Functional Dependency

Functional Dependency (FD) is a relationship between the attributes of a table dependent on each other

S_ID	Sname	Age	DoB
111	Rajesh	30	15-5-91
112	Rahul	40	12-2-81
113	Amit	20	11-4-01

Student

$X \longrightarrow Y$

X → Determinant, Y → Dependent

$S_{ID} \longrightarrow (S_{name}, age, DOB)$

$X \longrightarrow Y$

Types of FD

Trivial FD

$X \longrightarrow Y$ (Y is subset of X)

$(S_{ID}, S_{name}) \longrightarrow S_{ID}$

$(S_{ID}, S_{name}) \longrightarrow S_{name}$

$S_{ID} \longrightarrow S_{ID}$

$S_{name} \longrightarrow S_{name}$

Non-trivial FD

$X \longrightarrow Y$ (Y is not subset of X)

$S_{ID} \longrightarrow S_{name}$

Transitive FD

$X \longrightarrow Y$

$Y \longrightarrow Z$

then

$X \longrightarrow Z$

$S_{ID} \longrightarrow S_{name}$

$Name \longrightarrow Age$

then

$S_{ID} \longrightarrow Age$

Partial FD

$X \longrightarrow Y$

$(S_{name}, Age) \longrightarrow DoB$

$Age \longrightarrow DoB$

Full FD

$X \longrightarrow Y$

$(S_{name}, Age) \longrightarrow DoB$

Inference rules of functional Dependency

Inference rules for FD

Reflexivity

If B is a Subset of A ($B \subset A$) then $A \rightarrow B$

Augmentation

If $A \rightarrow B$ then $AC \rightarrow BC$

Transitivity

{If $A \rightarrow B$ and $B \rightarrow C$} then $A \rightarrow C$

Additivity OR Union

{If $A \rightarrow B$ and $A \rightarrow C$} then $A \rightarrow BC$

Projectivity OR Decomposition

{If $A \rightarrow BC$} then $A \rightarrow B$ and $A \rightarrow C$

Pseudo Transitivity

{If $A \rightarrow B$ and $CB \rightarrow D$} then $AC \rightarrow D$

Armstrong's Axioms

Additional Rules

Closure of an attribute Set (F+)

Closure of an attribute Set

The Set of all those attributes which can be functionally determined from an attribute set(F) is called Closure of an attribute set. It is represented by F^+

Example

Consider a relation R(A, B, C, D, E, F, G) with the functional Dependencies-

$A \rightarrow BC$
$BC \rightarrow DE$
$D \rightarrow F$
$CF \rightarrow G$

(i) Find Closure of A, D and (B,C).

Ans (i) Closure of A

$(A^+) = \{A, B, C\}$ since $A \rightarrow BC$
$= \{A, B, C, D, E\}$ $BC \rightarrow DE$
$= \{A, B, C, D, E, F\}$ as $D \rightarrow F$
$A^+ = \{A, B, C, D, E, F, G\}$ as $CF \rightarrow G$

Closure of D

$(D^+) = \{D, F\}$ using $D \rightarrow F$

Closure of (B, C).

$(B, C)^+ = \{B, C, D, E\}$ since $BC \rightarrow DE$
$\{B, C, D, E, F\}$ since $D \rightarrow F$
$(B, C)^+ \{B, C, D, E, F, G\}$ since $CF \rightarrow G$

09/07/202

Finding Candidate key and Super key using FDs

Find the Super keys and Candidate keys in the given relational schema $R(X, Y, Z, \omega)$ and FD

$$XYZ \rightarrow \omega$$
$$XY \rightarrow Z\omega$$
$$X \rightarrow YZ\omega$$

Ans (i) Finding Candidate keys

Essential attributes are those attributes that are not on the right side of any FDs.

$\Rightarrow$ is the only attribute that are not on the right side of any FDs

So Essential attribute = X

(ii) Closure of essential attributes

$(X)^+ = \{X, Y, Z, \omega\}$ as $X \rightarrow YZ\omega$ since Closure of X include all

X is the only Candidate key

Finding Super keys

X — essential attribute

2^3 Comb of $(Y, Z\omega)$ — Non-essential attributes Combination { all Combination of Non-essential attributes }

Total Super keys = 8 possible

Super keys

X
X Y
X Z
X ω
X Y Z
X Y ω
X Z ω
X Y Z ω

Numerical to find Super keys and Candidate keys

the Candidate and Super key in relation R (X, Y, Z, W) having FDs

$X \rightarrow Y$
$Y \rightarrow Z$
$Z \rightarrow X$

Finding Candidate keys

Determine essential attributes

essential attributes = Not on the RHS of any FDs.
= W

essential attributes = W

Closure of essential attributes

$W)^+ = \{W\}$ since closure of W does not gives all attributes of relation R.
So Check Combination W with all other non-essential attributes.

$(WX)^+ = (W, X, Y)$ as $X \rightarrow Y$
(W, X, Y, Z) as $Y \rightarrow Z$

$(WY)^+ = (W, Y, Z)$ as $Y \rightarrow Z$
(W, Y, Z, X) as $Z \rightarrow X$

$(WZ)^+ = (W, Z, X)$ as $Z \rightarrow X$
(W, Z, X, Y) as $X \rightarrow Y$

as closure of WX, WY and WZ include all attribute of Relation R, So

Candidate keys = WX, WY, WZ

Super keys
1. WXY ✓
2. WXZ ✓
3. WXYZ ✓
4. WY X ✗
5. WYZ ✓
6. WYXZ ✗
7. WZX ✗
8. WZY ✗
9. WZXY ✗

Total Super keys = 4

Canonical Covers

Simplified and minimal version of the given set of functional dependencies

(1) for FD having two or more attributes on RHS, Use decomposition rule, so that RHS of a FD has only one attribute

e.g. $X \to YZ$ can be decomposed

$X \to Y$ ⌐ $A \to BC$

$X \to Z$ ⌙ $A \to B, A \to C$

(2) for FD having two or more attributes on LHS, determine closure (i) with FD Considering (ii) without FD Considering

(i) If closure is same in both cases, eliminate FD.

(ii) If closure is different in both cases, do not eliminate FD.

(3) Eliminate transitive FDs.

$X \to Y$] $X \to Z$

Example Find Canonical Cover for

$F = \{ A \to BC, B \to C, A \to B, AB \to C \}$

Ans step-1 $A \to BC$ can be decomposed to

$A \to B$
$A \to C$ }

Since there is same FD $A \to B$ also exist, so eliminate one of them

$A \to B, A \to C, B \to C, AB \to C$

step-2 (i) $(AB)^+$ Considering $AB \to C$

$(AB)^+ = \{A, B, C\}$

(ii) $(AB)^+$ non considering $AB \to C$ Same

$(AB)^+ = \{A, B, C\}$

So eliminate $AB \to C$

$A \to B, A \to C, B \to C$

step-3 from $A \to B$ & $B \to C \Rightarrow A \to C$, so eliminate $A \to C$

$A \to B, B \to C$ Ans

Normalization at a glance

Normalization

① Reduce Redundancy
② Remove Anomalies

- Un-normalized form (UNF)
- First Normal Form (1NF) — Remove Repeating groups
- Second Normal form (2NF) — Remove Partial dependency
- Third Normal form (3NF) — Remove Transitive dependency
- Boyce-Codd Normal form (BCNF) — Every determinant must be candidate key
- fourth Normal form (4NF) — Remove Multivalued dependency
- Fifth Normal form (5NF) — Remove Join dependency

Lossless Decomposition with Example

Lossless decomposition: The decomposition of relation R into relations R_1 and R_2 is said to be lossless (No loss of Information) if at least one of the functional dependencies is in F^+

$R_1 \cap R_2 \rightarrow R_1$
$R_1 \cap R_2 \rightarrow R_2$

where F = set of functional dependencies

Q Suppose we decompose schema $R=(A,B,C,D,E)$ into $R_1(A,B,C)$ and $R_2(A,D,E)$

Show that this decomposition is lossless if the following set of functional dependencies exist

$$F = \{ A \rightarrow BC,\ CD \rightarrow E,\ B \rightarrow D,\ E \rightarrow A \}$$

Ans for decomposition to be lossless

$R_1 \cap R_2 \rightarrow R_1$
or
$R_1 \cap R_2 \rightarrow R_2$
is in F^+,

$R_1 = (A,B,C)$
$R_2 = (A,D,E)$
$R_1 \cap R_2 = \{A\}$

Now take closure of A

$\{A\}^+ = \{A, B, C, D, E\}$

since it covers all attributes of relation $R_1(A,B,C)$ OR $R_2(A,D,E)$ so

i.e. Lossless decomposition

Numerical-3 Finding of Candidate keys in Relation R

For the Relation Schema R(A, B, C, D, E)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

{Determine Candidate keys (CK)}

Sol Step-1 Find essential attributes

Essential attributes = attributes not on the RHS of all FDs = ϕ (No attribute)

Non essential attributes = (A, B, C, D, E)

Step-2 Now first check each non-essential attribute 'alone' for determine the closure of closure contains all attributes of a relation, then it will be a candidate key

$(A)^+ = \{A, B, C, D, E\}$ So CK

$(B^+) = \{B, D\}$ X So Not CK X

$(C^+) = \{C\}$ X So Not CK

$(D^+) = \phi$ X So Not CK

Since Non-essential attribute B, C, D does not 'alone' form candidate key (CK). So try their Combinations

$(BC)^+ = \{B, C, D, E, A\}$ So CK

$(BD)^+ = \{B, D, \}$ X So not CK

$(CD)^+ = \{C, D, E, A, B\}$ So CK

So Candidate keys are = A, E, BC, CD

Normalization

Definition: Process to remove redundancy from tables

Goals: To improve
(1) Data Integrity (accuracy / consistency of data)
(2) Scalability (ability of a system to continue work with added task)
(3) Storage efficiency (ability to storage & manage data in min space)

Implementation: Normalization can be done by splitting (dividing) an existing table into more than one tables which can be re-joined each time a query is executed

No. of Normal forms: (1) 1NF (2) 2NF (3) 3NF (4) BCNF (5) 4NF (6) 5NF

(1) 1NF (First Normal form)

A relation R is said to be in first Normal form if and only if each cell of a table contain a single value (atomic value)

Example — Un-normalized

Roll No	Name	Subjects
101	Rahul	OS
102	Amit	DBMS, SE
103	Rajesh	CD
104	Arun	DBMS
105	Harsh	OS, SE

⇒ 1NF

Roll No	Name	Subjects
101	Rahul	OS
102	Amit	DBMS
102	Amit	SE
103	Rajesh	CD
104	Arun	DBMS
105	Harsh	OS
105	Harsh	SE

(A, B) → C fully

A ↛ C

B ↛ C

Second Normal form (2NF)

1. It must be in 1NF (each cell has single value)
2. All non-key attributes are fully functional dependent on primary key (No Partial dependency)

Example

Customer Id	Store Id	Purchase Location
1	1	Delhi
1	3	Ghaziabad
2	1	Delhi
3	2	Mumbai
4	3	Ghaziabad

Primary key = (Customer Id, Store Id)

Non-key attribute = Purchase location

1. (Customer Id, Store Id) → Purchase location
2. Store Id → Purchase location

(Partial dependency exist)

So not in 2NF

Customer Id	Store Id
1	1
1	3
2	1
3	2
4	3

Store Id	Purchase Location
1	Delhi
2	Mumbai
3	Ghaziabad

Now in 2 NF

BCNF (Boyce Codd Normal Form)

3NF

Student

Rollno	Name	Voter id	age
1	Ravi	K0123	20
2	Varun	M034	21
3	Ravi	K786	23
4	Rahul	D256	21

L.H.S of each FD should be CK
Candidate Key or Super Key

CK: { Rollno, Voter id }

FD:
- Rollno → name (1st FD)
- Rollno → Voter id
- Voter id → age
- Voter id → Rollno

Third Normal Form (3NF)

A relation is said to be in 3NF if

(a) If it is in 2NF and

(b) There is no transitive dependency.

(i.e. $A \to B$, $B \to C$, then $A \to C$)

FDs

- Book id → Genre id
- Genre id → Genre type
- Book id → Price

Primary key – Book id

Book ID	Genre id	Genre Type	Price (Rs)
1	1	Gardening	25/-
2	2	Sports	14/-
3	1	Gardening	12/-
4	3	Travel	17/-
5	2	Sports	15/-

⇓

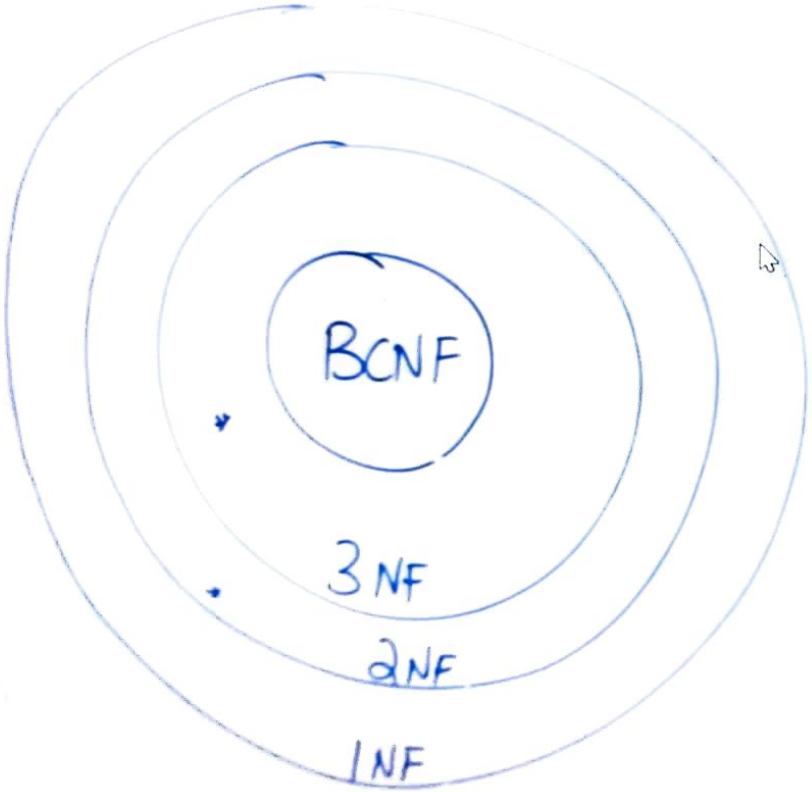
Transitive dependency exist

(So Not in 3NF)

Genre id	Genre type
1	Gardening
2	Sports
3	Travel

Book id	Genre id	Price
1	1	25/-
2	2	14/-
3	1	12/-
4	3	17/-
5	2	15/-

BCNF (Boyce



L.H.S of each FD sh
Candidate Key

CK: { Rollno, Voter

FD: { Rollno → V
Voterid Rollno →
Voterid →
Voterid →

Fourth Normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form (BCNF) and has no multi-valued dependency.
- MVD (multi-value dependency) occurs when two or more independent multi-valued facts about the same attribute occurs within the same relation.
- MVD is denoted by

$$X \rightarrow\rightarrow Y$$

It will be readers "there is a multi-valued dependency of Y" or "X multi-determines Y".

Example ⇒ Faculty

It will be readers "there is a multi-valued dependency of Y" or ~~X~~ multi-determines Y".

Example ⇒ Faculty

faculty	Subject	Committee
Kailash	DBMS	Placement
Kailash	Java	Placement
Kailash	C	Placement
Kailash	DBMS	Scholarship
Kailash	Java	Scholarship
Kailash	C	Scholarship

The given faculty table is in 3NF, but the Subject and Committee are two independent Entity. Hence there is no relationship between Subject and Committee.

In the faculty relation a faculty

Kailash	DBMS	Placement
Kailash	Java	Placement
Kailash	C	Placement
Kailash	DBMS	Scholarship
Kailash	Java	Scholarship
Kailash	C	Scholarship

The given faculty table is in 3NF, but the Subject and Committee are two independent Entity. Hence there is no relationship between Subject and Committee.

In the faculty relation, a faculty with faculty name Kailash Contains three Subject DBMS, Java and C, and two committee placement

dependency on faculty_name, which leads to unnecessary repetition of data.

Kailash → → Placement
Kailash → → Scholarship

So to make the above table into 4NF, we can decompose it into two tables:

Table 1 faculty_Course

faculty	Subject
Kailash	DBMS
Kailash	Java
Kailash	C

Table 2 faculty - Committee

faculty	Committee
Kailash	Placement
Kailash	Scholarship

09/07/20

Fifth Normal form (5NF)

- The 5NF (fifth Normal form) is also known as project-join Normal form.
- A relations is in fifth Normal form (5NF), if it is in 4NF, and won't have Lossless decomposition into Smaller tables.
- 5NF is satisfied when all the tables are broken into a many tables as possible in order to avoid redundancy After that you combined these all tables if it is Equal to original table then 5NF.
- You Can also Consider that a relation is in 5NF, if the Candidate key implies every join dependency in it.

Example ⇒ faculty (original table)

We can also consider that a relation is in 5NF, if the Candidate key implies every join dependency in it.

Example ⇒ faculty (original table)

faculty	Subject	Committee
Kailash	DBMS	Placement
Kailash	Java	placement
Kailash	C	placement
Kailash	DBMS	Scholarship
Kailash	Java	Scholarship
Kailash	C	Scholarship

the given table is not in 4NF and 5NF first we convert it in 4NF with converting it two sub table.

Table 1 faculty - Subject

faculty	Subject
Kailash	DBMS
Kailash	Java
Kailash	C

faculty	Committee
Kailash	Placement
Kailash	Scholarship

To Convert it in SNF, we join both table1 and table2 if it give the result same as original table (faculty) then it's in SNF otherwise not in SNF.

faculty	Subject	Committee
Kailash	DBMS	Placement
Kailash	DBMS	Scholarship
Kailash	Java	Placement
Kailash	Java	Scholarship
Kailash	C	Placement
Kailash	C	Scholarship

SNF

TRANSACTION PROCESSING CONCEPTS

TRANSACTION SYSTEM

- Collection of operations that form a single logical unit of work are called transaction.
- A trasaction is a unit of program execution that accesses and possibly updates various data items.
- Trasaction is define as a logical unit of database processing that includes one or more database access operations.

Transaction access data using two operations:

1) Read(X) :⇒ Read operation is used to

of database processing that includes one or more database access operations.

Transaction access data using two operations:

1) Read(X) :⇒ Read operation is used to read the value of Account x from the database and stores it in a buffer in main memory.

2) Write(X) :⇒ Write operation is used to write the value back to the database from the buffer.

Let's take an Example to debit trasaction from an account which Consists of following operations:

The second operation will decrease the value of X by 500. So buffer will Contain 3508.

- The third operation will write the buffer's value to the database. So x's final value will be 3500.

But it may be possible that because of the failure of hardware, Software or power, etc. that trasaction may fail before finished all the operations in the set.

for Example:⇒ If the given trasaction, the debit trasaction fails after Executing operation 2 then X's value will remain 400 in the database which is not acceptable by the bank.

To solve this problem, we have two important operations:

1) Commit:⇒ It is used to Save the work done permanently.

main memory.

2) Write(x) :⇒ Write operation is used to write the value back to the database from the buffer.

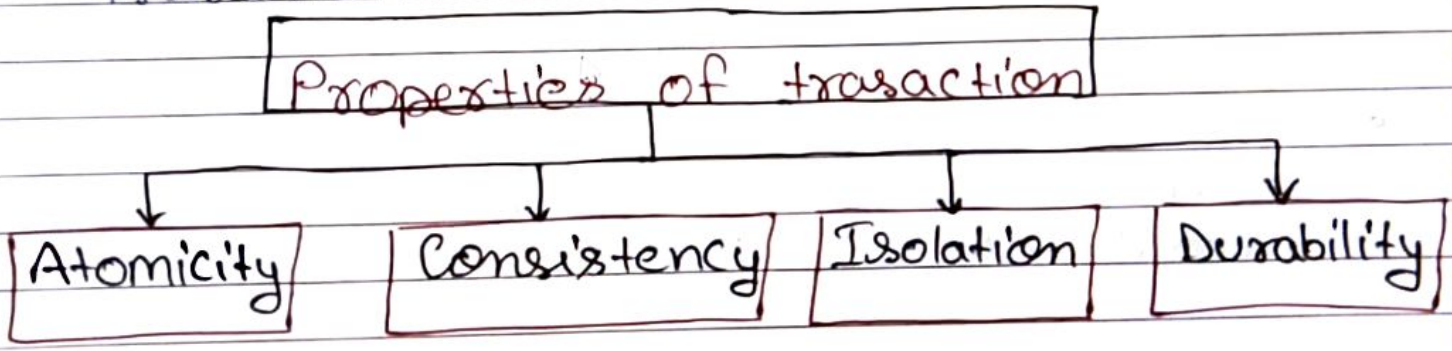
Let's take an Example to debit trasaction from an account which Consists of following operations: X=1000 Y=1000

1) R(X);
2) X = X-500;
3) W(X)

T_1	T_2
read(x)	read(Y)
X = X-500	Y = Y+500
write(x)	write(Y)
(X = 500)	Y = 1500

ACID properties of Transaction

Transactions have four basic properties. which are called ACID properties. These are used to maintain Consistency in a database, before and after the trasaction.



1) Atomicity ⇒ • It States that all operations of the transaction take place at once if not, the transaction is aborted.

Consistency | Isolation | Durability

1) Atomicity ⇒ • It States that all operations of the transaction take place at once if not, the transaction is aborted.
• There is no midway such as the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

Atomicity involves the following operations:-

Abort: If a transaction aborts then all the changes made are not visible.

Commit: If a transaction commit then all the changes made are visible.

Example ⇒ Let's assume that following transaction

Atomicity involves the following operations:-

Abort: If a transaction aborts then all the changes made are not visible.

Commit: If a transaction commit then all the changes made are visible.

Example: ⇒ Let's assume that following transaction T consisting of T_1 and T_2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from Account A to Account B.

2) Consistency ⇒ • The integrity constraints are maintained so that the database is consistent before and after the transaction.

- The execution of a transaction will leave a database in either its prior stable state or a new stable state.
- The consistency property of database states that every transaction sees a consistent database instance.
- The transaction is used to transform the database from one consistent state to another consistent state.

for Example ⇒ Let's assume that following transaction T Consisting of T1 and T2. A Consists of Rs 600 and B Consists of Rs 300. Transfer Rs 100 from Account A to Account B.

T1	T2
Read (A)	Read (B)
A = A - 100	B = B + 100
Write (A)	Write (B)

The total amount must be maintained before or after the transaction

Total before T occurs = 600 + 300 = 900
Total after T occurs = 500 + 400 = 900

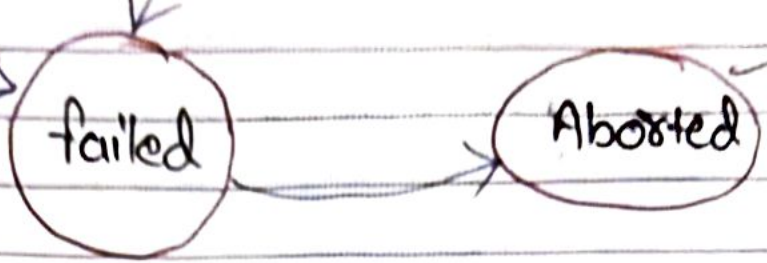
Therefore, the database is consistent. In the case when T_1 is completed but T2 fails, then inconsistency will occur.

3) Isolation ⇒ • It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.

- In isolation, if the transaction T_1 is being executed and using the data item x, then that data item can't be accessed by any other transaction T2 until the transaction T_1 ends.

- The Concurrency Control subsystem of the DBMS enfored the isolation property.

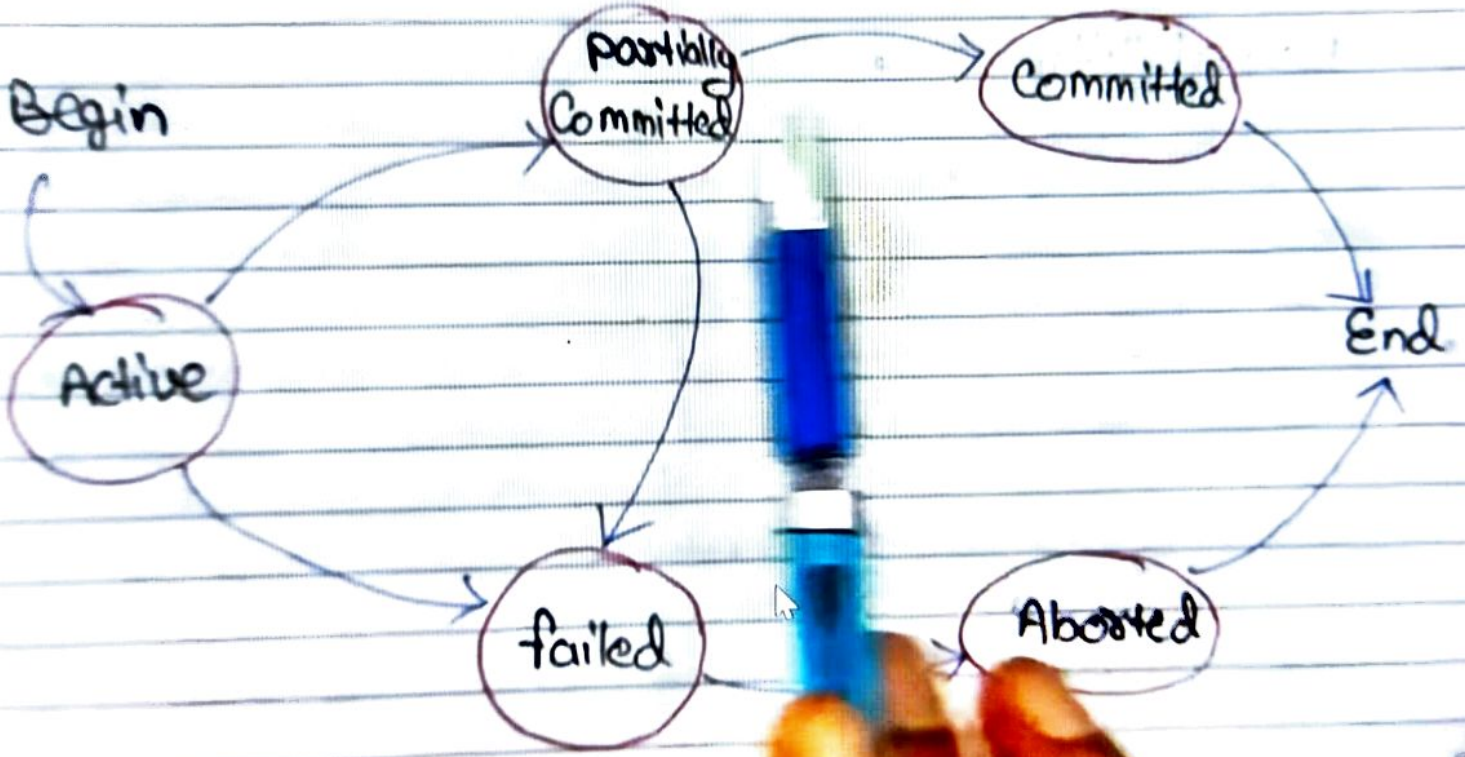
4) Durability ⇒ • The durability property is used to indicate the performance of the database's Consistent State. It States that the transaction made the permanent changes.
- They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. when a transaction is Completed, then the database reaches a state known as the Consistent state. that Consistent State cannot be lost, even in the Event of a system's failure.
- The recovery subsystem of the DBMS has the responsibility of Durability property.



1) Active State ⇒ • It is the first state of every transaction. In this state, the transaction is being executed.
• for Example ⇒ Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

2) Partially Committed ⇒ • In the partially Committed state, a transaction executes its final operation, but the data is still not saved to the database.
• In the total mark Calculation Example, a final display of the total marks step is executed in this state.

In a database, the transaction can be in one of the following states:-



1) Active State ⇒ • It is ... tate of ... 's state, Every tra... the transaction is ... Active

now permanently saved on the database System

4) failed State ⇒ • If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.

• In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

5) Aborted ⇒ • If any of the checks fail and the transaction has reached a failed state then the database recovery System will make sure that the database is in its previous consistent state. If not

3) Committed ⇒ • A transaction is said to be in a committed state if it executes all its operations successfully.
• In this state, all the effects are now permanently saved on the database System.

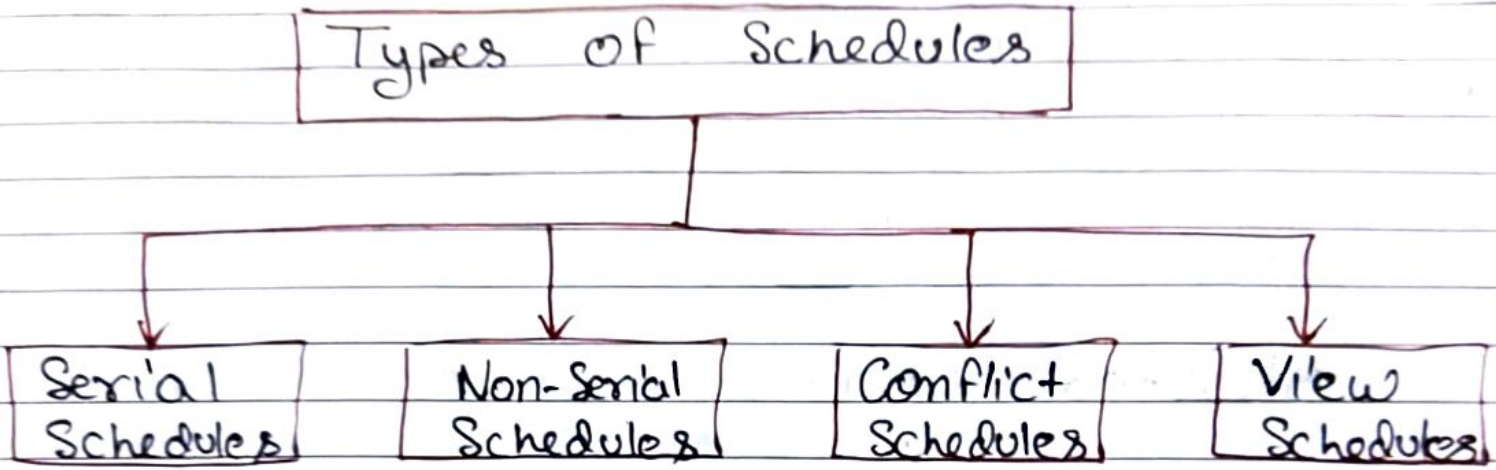
4) failed State ⇒ • If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
• In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction

if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

5) Aborted ⇒ • if any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous Consistent state. If not then it will abort or rollback the transaction to bring the database into a Consistent state.

• After aborting the transaction, the database recovery module will select one of the two operations:-

1) Re-Start the transaction



1) Serial Schedules ⇒ The Serial Schedule is a type of Schedule where one transaction is Executed Completely before Starting another transaction. In the Serial schedule, when the first transaction Completes its Cycle, then the next transaction is Executed.

T1	T2

Serializability of Schedules

Schedule ⇒ When several transaction are executing concurrently then the order of execution of various instruction is known as a Schedule.

The concept of serializability of schedules is used to identify which schedules are connected when transaction executions have interleaving of their operations in the schedules.

Types of Schedules

1) Serial Schedules ⇒ The Serial Schedule is a type of schedule where one transaction is executed completely before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

T_1	T_2
read_item(X);	
X = X − N;	
write_item(X);	
read_item(Y);	

```
Y = Y+N;
write_item(Y);
```

```
read_item(X);
X = X+m;
write_item(X);
```

Schedule A

The Schedules A is called serial scheduler. entire transaction are perform in serial order T1 and then T2 or T2 and Then T1.

2) Non-Serial Schedules ⇒ • if interleaving of operations is allowed, then there will be non-serial schedule.
• A Schedules are called non-serial

2) Non-Serial Schedules ⇒ • if interleaving of operations is allowed. Then there will be non-serial schedule.

• A Schedules are called non-serial Schedule if the operations of each transaction are executed non-consecutively with interleaved operations from the other transaction.

T_1	T_2
read_item(X);	
X = X + N;	
	read_item(X);
	X = X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y = Y + N;	
write_item(Y);	

3) Conflict Schedules ⇒ • A Schedules is Called Conflict Serializablity if after Swapping of non-Conflicting operations, it can transform into a Serial Schedule.

- The Schedule will be a Conflict Serializable if it is Conflict Equivalent to a Serial Schedule.

Conflicting Operations

The two operations become Conflicting if all Conditions Satisfy:

1. Both belong to Separate transactions.
2. They have the Same data item
3. They Contain at least one write operation.

Example ⇒ 1) Swapping is possible only if S1 and S2 are logically Equal

Example ⇒ 1) Swapping is possible only if S1 and S2 are logically Equal

T_1	T_2
Read(A)	
	Read(A)

Schedule S1

Swapped →

T_1	T_2
~~Read(A)~~	Read(A)
Read(A)	

Schedule S2

Here S1 = S2 that means it is non-conflict.

2)

T	T_2
Read(A)	
	Write(A)

Swapped ⟹

T_1	T_2
	Write(A)
Read(A)	

Here $S_1 \neq S_2$ that means it is Conflict.

4) View Serializability / Schedule ⇒ • A Schedule will view Serializable if it is View Equivalent to a serial Schedule.

- if a schedule is Conflict Serializable, then it will be view Serializable.
- The View Serializable which does not Conflict Serializable Contains blind writes.

Two Schedules S1 and S2 are said to be view Equivalent if they Satisfy the following Conditions:-

1) Initial Read ⇒ An initial read of both Schedules must be the same. Suppose two Schedule S1 and S2. In Schedule S1, if a transaction T1 is reading the data item

Conflict Serializable Contains blind writes.

Two Schedules S1 and S2 are Said to be view Equivalent if they Satisfy the following Conditions:-

1) Initial Read ⇒ An initial read of both Schedules must be the same. Suppose two Schedule S1 and S2. In Schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T_1	T_2
read(A)	
	write(A)

Schedule S1

T_1	T_2
	write(A)
Read(A)	

Schedule S2

Above two Schedules are view Equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

by T2 and in S2, T3 is reading A updated by T1.

3) final write ⇒ A final write must be the same between both the Schedules.
In Schedule S1, if a transaction T1 update A at last then in S2, final writes operations should also be done by T1.

T_1	T_2	T_3
Write(A)		
	Read(A)	
		write(A)

T_1	T_2	T_3
	Read(A)	
Write(A)		
		Write(A)

Above two Schedules is view Equal because final write Operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

two schedule S1 and S2. In Schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A

T_1	T_2
read(A)	
	write(A)

Schedule S1

T_1	T_2
	write(A)
Read(A)	

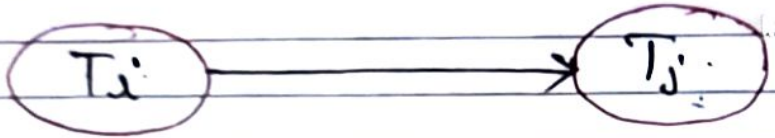
Schedule S2

Above two Schedules are view Equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

2) Update Read ⇒ In Schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated

2. Create a node $T_i \rightarrow T_j$ if T_i executes write(Q) before T_j executes write(Q).
3. Create a node $T_i \rightarrow T_j$ if T_i executes write(Q) before T_j execute write(Q).

Precedence graph for schedule S



- A precedence graph contains a single edge $T_i \rightarrow T_j$, then all the instructions of T_i are executed before the first instruction of T_j is executed.

Testing of Serializability

Serialization Graph is used to test the Serializability of a schedule.
for testing of serializability the simple and effecient method is to Construct a directed graph, Called a precedence graph from S.

$$G_1 = (V, E)$$

Where V Consists a Set of vertices and E Consists a set of edges. The set of vertices is used to Contain all the transactions participating in the Schedule.
The Set of edges is used to Contain all edges $T_i \rightarrow T_j$ for which one of the three conditions holds:

1: Create a node $T_i \rightarrow T_j$ if T_i executes write (Q) before T_j executes read (Q).
2: Create a node $T_i \rightarrow T_j$ if T_i executes read (Q) before T_j Executes write (Q).
3: Create a node $T_i \rightarrow T_j$ if T_i executes write (Q) before T_j execute write (Q).

- A precedence graph for Schedule S Contain a Cycle, then S is non-serializable. If the precedence graph has no Cycle, the S is known as Serializable.

for Example ⇒

T_1	T_2	T_3
Read(A)		
	Read(B)	
$A := f_1(A)$		
		Read(C)
	$B := f_2(B)$	
	Write(B)	
		$C := f_3(C)$
		Write(C)
Write(A)		
		Read(B)
	Read(A)	
	$A := f_4(A)$	
Read(C)		
	Write(A)	
$C := f_5(C)$		
Write(C)		
		$B := f_6(B)$

	A: t_4(A)	
Read(C)		
	Write(A)	
C: t_5(C)		
Write(C)		
		B: t_6(B)
		Write(B)

Schedule SL

Explanation:

Read(A): In T_1, no subsequent writes to A, so no new edges.
Read(B): In T_2, no subsequent writes to B, so no new edges.
Read(C): In T_3, no subsequent write to C, so no new edges.

write(B): B is subsequently read by T_3, so add edge $T_2 \rightarrow T_3$.

write(C): C is subsequently read by T_1, so add edge $T_3 \rightarrow T_1$.

write(A): A is subsequently read by T_2, so add edge $T_1 \rightarrow T_2$

write(A): In T_2, no subsequent reads to A, so no new edges

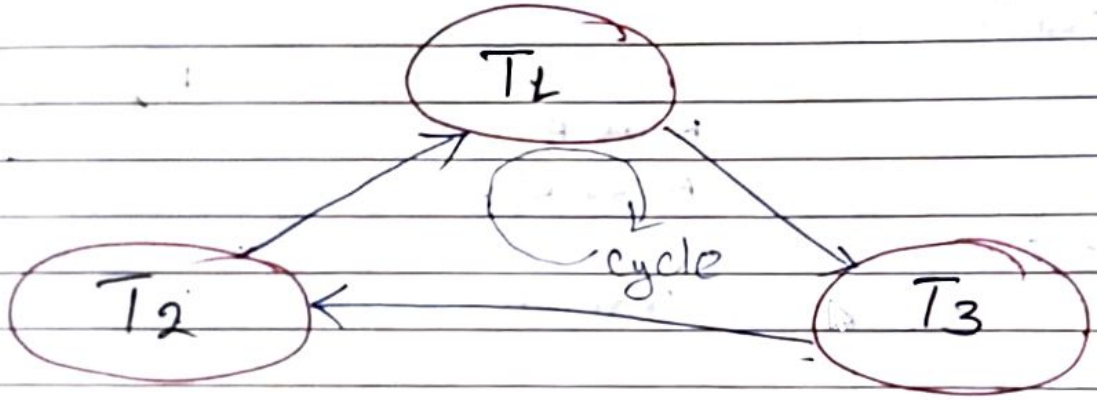
write(C): In T_1, no subsequent read to C, so no new edges.

write(B): In T_3, no subsequent read to B, so no new edges.

no new edges.

Write (B): In T3, no subsequent read to B, so no new edges.

Precedence graph for Schedule S1:



The precedence graph for Schedule S1 contains a cycle that's why Schedule S1 is non-serializable.

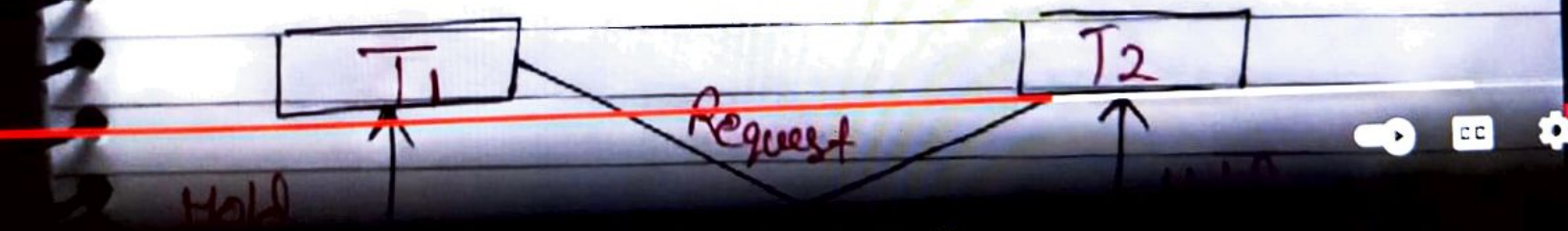
- A deadLock is a condition where two or transactions are waiting indefinitely for another to give up locks.
- A System is said to be in deadLock state when in a set of transactions, every tra is waiting for another transaction to its operations.
- A deadLock is a state of statement that result when two or more transactions Each waiting for locks held by the oth to be released.

for Example ⇒ In the Student table transact holds a lock on some r need to update some rows in th Simultaneously, transaction T2

to be released.

for Example ⇒ In the Student table, transactio holds a lock on some rows and need to update some rows in the grade table. Simultaneously, transaction T2 hold Locks o some rows in the grade table and need update the rows in the Student table he by transaction T1.

Now the main problem arises. Transactio is waiting for T2 to release its Locks and Similarly transaction Ta is waiting for T1 release its Lock: All activities come to halt state and remain at a standstill.



ample | Database Management System Tutorials in Hindi

810 Share Download ...

All From the series From CS

Difference b

ENG

Conflict Serializability

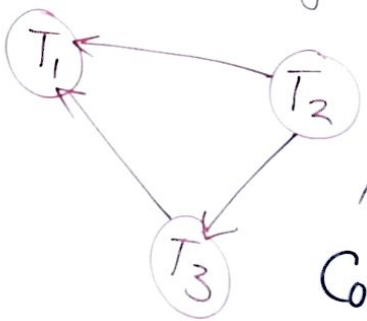
R-W
W-R
W-W

S

T_1	T_2	T_3
~~R(x)~~		
		~~R(y)~~
		~~R(x)~~
	~~R(y)~~	
	~~R(z)~~	
		~~w(y)~~ ← R(y) w(y)
	~~w(z)~~	
z)		
(x)		
w(z)		

- Check Conflict pairs in other transactions and draw edges

Precedence graph



Loop / Cycle

No loop / cycle

Conflict Serializable

↓

Serializable

TRANSACTION & CONCURRENCY CONTROL :

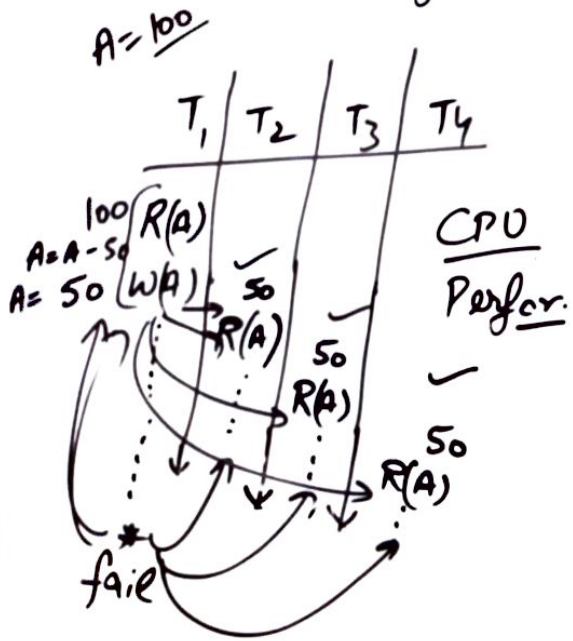
RECOVERABILITY OF SCHEDULE

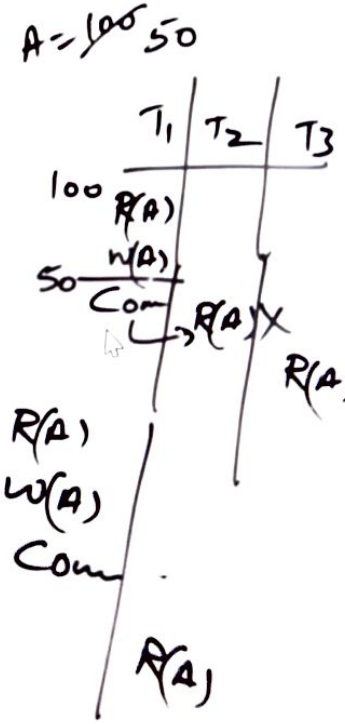
IF THERE IS NO. OF TRANSACTIONS IN A SCHEDULE & ONE TRANSACTION IS READING SOME OTHERS UPDATED INSTRUCTION & AFTER THAT IF THERE IS AN FAILURE OR ERROR THEN, IF WE ARE ABLE TO RETURN IN CONSISTENT STATE THEN IT IS RECOVERABLE SCHEDULE OTHERWISE IRRECOVERABLE SCHEDULE

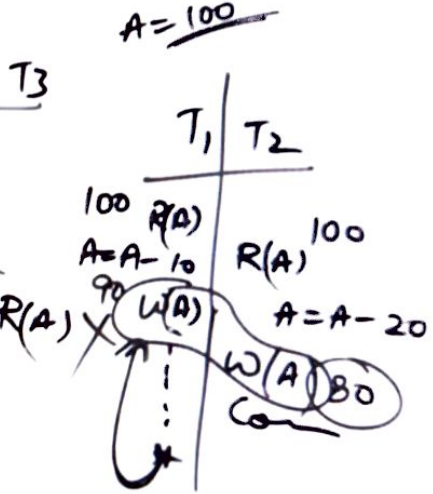
T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
COMMIT	
	COMMIT

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
	COMMIT
FAILURE	

Cascading Schedule Vs. Cascadeless Schedule







(1) Log based recovery -

The log is a sequence of records which maintains the records of all the update activities performed by a transaction.

It is maintained during the normal transaction processing, it contains enough information to recover from failures.

Types of log records that are maintained in a log -

1) Start record - The log record [Ti, start] is used to indicate that the transaction T_i has started

Recovery from transaction failures

The purpose of database recovery is to bring the database into last consistent state, which existed before the failure.

To recover from transaction failures, Atomicity of transactions as a whole must be maintained that is either all the operations are executed or none.

There are two types of techniques, which can help a DBMS, in recovering & maintaining the atomicity of a transaction :-

(1) Log based recovery

(2) Shadow paging.

performed by a transaction.

It is maintained during the normal transaction processing it contains enough information to recover from failures.

Types of log records that are maintained in a log -

1) **Start record** - The log record [Ti, start] is used to indicate that the transaction T_i has started

2) **Updated log record** - The log record

$[T_i, x, V_1, V_2]$

is used to indicate that the transaction T_i has performed an update operation on the data item x, having old value V_1 & new value V_2.

3) **Read record** - The log record

$[T_i, x]$

is used to indicate that the transaction T_i has read the data item x from the database.

4) **Commit record** - The log record

$[T_i, \text{commit}]$ is used to indicate that the transaction T_i has committed

5) **Abort record** – The log record

$[T_i, \text{abort}]$

is used to indicate that the transaction T_i has aborted & needs to be rollback.

Example

A = 1000

B = 2000

C = 700

T_1	T_2
read(A)	read(C)
A = A - 50	C = C - 100
write(A)	write(C)
read(B)	
B = B + 50	
write(B)	

log record

[T_0, start] [T_1, start]

[T_0, A]

[T_0, A, 950] [T_1, A]

[T_0, B, 2050] [T_1, A, 1000, 950]

[T_1, B]

[T_1, B, 2000, 2050]

[T_1, commit]

[T_2, start]

[T_2, C]

[T_2, C, 700, 600]

[T_2, commit]

RECOVERY SYSTEM :- TYPES OF FAILURES

#1) TRANSACTION FAILURE :- DUE TO SOME REASONS TRANSACTIONS GET FAIL, SO SOME REASONS ARE

A) LOGICAL ERRORS :- BAD INPUTS, DATA NOT FOUND, OVERFLOW, RESOURCE LIMIT EXCEEDED.

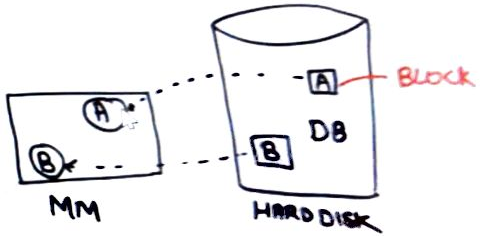
B) SYSTEM ERRORS :- DEDALOCK

$\frac{3}{0} = \infty$

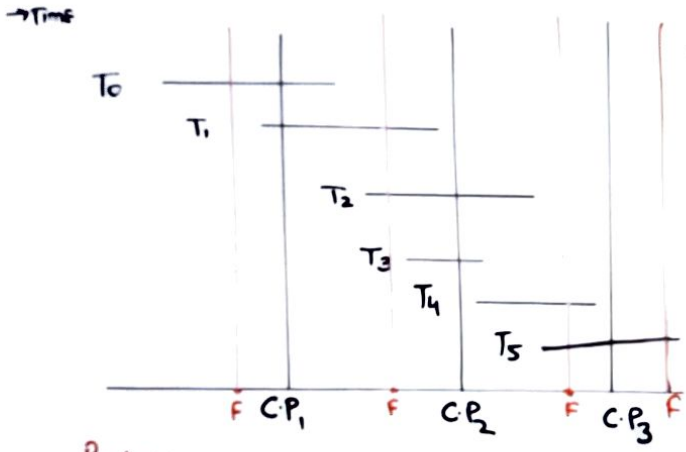
int : 268

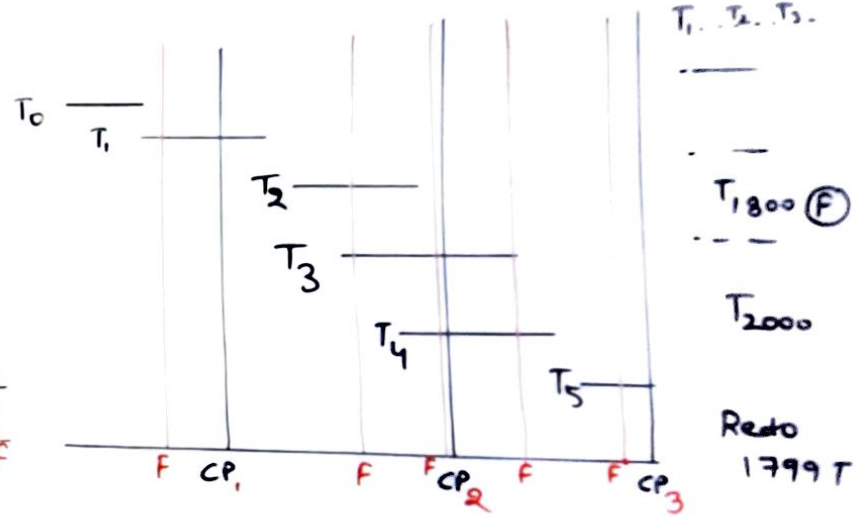
#2) SYSTEM CRASH :- H/W MALFUNCTION, BUG IN DATABASE SOFTWARE OR OPERATING SYSTEM.

#3) DISK FAILURE :- A DISK BLOCK LOSSES ITS CONTENT DURING DATA TRANSFER OPERATIONS. SO COPIES OF DATA ARE STORED / BACKUP.



RECOVERY SYSTEM : CHECK POINTS





Redo list

$\langle T_0, T_1 \rangle$

Undo list

$\langle T_1 \text{ START} \rangle$

$\langle T_1 \text{ COMMIT} \rangle$

$\langle T_1 \text{ Checkpoint}_2 \rangle$

Transaction & Concurrency Control : Why Concurrency Control Is Needed ?

2:30

Concurrency Leads To Several Problems

- # Lost Update Problem
- # Dirty Read Problem
- # Uncommitted Dependency Problem
- # Unrepeatable Read Problem
- # Inconsistent Analysis Problem
- # Phantom Read Problem.

#1) Lost Update Problem :- If any transaction T_j updates any variable 'A' at time 't' without knowing the value of 'A' at time 't' then this may leads to lost update problem.

T_1		T_2
R(A)	t_1	
A=A-500	t_2	
	t_3	Read A
	t_4	A = A+700
Write(A)	t_5	
	t_6	Write(A)

A=2500

T_1	T_2
2500	2500
A=2000	2500+700 = 3200
A= (W) A=2000	A= 3200
2000+700=2700	

Transaction & Concurrency Control : Advantages Of Concurrency

#1) **Resource Utilization**: If One Transaction Is Busy In Using CPU Then Other Can Use To Devices.

Improved Throughput :- More No. Transaction Is Completed/Executed Due To Concurrency.

#3) **Reduced Response Time** :- Less Time Is Needed For Completing Transaction.

#4) **Reduced Waiting Time**: Transaction Has To Wait Less For Getting Resource

Time	T_1	T_2	T_3
t_1		R	
t_2	R		
t_3			W
t_4		R	
t_5		R	
t_6		W	
t_7	W		
t_8	R		
t_9			R

Serial:- $T_2 \longrightarrow T_1 \longrightarrow T_3$

Transaction & Concurrency Control: Why Concurrency Control Is Needed ?

Concurrency Leads To Several Problems

- Lost Update Problem
- Dirty Read Problem
- Uncommitted Dependency Problem
- Unrepeatable Read Problem
- Inconsistent Analysis Problem
- Phantom Read Problem.

T_1 A = 600 - 200
= 400

T_2 : A = 400 + 600
= 1000

A = 1000

T_1 = A = 600

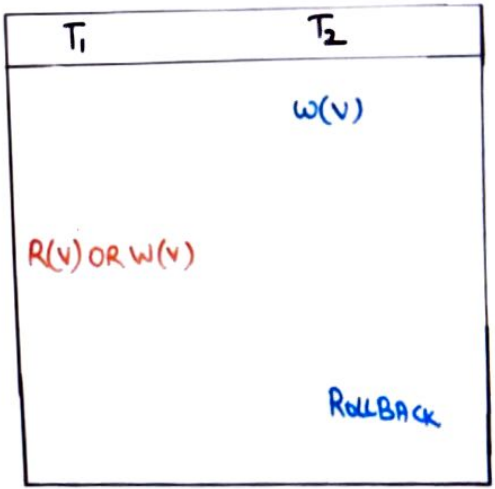
T_1	T_2
R(A)	
A = A - 200	
W(A)	R(A) → DRP
	A = A + 600
	W(A)
R(B)	
Failure (Rollback) Error	

#2) Dirty Read Problem :- Reading The Data Value Of A Variable Before Commiting May Lead To 'Dirty Read Problem'.

2.00 TRANSACTION & CONCURRENCY CONTROL : WHY CONCURRENCY CONTROL IS NEEDED ?

CONCURRENCY LEADS TO SEVERAL PROBLEMS

LOST UPDATE PROBLEM
DIRTY READ PROBLEM
UNCOMMITTED DEPENDENCY PROBLEM
UNREPEATABLE READ PROBLEM
INCONSISTENT ANALYSIS PROBLEM
PHANTOM READ PROBLEM.



#3) THE UNCOMMITTED DEPENDENCY PROBLEM:-

THIS PROBLEM ARISES IF ANY TRANSACTION Tj UPDATES ANY VARIABLE 'V' & ALLOWS R OR W OPERATION ON 'V' BY ANY OTHER TRANSACTION BUT Tj ROLLED BACK DUE TO FAILURE.

WHY CONCURRENCY CONTROL IS NEEDED ?

CONCURRENCY LEADS TO SEVERAL PROBLEMS

- # LOST UPDATE PROBLEM
- # DIRTY READ PROBLEM
- # UNCOMMITTED DEPENDENCY PROBLEM
- # UNREPEATABLE READ PROBLEM
- # INCONSISTENT ANALYSIS PROBLEM
- # PHANTOM READ PROBLEM.

A = 600 + 200 = 800
B = 700 + 800
= 1500
B = 1500

T_1	T_2
R(A) 600	
	R(A)
	A = A + 200
	W(A)
R(A) 800	
R(B)	
B = B + A	
W(B)	

#4) UNREPEATABLE READ PROBLEM :-

IT OCCURS WHEN A TRANSACTION TRIES TO READ THE VALUE OF DATA ITEM TWICE, & ANOTHER TRANSACTION UPDATE SAME DATA IN ITEM IN BW THE TWO READ OPERATIONS OF FIRST TRANSACTION

Transaction & Concurrency Control : Why Concurrency Control Is Needed ?

Concurrency Leads To Several Problems

- Lost Update Problem
- Dirty Read Problem
- Uncommitted Dependency Problem
- Unrepeatable Read Problem
- Inconsistent Analysis Problem
- Phantom Read Problem.

#5) Inconsistent Analysis Problem :-

One Transaction Read Variable But Before Committed By Another Transaction

T_1	T_2
R(a) a=10	t_1
R(b) b=20	t_2
	t_3 W(a) a=50
	t_4 Commit
Add(a+b)	t_5
10+20=30 NOT 70 Inconsistent	

2:06 / 2:39

7.9- Inconsistent Analysis Problem In DBMS | Need Of Concurrency Control in DBMS | Complete Syllabu

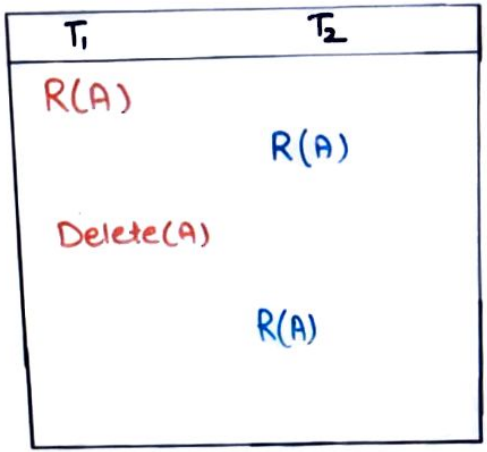
TutorialsSpace- Er. Deepak G... 113K subscribers Join Subscribe 115 Share Download

09/07/2024 17:57

Transaction & Concurrency Control : Why Concurrency Control Is Needed ?

Concurrency Leads To Several Problems

- Lost Update Problem
- Dirty Read Problem
- Uncommitted Dependency Problem
- Unrepeatable Read Problem
- Inconsistent Analysis Problem
- Phantom Read Problem.



#6) Phantom Read Problem:-

This Problem Occurs When Transaction Reads Same Variable From The Buffer & It Reads It Again Later, If Finds It Doesn't Exits.

Concurrency Control Techniques :-

To avoid problems related to concurrency we have to apply some protocols (Rules) on transactions.

Different Concurrency Control Techniques

#1) Lock Based Protocols
#2) Time Stamped Based Protocols
#3) Vaudation Based Protocols
#4) Multinersion Concurrency Control Protocols.
#5) Graph Based.

Lock Based Protocols

Granularity of Locks
Types of Locks
Types of Lock Based Protocols

1) Locking Protocol
2) 2 Phase Locking Protocols & Their Types.

Concurrency Control Techniques :- Lock Based Protocols :- Granularity of Locks

The level & type of information that the lock protects is called locking granularity. OR In other words, the size of data items that the data manager locks is called locking granularity.

Locking can occur on following levels:-

A) Coarse Granularity

Locks on Table / Files / Database

It reduces no. of transaction to be executed ↓ Through Put ↑ Response Time

↓ Low Concurrency

B) Fine Granularity :- Locks on Records / Fields

Much lock overhead, ↑ High Concurrency

↑ Through Put ↓ Response Time

C) Intermediate Granularity :- Locks on Pages which are of fixed sizes like 4KB, 8KB, 16KB...

A table may span into several pages & a page may contains several rows of a table or more tables.

It gives moderate concurrency & moderate lock overhead

Concurrency Control Techniques : Lock Based Protocols : Types Of Locks

There are 2 types of Lock :-

(i) Shared Locks - If T_i has obtained shared lock on data item A then T_i can read A but cannot modify A. It is denoted by 'S' & T_i is said to be in shared lock mode.

(ii) Exclusive Lock :- It is read-write lock. If T_i having this lock on A then it can do both read & write. It is denoted by X.

Compatible Lock Modes

T_1 \ T_2	S	X
S	Possible	Not Possible
X	Not Possible	Not Possible

S-Lock(A) — Shared Lock on A

X-Lock(A) — Exclusive Lock on A

Unlock(A) — Free A

Concurrency Control Techniques :- Lock Based Protocols :-

Two Phase Locking

In this every Transaction Issue Lock & Unlock Request In 2 Phases

Growing Phase :- Only obtain new locks.

Shrinking or Contracting Phase :- Release Lock only.

Lock Point :- When It obtained final lock.

#1) Basic Two Phase Locking Protocol

Advantages :- Ensure Serializability

Disadvantages :- Cascading Rollback.

T_1	T_2
X-Lock(A)	S-Lock(A)
R(A)	R(A)
A=A-500;	S-Lock(B)
W(A)	R(B)
X-Lock(B)	Display (A+B)
R(B)	Unlock(A)
B=B+500	Unlock(B)
W(B)	
Unlock(A)	
Unlock(B)	

Improvement In Locking

Concurrency Control Techniques :- Lock Based Protocols - Locking

1) Locking

Advantages - It maintains Serializability

- Solve Concurrency Problem

Disadvantages - It leads to new problem ie Deadlock

A = 1000
B = 300

T_1	T_2
X-Lock(A)	
R(A) A=1000	
A=A-500	
W(A) A=500	
Unlock(A)	
	S-Lock(A)
	R(A) A=500
	Unlock(A)
	S-Lock(B)
	R(B) 300
	Unlock(B)
	Display(A+B) 800
X-Lock(B)	
R(B) 300	
B=B+500	
W(B) 800	
Unlock(B)	

T_1	T_2
X-Lock(A)	S-Lock(A)
R(A)	R(A)
A=A-500;	S-Lock(B)
W(A)	R(B)
X-Lock(B)	Display (A+B)
R(A)	Unlock(A)
B=B+500	Unlock(B)
W(B)	
Unlock(A)	
Unlock(B)	

Improvement In Locking

Concurrency Control Techniques :- Lock Based Protocols :-

Two Phase Locking 2PL

#2) **Strict Two Phase Locking Protocol** : All X-lock are kept until the end of transaction or it committed.

Advantage :- No cascading Roll back

#3) **Rigorous Two-Phase Locking Protocol** :- All locks will be kept until transaction commit.

Lock Conversion :- Two improve the effiency of 2-PL, modes of lock can be converted.

Upgrade :- S lock → X-lock

Down Grade :- X-lock → S-lock

T_1	T_2
X-Lock(A)	
R(A)	
A=A-500;	
W(A)	
X-Lock(B)	
R(A)	
B=B+500	
W(B)	
COMMIT	
UNLOCK(A)	
UNLOCK(B)	
	X-Lock(A)
	X-Lock(B)
	=

Concurrency Control Techniques :- Time Stamp-Based Protocol- Introduction

Used To Order Conflict Transactions (Those Transactions Which Need Same Data Items For Execution)

Time Stamp Or Unique No. Is Assigned To Every Ti Transaction Before It Starts Execution Which Is Denoted By TS (Ti).

For Any Two Transaction Ti & Tj $TS(T_i) > TS(T_j)$ OR $TS(T_i) < TS(T_j)$, Not =

Time Stamp Can Be Given By Using

i) System Clock
ii) Logical Counter

Two Time Stamps Are Associated With Each Data Item 'A' Of Database Are

#1) Write-TS(A): It Is Equal To Time Stamp Of Most Recent Transaction That Execute W(A) Successfully.

#2) Read-TS(A): It Is Equal To Time Stamp Of Most Recent Transaction That Executed R(A)

T_1	T_2	T_3
W(A)		
	R(A)	
		W-3(A)
	W(A)	
	Read-2(A)	

Concurrency Control Techniques :- Time Stamp-Based Protocol- Ordering

We Order Conflicting Read & Write Operations By Using Following Rules :-

1) Any T_i, To Execute R(A)

i) If $Ts(T_i) < W\text{-}TS(A)$ Then R(A) Is Rejected & Rollback, As T_i Tries To Read Value Of A Which Is Overwritten By Any Other T_j

R-W
W-W
W-R

$TS(T_i)=6$	$Ts(T_j)=9$
	W(A)
R(A)	

ii) If $Ts(T_i) \geq W\text{-}TS(A)$, Then R(A) Is Executed & Read-TS(A) Is Set To Time Stamp Of Max. $TS(T_i)$

$Ts(T_j)=29$	$Ts(T_i)=65$	$Ts(T_i)=65$
W(A)		W(A)
	R(A)	R(A)

2) Any T_i To Execute W(A)

i) If $Ts(T_i) < Read\text{-}TS(A)$ Then Reject W(A) & Rollback Bic A Varible Is Previously Used By Another Transaction. Roll Back Avoid Inconsistent Analysis Problem.

$Ts(T_i)=7$	$Ts(T_j)=9$
	R(A)
W(A)	

ii) If $Ts(T_i) < Write\text{-}TS(A)$, W(A) Will Be Rejected & Rollback As T_i Tries To Write Obsolete Value Of A. It Avoids Lost Update Problem.

$Ts(T_i)=11$	$Ts(T_j)=16$
	W(A)
W(A)	

iii) Otherwise Execute W-TS(A)

Concurrency Control Techniques :- Validation Based Protocols :- Introduction

- Used where transactions are read only & conflicts are low.
- Every T_i has to go through the following phases:

#1) **Read Phase** :- In this phase T_i reads all data & store them in temporary variables (local variables of T_i). After reading all the write operations are made on temporary variables instead of actual DB.

#2) **Validation Phase** :- In this validation test is performed to determine whether changes in actual database can be made.

#3) **Write Phase** :- If T_i clear the validation test then actual changes are made to DB.

Validation test ensures the violation free execution of transaction.

Timestamp is used to determine when to start validation test.

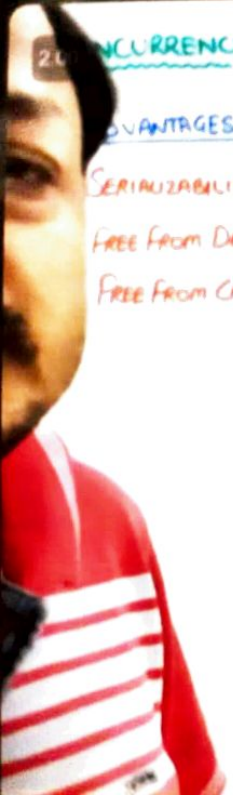
Every T_i is associated with three time stamps which are

Start (T_i) :- It gives time when T_i start execution.

Validation (T_i) :- It gives time when T_i finishes its read phase & starts its validation phase.

Finish (T_i) :- It gives time when T_i finished its execution or write phase.

If any transaction failed in validation test then its aborted & rollback.



NCURRENCY CONTROL TECHNIQUES : TIME STAMP-BASED PROTOCOL- ADVANTAGES & DISADVANTAGES

VANTAGES:

SERIALIZABILITY IS MAINTAINED

FREE FROM DEADLOCK

FREE FROM CASCADING ROLLBACKS

DISADVANTAGES :-

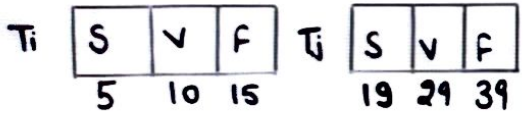
i) STARVATION IS POSSIBLE FOR LONG TRANSACTIONS WHICH WERE FORCEFULLY RESTARTED DUE TO CONFLICTING SHORT TRANSACTION.

ii) SCHEDULES ARE NOT RECOVERABLE SOMETIMES

Concurrency Control Techniques :- Validation Based Protocols :-

To Clear All The Validation Test By T_i, Then T_i Must Satisfy One Of The Following Condition :-

(i) Finish(T_i) < Start(T_j) :- It means T_i Is Older Transaction & It Get Finished Before T_j Starts. (No Overlap)



(ii) Finish(T_i) < Validate(T_j) :- This Ensures Actual Write By T_i & T_j Will Not Overlap

(iii) Validate(T_i) < Validate(T_j) It Ensure That T_i Has Completed Read Phase Before T_j Completes Read Phase.

Advantages :-

1) Maintains Seriazability.
2) Free From Cascade Rollback
3) Less Over Head Then Other Protocols.

Disadvantages

1) Starvation Of Long Transaction Transactions Due To Conflicting Transaction.

Validation Test :-

T_9	T_{10}
R(A)=10	
	R(B)
	B=B*5
R(B)=15	
	R(A)
	A=A+10
(Validate)	
Display(A+B)	
	(Validate)
	W(B)
	W(A)

Concurrency Control Techniques :- Multiversion Concurrency Control Protocols :-

This helps in stopping delay for Read operations.

It maintains the different versions of data items.

Each Write (operation W(A) create a new version of A

When any transaction wants to read A then appropriate version of A for R(A) is selected by control M/G.

Multiversion Time Stamp Ordering :-

- Each T_i is associated with unique time stamp. $TS(T_i)$

Each data item A_n (n is version of A) has 3 fields

i) Content :- Data value for that version.

ii) Write-Ts(A_n) : It is equal to timestamp of Trn. that has created A_n

(iii) Read-Ts(A_n) :- Time stamp of most recent transaction that successfully R(A_n)

Consider a transaction T_i & A_n be any data item with version n whose write timestamp is less than or equal to $TS(T_i)$.

i) Read(A_n) By T_i = Content of A_n are returned to T_i.

(ii) Write(A_n) By T_i = T_i roll backs if $Ts(T_i) <$ Read-Ts(A_n)

If $Ts(T_i)$ = Write-Ts(A_n) . Then new version is created with new content

---

$TS(T_1)=4$, $TS(T_2)=2$, $TS(T_3)=6$ $T_i=5$,

W(A) = 10

W(A) = 6

W(A) = 70

	C	W	R
A_1	6	2	R-TS
A_2	10	4	R-TS (5)
A_3	70	6	R-TS (6)

i) Read(A_2) By T_i

(ii) Write(A_3)

(iii) $Ts(T_i)$ = W-Ts(A_m) $A = 90$

A_4	90	T_i	R-TS (5)

Concurrency Control Techniques :- Multiversion Concurrency Control Protocols :-

Multiversion Two Phase Locking :-

It overcomes the disadvantages of multiversion timestamp ordering.

→ Single timestamp is given to every version of data items

→ Here timestamp counter (TS-counter) is used instead of logical counter & system clock

When ever transaction commits, TS-counter incremented by 1.

→ Updation are based upon rigorous two phase locking.

→ Read only transaction are based upon multiversion time-stamp ordering.

2.00 TRANSACTION & CONCURRENCY CONTROL :

SCHEDULE

- IT IS SERIES OF OPERATIONS FROM ONE TRANSACTION TO ANOTHER TRANSACTION.
- IT IS USED TO PRESERVE THE ORDER OF OPERATION IN EACH OF INDIVIDUAL TRANSACTION.

SCHEDULE ARE OF 3 TYPES :-

#1) SERIAL SCHEDULE #2) NON SERIAL SCHEDULE
#3) SERIALIZABLE SCHEDULE

#1) SERIAL SCHEDULE :-

- ONE TRANSACTION IS EXECUTED COMPLETELY BEFORE STARTING ANOTHER TRANSACTION.
- WHEN THE FIRST TRANSACTION COMPLETES ITS CYCLES, THEN THE NEXT TRANSACTION STARTS.

T_1	T_2
R(A)	
R(B)	
A = A+B	
W(A)	
	R(A)
	A = A - 500
	W(A)

SERIAL SCHEDULE

** FOR SET OF n TRANSACTION $n!$ DIFFERENT VALID SCHEDULE CAN BE POSSIBLE.

Transaction & Concurrency Control : Schedule

- It Is Series Of Operations From One Transaction To Another Transaction.
- It Is Used To Preserve The Order Of Operation In Each Of Individual Transaction.

Schedule Are Of 3 Types :-

#1) Serial Schedule #2) Non Serial Schedule
#3) Serializable Schedule

T_1	T_2
R(A)	
R(A)	
	R(A)
	R(B)
	A = A+B
A = A-B	W(A)
W(A)	

#2) Non-Serial Schedule - In This Schedule Instruction Of Transaction Execute Concurrently.

Interleaving Of Instructions Are Allowed Here

No. Of Non-Serial Schedule For 3 Tr^n.

T_1	T_2	T_3
4	2	3

Total No. Of Schedule $\frac{(2+3+4)!}{2!+3!+4!} = 1260$

Total Serial Schedule 3! = 6

Total No. Of Non-Serial Schedule
1260 - 6 = 1254

Transaction & Concurrency Control :

Schedule

- It Is Series Of Operations From One Transaction To Another Transaction.
- It Is Used To Preserve The Order Of Operation In Each Of Indivigual Transaction.

Schedule Are Of 3 Types :-

#1) Serial Schedule #2) Non Serial Schedule
#3) Serializable Schedule

#3) Serializable Schedule :- A Non-Serial Schedule Is Serializable If Its Result Is Equal To The Serial Schedule.

A= 500
B= 600
C= 700
D= 800

A= 500+600 = 1100
B= 600+700= 1300

T_2= A= 1100+700 = 1800
B= 1300+800= 2100

A= 500+600 = 1100
T_2 A= 1100+700=1800
B= 600+700=1300
1300+800=2100

T_1	T_2
R(A)	
R(B)	
A= A+B	
W(A)	
	R(A)
	A= A+C
	W(A)
R(C)	
B= B+C	
W(B)	
	R(B)
	B= B+D
	W(B)

Serializable Sch.

Transaction & Concurrency Control : Serializability & Testing of Serializability

Serializibility helps in identifying which non-serial schedule gives consistent result as given by serial schedule.

Test for Serializability of a Schedule

A precedence graph is used to test the serializability of a schedule.

G(V,E) V: set of vertices (all transactions)
E: set of edges $T_i \rightarrow T_j$

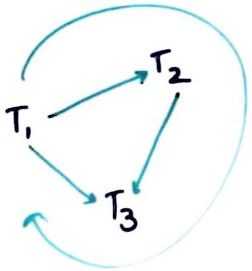
For which one of 3 conditions holds.

S	T_1	T_2
	W(Q)	R(Q)
	R(Q)	W(Q)
	W(Q)	W(Q)

If precedence graph contains no cyclic, means schedule S is serializable schedule

R(Q)	R(Q)	No edge

Precedence Graph



T_1	T_2	T_3
R(A)		
R(C)		
W(A)		
	R(B)	
W(C)		
	R(A)	
		R(C)
	W(B)	
		R(B)
		W(C)
	W(A)	
		W(B)

Transaction & Concurrency Control : Conflict Serializability

A Schedule Is Called Conflict Serializability If After Swapping Of Non Conflicting Operations, It Can Be Transform In To Serial Schedule (Conflict Equivalent To Serial Schedule)

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)

Conflicting Operations :-

T_1	T_2
W(A)	W(A)
W(A)	R(A)
R(A)	W(A)

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

Conflict Equivalent

Transaction & Concurrency Control :

- A Schedule Will Be View Serializable If It Is View Equivalent To A Serial Schedule.
- If A Schedule Is Conflict Serializable, Then It Will Be View Serializable.
- The View Serializable Which Doesnot Conflict Serializable Contains Blind Write

View Equivalent

Two Sched. S1 & S2 Are View Equivalent If They Satisfy The Following Condition.

#1) Initial Read :- An Initial Read Of Both Schedule Is Same

S1

T_1	T_2
R(A)	
	W(A)

S2

T_1	T_2
	W(A)
R(A)	

View Serializability

#2) Update Read :- In Schedule S_1, If T_i Is Reading A Which Is Updated By T_j Then In S_2 Also.

S1

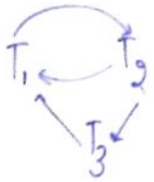
T_1	T_2	T_3
W(A)		
	W(A)	
		R(A)

S2 ×

T_1	T_2	T_3
	W(A)	
W(A)		
		R(A)

#3) Final Write :- Final Write Must Be Same B/w Both The Schedule.

Loop NCS



T_1	T_2	T_3
W(A)		
	R(A)	
		W(A)

T_1	T_2	T_3
	R(A)	
W(A)		
		W(A)

Transaction & Concurrency Control :

With 3 transactions, the total no. of possible serial schedule = 3! = 6

$S_1 = \langle T_1, T_2, T_3 \rangle$
$S_2 = \langle T_1, T_3, T_2 \rangle$
$S_3 = \langle T_2, T_1, T_3 \rangle$
$S_4 = \langle T_2, T_3, T_1 \rangle$
$S_5 = \langle T_3, T_1, T_2 \rangle$
$S_6 = \langle T_3, T_2, T_1 \rangle$

S_1

T_1	T_2	T_3
R(A)		
W(A)		
	W(A)	
		W(A)

Example of View Serializability

T_1	T_2	T_3
R(A)		
	W(A)	
W(A)		
		W(A)

1) Initial Read ✓
2) Update Read ✓
3) Final Write ✓

NSACTION & CONCURRENCY CONTROL :

RECOVERABILITY OF SCHEDULE

RE IS NO OF TRANSACTIONS IN A SCHEDULE

TRANSACTION IS READING SOME OTHERS

INSTRUCTION & AFTER THAT IF THERE IS

OR ERROR THEN, IF WE ARE ABLE

N IN CONSISTENT STATE THEN IT IS

RABLE SCHEDULE OTHERWISE IRRECOVERABLE

E

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
(F) COMMIT	
	COMMIT
FAILURE	

Recoverable (S)

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
FAILURE	
	COMMIT

IRRECOVERABLE (S)

Transaction & Concurrency Control : Cascading Rollback & Cascade Less Schedule :-

Cascading Rollback:- Due To Failure One Transaction, Some Times We Have To Rollback Many Transactions Which Is Called Cascading Rollback.

Cascadeless Schedule: This Schedule Avoid The Cascade Rollback. We Have To Commit That Transaction Which Updates Data Item Before It Is Read By Other Transaction.