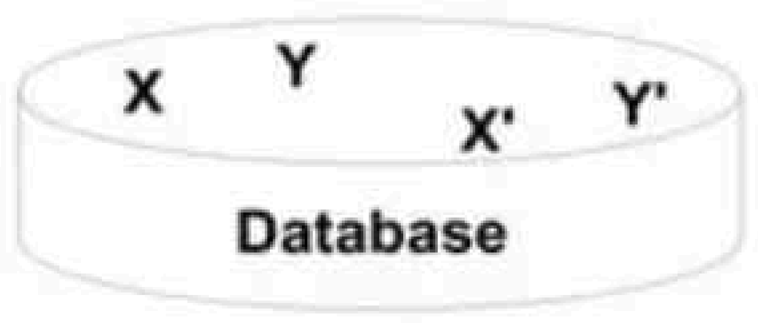
3. SHADOW PAGING:

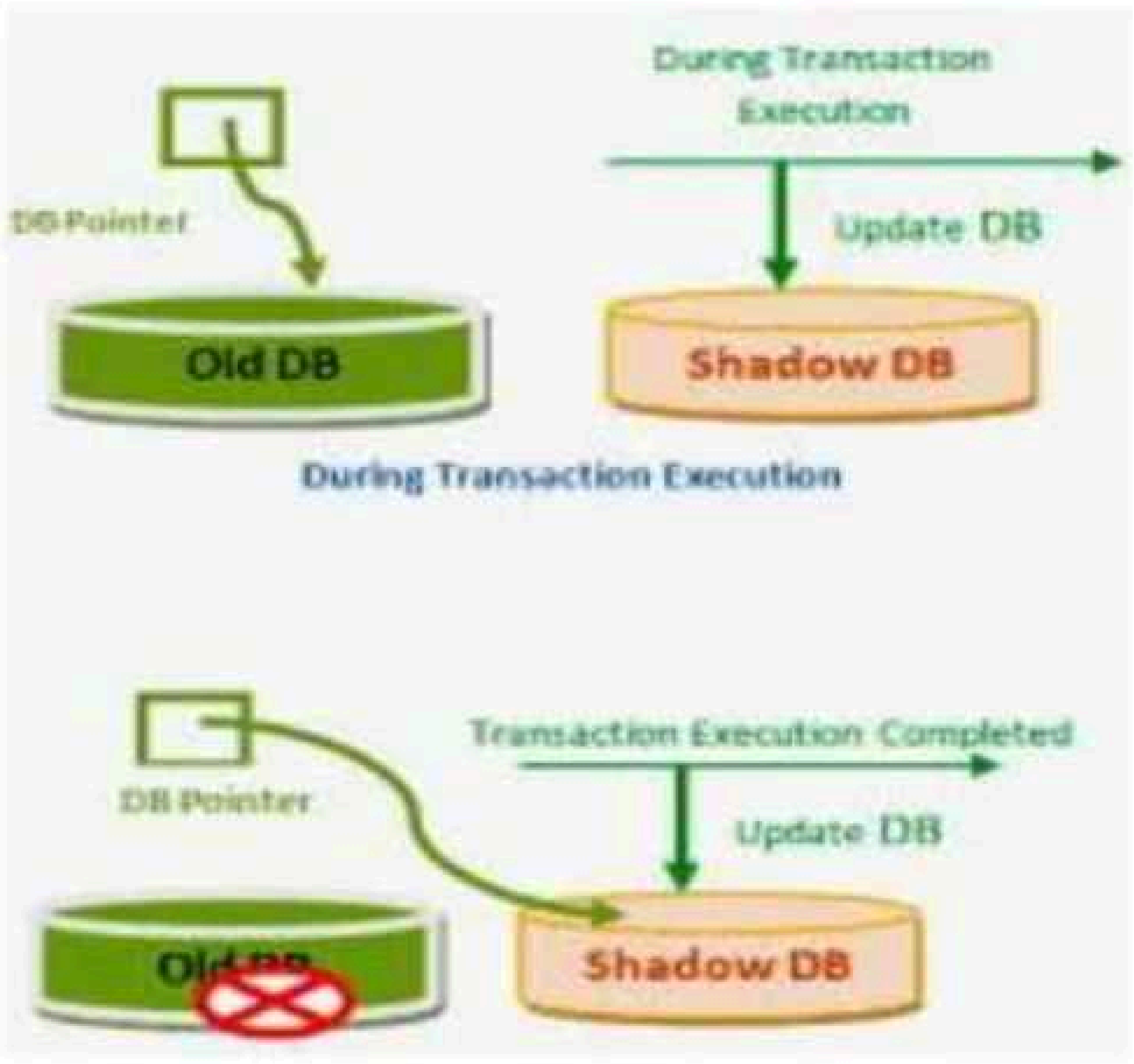
- The AFIM (After Image) does not overwrite its BFIM (Before Image) but recorded at another place on the disk.
- Thus, at any time a data item has AFIM and BFIM (Shadow copy of the data item) at two different places on the disk.

DBMS UNIT IV.pptx

Types of Failure

3. SHADOW PAGING: (Cont…)



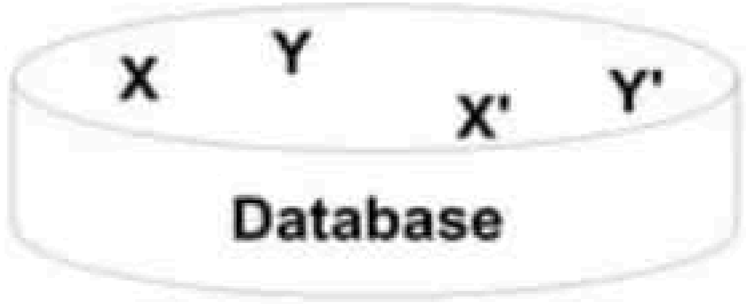


DBMS UNIT IV.pptx

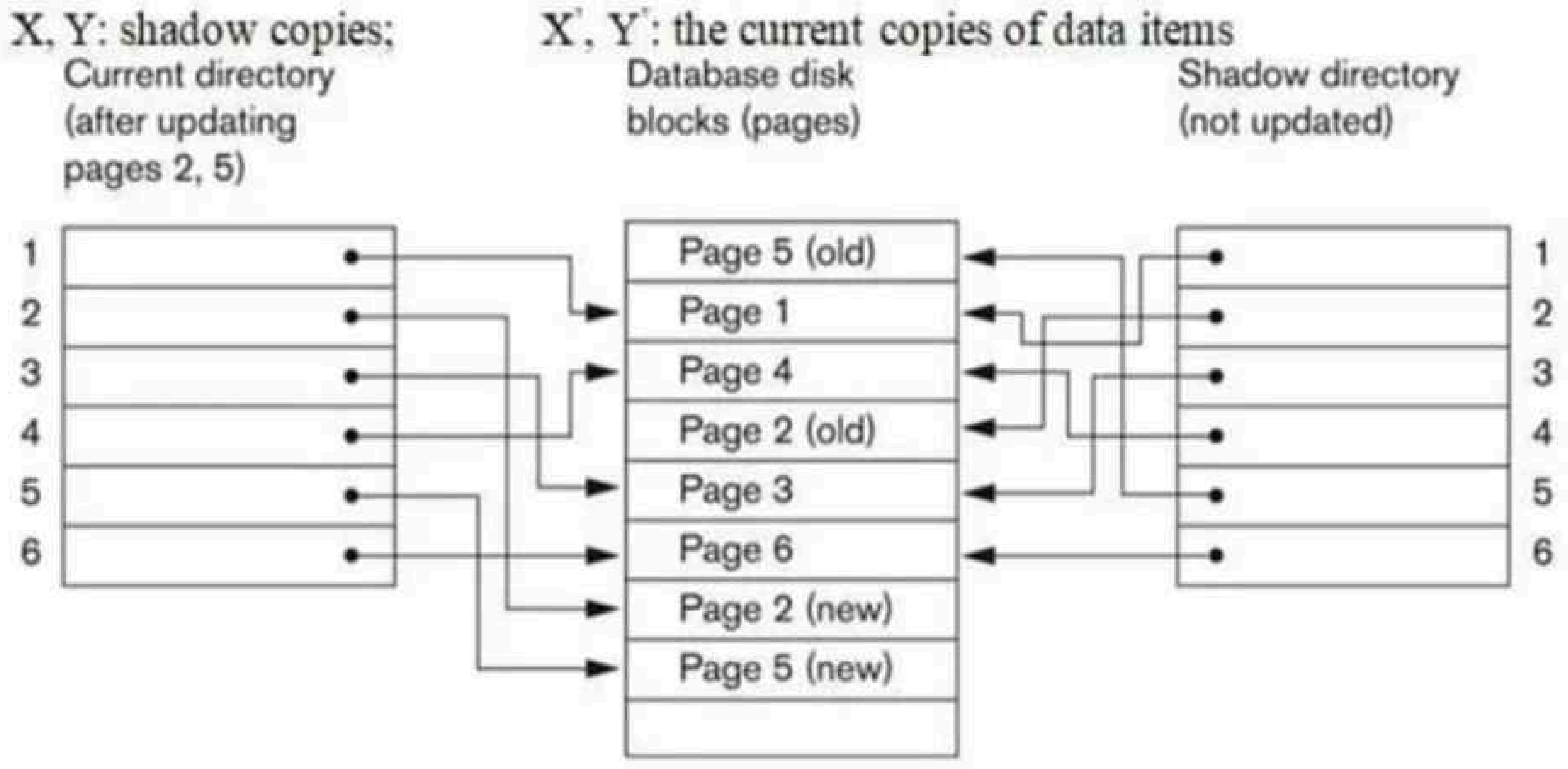
Janagi Raman S

Types of Failure

3. SHADOW PAGING: (Cont...)



X, Y: shadow copies; X', Y': the current copies of data items



DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

3. SHADOW PAGING:

- To recover from a failure, it is sufficient to free the modified pages and discard the current directory.
- The state of the database before transaction execution is available through the shadow directory.
- Database can be returned to its previous state that was executing when the crash occurred.
- We can categorized as a **NO-UNDO/NO-REDO** technique for recovery Logs and checkpoints must be incorporated into the shadow paging technique.

DBMS UNIT IV.pptx

Janagi Raman S

Types of Failure

3. SHADOW PAGING:

- Disadvantages:

  Complex storage management strategies, the overhead of writing shadow directories to disk, garbage collection overhead (old pages referenced by the shadow directory).