

Normalization

it is a technique to remove or reduce the redundancy from a table.

if there is row-level duplicacy, we can assign a primary key to remove dupicants.

we will set a column as primary key and it will take care of the duplicate values.

Redundancy from a tab

Primary Key (Unique + Not Null)

SID	Sname	Age
1	RAM	20
2	Vishnu	25
1	RAM	20

Row level

in column level redundancy some columns can have the same entries.

there three type of problem can occur :

1. Insertion anomaly - problem for new entries
2. deletion anomaly - useful data may be lost
3. updation anomaly - many duplicate entires will be updated.

Normalization

In relation Anomaly
Deletion Anomaly
Update Anomaly

Note: Not Null

Student ID	Sname	Cid	Cname	FID	Fname	Salary
1	RAM	C ₁	DBMS	F ₁	John	30000
2	Ravi	C ₂	JAVA	F ₂	Bob	40000
3	Nitin	C ₁	DBMS	F ₁	John	30000
4	Ankit	C ₁	DBMS	F ₁	John	30000
5	:	:	:	:	:	

Normalization: decomposes the tables till the redundancy is removed.

1st Normal Form:

table should not contain any multivalued attribute.

like in below eg. 1 has course c/c++ that should not be the case.

Lec-21: First Normal form in DBMS in HINDI | 1st Normal form क्या होती है

EF Code

First Normal Form

→ Table Should not Contain Any multivalued Attribute.

Not in 1st NF

Rollno	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	Onkar	C/DBMS

To convert a table to 1st normal form we have 3 solutions:

1. we can take the multiple values as separate entries like one entry as (1 sai c) and another as (1 sai c++) and we can make the primary key composite of (rollno and course).

Table Should not Contain
multivalued Attribute.

Not. in 1 st NF	
Name	Course
Sai	C/C++
Harsh	Java
Onkar	C/DBMS

Primary Key.

Rollno	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Onkar	C
3	Onkar	DBMS

Rollno Course

2. we can take the maximum number of values an attribute is having and make that number of separate columns.

1st Normal Form

Table Should not Contain
any multivalued Attribute.

Not. in 1 st NF	
Rollno	Name
1	Sai
2	Harsh
3	Onkar

Primary Key: Rollno

Rollno	Name	Course1	Course2
1	Sai	C	C++
2	Harsh	Java	Null
3	Onkar	C	DBMS

Course

Null Null
Filling null to so many values
cannot be a good representation

3. The table can be divided into two different tables and the primary key of the base table will be the foreign key of the referencing table.

Any multivalued functional dependency foreign key

Rollno	Name
1	Sai
2	Harsh
3	Aman

Rollno	Course
1	C
1	C++
2	Java
3	C
3	DBMS

Rollno	Course
1	S
2	H
3	O

Closure Method

use to find the all candidate keys.

if the closure of an element can determine all the elements of table it is a candidate key.

Lec-22: Finding Closure of Functional dependency in DBMS | Easiest & Simplest way

Rollno A → B B → C C → D D → A
Closure Method

$R(A B C D E)$
 $FD = \{ A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A \}$

→ Candidate key

$R(A B C D)$

$CK = \{ A \}$ $FD = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D \}$
 $A^+ = BCDA$

$R(A B C D)$
 $FD = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D \}$

$R(ABCD)$
 FD
 $CK = \{A\}$ $\left\{ A \rightarrow B, B \rightarrow C, C \rightarrow D \right\}$
 $A^+ = BCDA$ $(AB)^+ = ABCD$
 $B^+ = BCD$
 $C^+ = CD$
 $D^+ = D$
 $AB = CKX$
 SufKey

◀ ▶ ⏪ ⏹ 5:55 / 17:46 • Example 2 >

$R(ABCD)$
 FD
 $CK = \{A\}$ $\left\{ A \rightarrow B, B \rightarrow C, C \rightarrow D \right\}$
 $P_{\text{prime}} = A$ $A^+ = BCDA$ $(AB)^+ = ABCD$
 $N_{\text{non prime}} = \{B, C, D\}$ $B^+ = BCD$
 $C^+ = CD$
 $D^+ = D$
 $AB = CKX$
 SufKey

here AB combine is not a candidate key although it determines all the elements of the relation, it is combinly determining. so, it's not a candidate key it's a super key. a candidate key is a minimalist key.

prime attribute: it is the attribute from relation which is used in making the candidate key.

$R(ABCD)$

$$FD = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A \}$$

$$A^+ = ABCD$$

$$B^+ = BCDA$$

$$C^+ = CDAB$$

$$D^+ = DABC$$

Prime Attribute

$$= \{A, B, C, D\}$$

Non Prime Attribute

$$= \{\emptyset\}$$

Functional dependencies:

they are of two types:

1. trivial FD: Valid in all cases.

if $x \rightarrow y$ (x determines y) then y is a subset of x .

and if we take the intersection of LHS n RHS will never be NULL. ($x \cap y \neq \text{NULL}$)

2. non-trivial FD: if $x \rightarrow y$ then y is not a subset of x

and their intersection is always null.

y is dependent and x determinant.

Non-trivial FD:

$$\text{id} \rightarrow \text{Event}$$

$$\text{id} \rightarrow \text{Location}$$

$$\text{id} \rightarrow \text{Sid} \rightarrow \text{Sname}$$

$$\text{id} \rightarrow \text{Sid} \rightarrow \text{Semester}$$

$$\text{id} \rightarrow \text{Sid} \rightarrow \text{PhoneNo}$$

$$x \cap y = \emptyset$$

Valid: Y is Determined by X.

Sid → Sname	
1	Ranjit
2	Yash

Sid → Sname	
1	Ranjit
1	Ranjit

Invalid:

Sid → Sname	
1	Ranjit
1	Yash

Properties of FD:

Reflexivity: if y is subset of X then $X \rightarrow Y$

Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$

Transitive: if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Union: if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Decomposition: if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Predistributive: if $X \rightarrow Y$ and $WY \rightarrow Z$ then $WX \rightarrow Z$

Composition: if $X \rightarrow Y$ and $Z \rightarrow W$ then $XZ \rightarrow YW$

whenever y has any confusion it will go to determinant. that whether it is valid or not.

2nd Normal Form:

table or relation must be in 1st Normal form

All the non-prime attributes should be fully functional dependent on

Candidate Key.

Customer	
Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

Customer ID is Prime Attribute

2nd NF

Customer		
Customer ID	Store ID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Candidate key: Customer ID, Store ID

Prime Attribute: Customer ID, Store ID

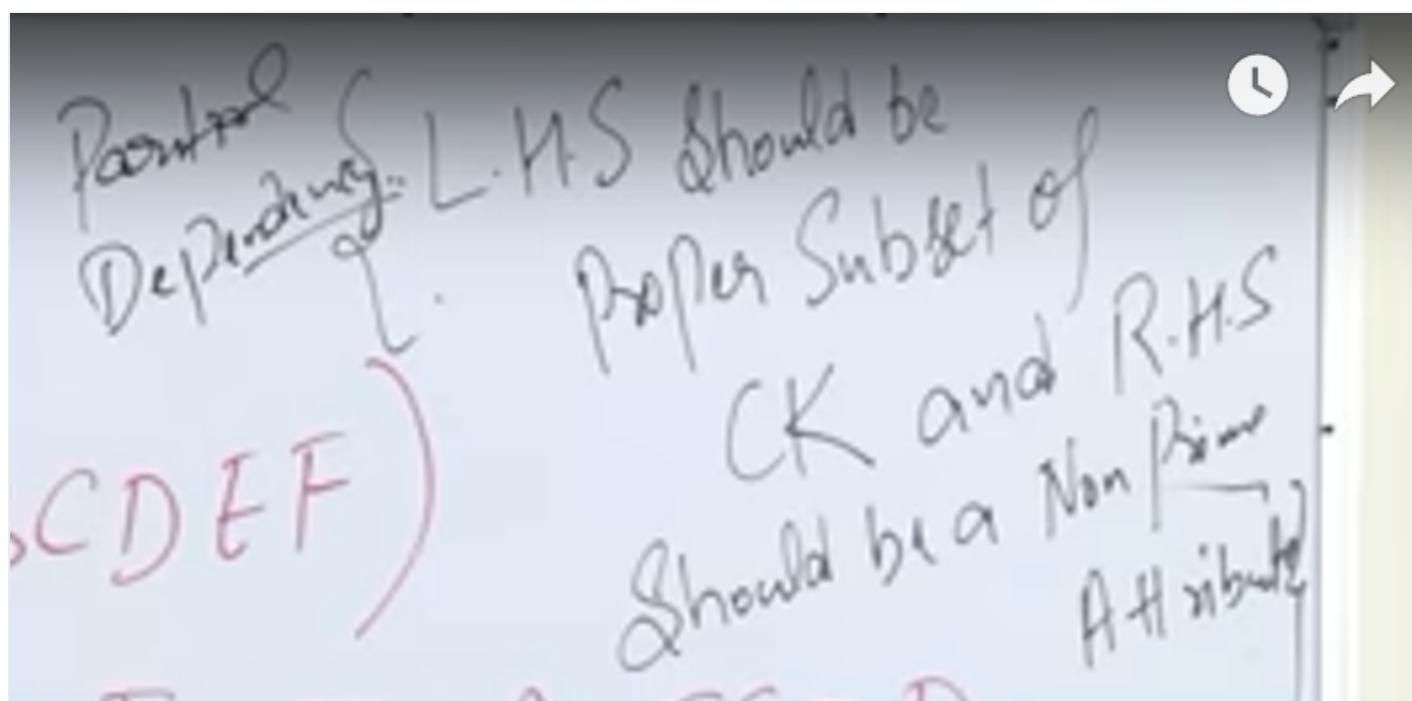
Non-prime: Location

There Should be No Partial Dependency in
the relation.

if any non-prime attribute is determined by the part of the candidate key (for eg AB and $B \rightarrow C$ and C is non-prime, so C is determined by B, this is called partial dependency which should not be present).

if this condition appears:

LHS is a proper subset of CK and RHS is a non prime attribute.



then the table is not in the 2nd normal form.

3rd Normal Form

a non prime attribute cannot be determined by a non prime attribute.

each FD
= LHS must be a CK or SK (OR)
RHS is a prime attribute

3rd NF Third Normal Form

- table or relation must be in Second Normal form
- and
- there should be no transitive dc in table.

$CK = \{ \text{Rollno} \}$
 $FD: \text{Rollno} \rightarrow \text{State}$
 $PK = \{ \text{Rollno} \}$
 $NPAs = \{ \text{State, City} \}$
 $\text{Rollno} \rightarrow \text{State and State} \rightarrow \text{City}$

Rollno	State	City
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

$R(ABCD)$
 $FD: AB \rightarrow CD, D \rightarrow A$
 $CK: AB^+ = ABCD$
 $CK: \{ AB \}$
 $CK: \{ AB \}$
 $DG^+ = DBAC$
 $PA = \{ A, B, D \}$
 $NPA = \{ C \}$

$FD: AB \rightarrow C, C \rightarrow B$

$CK: AB$

$PA: A$

$NPA:$

condition for a table to be in 3rd NF:

LHS of all FD

is CK or SK OR

RHS is Prime Attribute

7

Boyce codd normal form (BCNF)

a special case of 3rd normal form.

BCNF always have lossless decomposition.

Condition: In LHS of all function dependency, candidate key should be present.

⇒ Table should be in 3rd Normal form

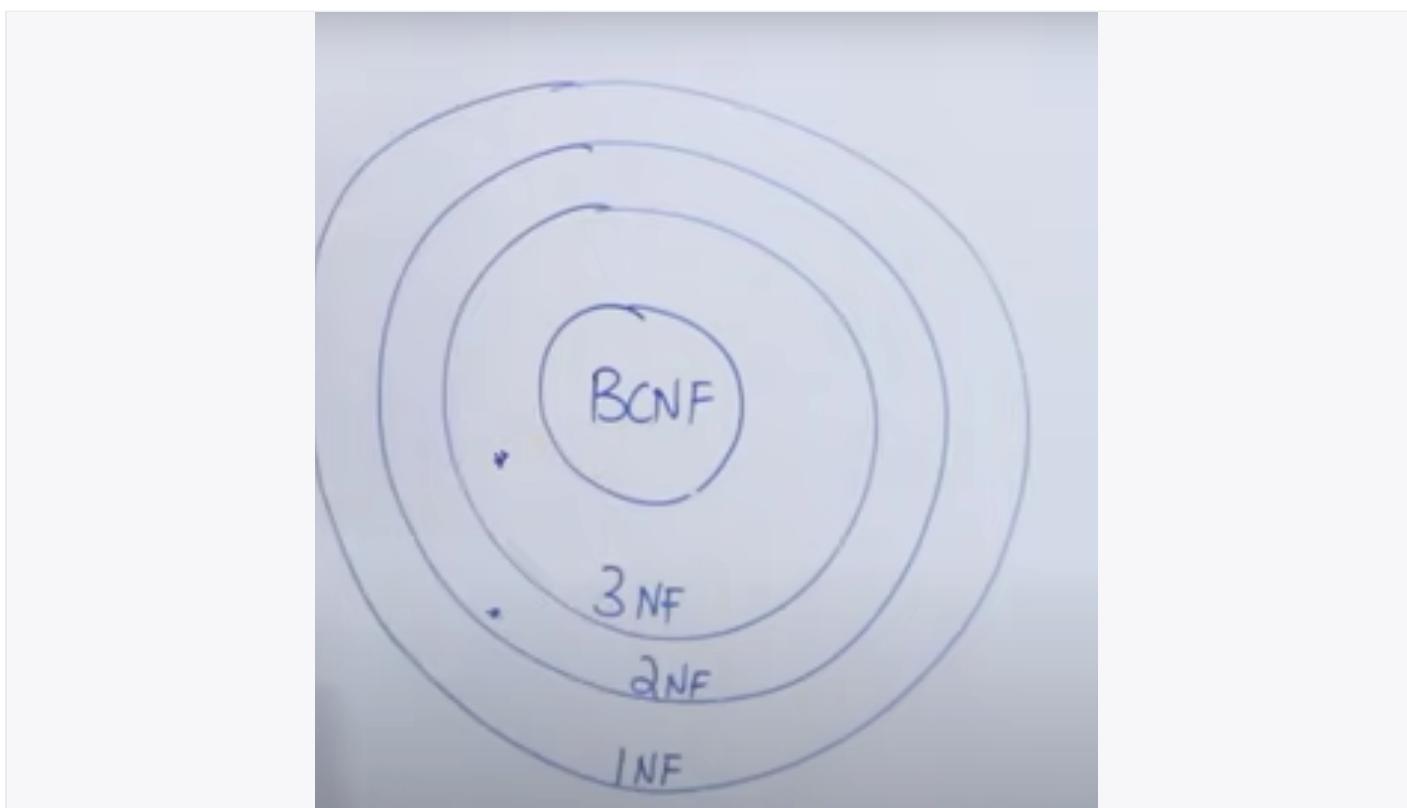
3NF

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

CK: { Rollno, Voterid }

FD:

- ✓ Rollno → name ^{1st FD}
- Valid Rollno → Voterid
- Valid Voterid → age
- Valid Voterid → Rollno



Lossy vs lossless Decomposition

find the value of C if the value of A=1

Select R₂.C from R₂ Natural Join R₁
Where R₁.A = '1'

R

A	B	C
1	2	1
2	2	2
3	3	?

R₁(AB)

R₂(BC)

R

A	B
1	2
2	2
3	3

Natural Join = Cross Product +

R ₁	B	R ₂	C
1	2	2	1
1	2	2	2
1	2	3	2
2	2	2	1
2	2	2	2
2	2	3	2
3	3	2	1
3	3	2	2
3	3	3	2

◀ ▶ ⏪ ⏹ 11:08 / 22:16

CC 🔍

as in original table value of c at a=1 was just 1 but after merging the table the values become 2 hence this is a flaw and make the data base inconsistent

? Natural Join =

R

A	B	C
1	2	1
2	2	2
3	3	2

R₁(AB)

R₂(BC)

two values of c
at a = 1

R'

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

R

because of inconsistency this decomposition is called lossy-decomposition if it was not there then it would be lossless decomp.

here the entries of A=1 becomes 2 after the decomposition as this is inconsistency in data base hence this is Lossy

1

2

2

2

2

2

2

2

2

2

2

2

2

! Natural Join = (R1 x R2)

$\rightarrow R_1(AB)$

$\rightarrow R_2(BC)$

R'

as the remaining three were also present in the previous tabel

A	B	C
1	2	1
1	2	1

R_1

A	B
1	2
1	2

when we decompose a table into other tables we put a attribute common in both tables, we should follow criteria otherwise inconsistency may occur:

→ the common attribute between the split tables should be candidate key or super key of either R_1 or R_2 or both.

→ Common Attribute should be CK or SK of either R_1 or R_2 or Both.

Splitting Rule

1	2	3
1	2	3

then we will get a lossless relation.

condition for lossless relation :-

1. $R_1 \cup R_2 = R$ (R is the original relation).

2. when we take the intersection of the split tables it should not be null

→ $R_1 \cap R_2 \neq \text{NULL}$

3. the common attribute should be C.Key/S.Key of R_1 or C.Key/S.Key of R_2 or both

Summary of all normal form:

1 st Normal form	2 nd Normal form	3 rd Normal form	BCNF	4 th Normal form	5 th Normal form
<ul style="list-style-type: none"> * No Multivalued attribute Only Single valued <p>(AB) \rightarrow C</p>	<ul style="list-style-type: none"> * In 1st NF + No Partial Dependency * Only Full Dependency 	<ul style="list-style-type: none"> * In 2nd NF + No Transitive Dependency * No Non-Prime should determine non-prime <p>X \rightarrow Y \rightarrow Z</p>	<ul style="list-style-type: none"> * In 3rd NF + L.H.S must be CK or SK 	<ul style="list-style-type: none"> * In BCNF + No Multivalued Dependency X $\rightarrow\!\!\!\rightarrow$ Y 	<ul style="list-style-type: none"> * In 4th NF + Lossless Decomposition

Practise Question-

Lec-29: Practice Question on Normalization | Database Management System

R(ABCDEF), Check the Highest Normal form?

FD: { AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A }

Step 1: Find all CKs in Relation

CK: { AB, FB, EB, CB }

Step 2: Write all Prime Attributes.

{ A, B, C, E, F }

Step 3: Write all Non-Prime Atts.

{ D }

$A^+ = A$
 $B^+ = B$

$AB^+ = ABCDEF$
 $FB^+ = FBACDE$
 $EB^+ = EBFACD$
 $CB^+ = CBDEFA$

Exit full screen (f)

Cover and equivalence of function dependency

Equivalence of Functional Dependency

$$X = \{A \rightarrow B, B \rightarrow C\} \quad | \quad Y = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$$

we have to see if X covers Y $\underline{X \geq Y}$
 if Y covers X $\underline{X \leq Y}$
 $\boxed{X \equiv Y}$

If these both conditions satisfies then X is equivalent to Y

Introduction >

if the given check does Y cover X , then take the RHS that is X and now take the LHS that is its functional dependencies and check it.

Now we will take the element from Y and take closure of it from X .

and then look for the all dependencies are covered in closure from Y taken from X .

Equivalence of Functional Dependency

$$X = \{A \rightarrow B, B \rightarrow C\} \quad | \quad Y = \{\underline{A \rightarrow B}, \underline{B \rightarrow C}, \underline{A \rightarrow C}\}$$

if X covers Y $\underline{X \geq Y} \checkmark$

$$\begin{array}{l} A^+ = ABC \\ B^+ = BC \end{array}$$

hence all the dependencies taken from X are covered from Y , so X covers Y

Equivalence of Functional Dependency

$$X = \{A \rightarrow B, B \rightarrow C\} \quad | \quad Y = \{A \rightarrow B, \underline{B \rightarrow C}, A \rightarrow C\}$$

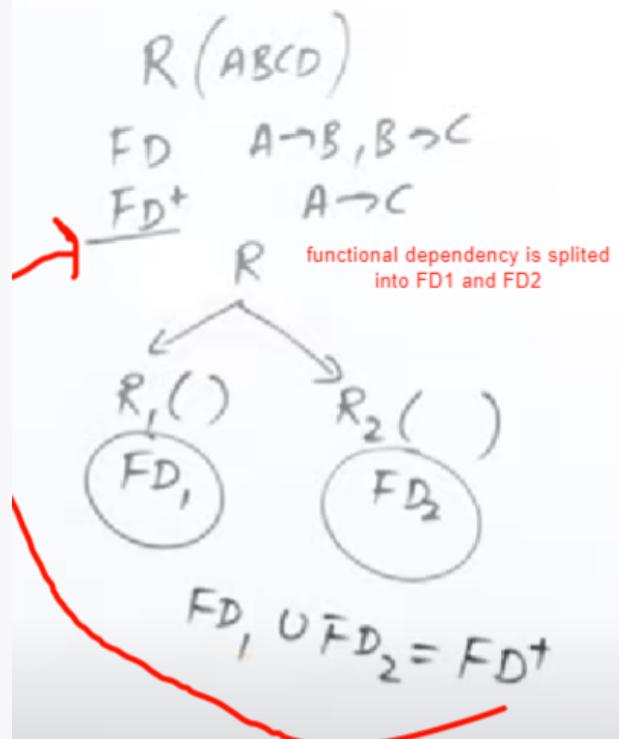
Y covers X. $X \geq Y$ $Y \geq X$

here, Y is also covering X

$A^+ = ABC$ $B^+ = BC$ $X \equiv Y$ ✓

So they both are equivalent

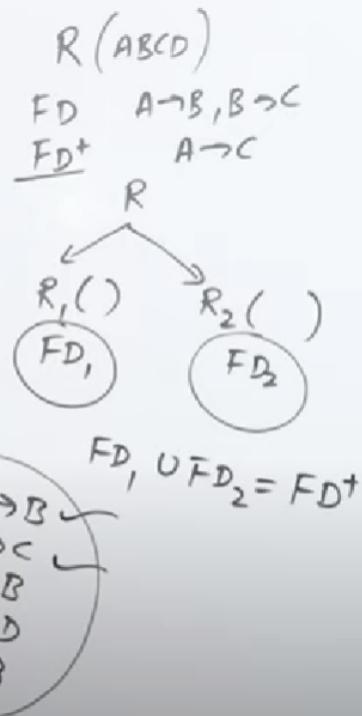
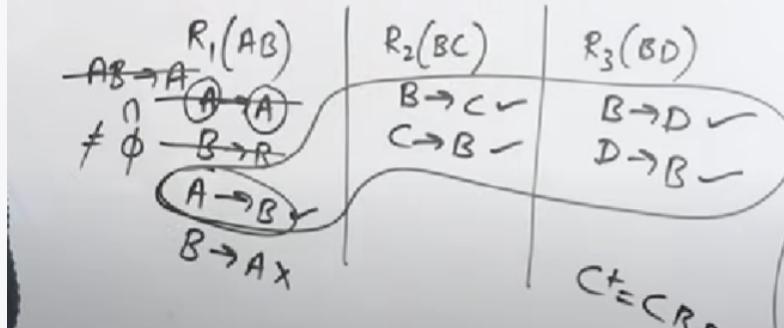
Dependency Preserving Decomposition



when we decompose the table to normalize it we also split the functional dependencies and if we take the union of all the functional dependencies then it should be equal to the original functional dependency.

Dependency Preserving Decomposition

Let $R(ABCD)$ with Functional Dependencies
 $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B\}$
 $B^+ = BCD$ $C^+ = CBD$
 R is decomposed into $R_1(AB), R_2(BC), R_3(BD)$



Transaction management :

Lec-69: Introduction to Transaction Concurrency in HINDI | Database Management System

Transaction

Transaction: It is a set of operations used to perform a logical unit of work.

→ A transaction generally represent change in database.

Work - Withdraw money.

→ Transaction generally represent Change in database.

Work - Withdraw money.

Read, Write
Commit
Database
Access.

Change

Server
(Database)



Transfer.

$$R(A) - 1000$$

$$A = 1000$$

$$B = 2000$$

$$A = A - 500$$

$$W(A) - 500$$

$$R(B) - 2000$$

$$B = B + 500$$

$$W(B) - 2000$$

$$\text{Commit}$$

ACID Properties



Atomicity Consistency Isolation Durability

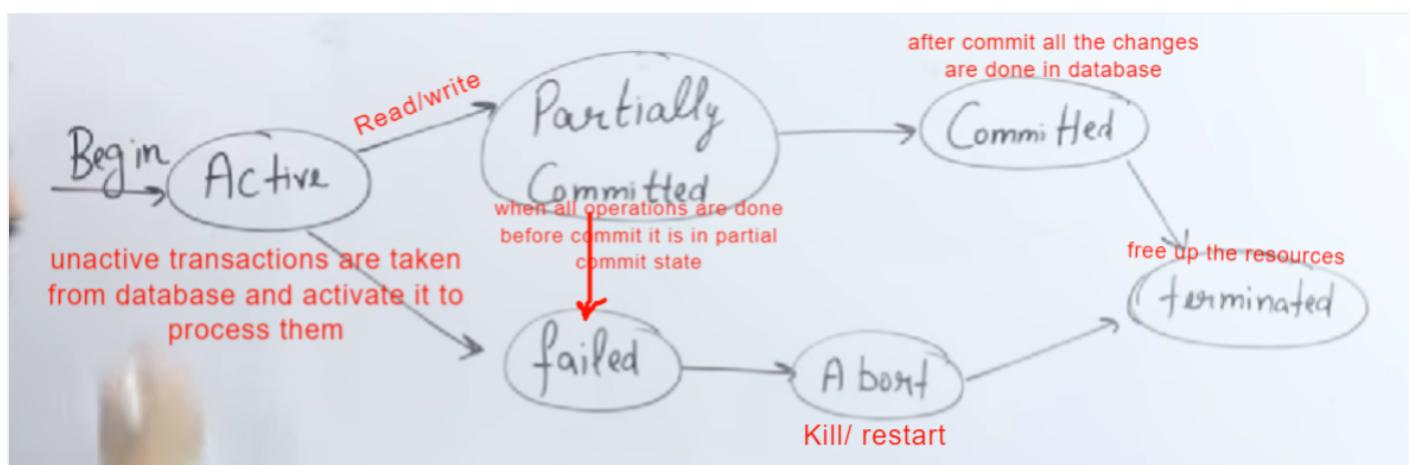
→ atomicity (either all or none): if the transaction fails before the commit then we will roll back the transaction. A failed transaction can't be resumed it can only restart.

→ consistency: before the transaction start and after the transaction completes, sum of the money or work should be same.

→ isolation: to make a parallel transaction to a serial transaction, to make it consistent.

→ durability: all the changes in the database should be permanent.

Transaction states:



Schedule

Schedule - it is chronological execution sequence of multiple transactions.

schedule is, multiple transaction kis sequence mai execute ho rhi hai.

the schedule is of two types ⇒ 1. serial 2. parallel

we categorize schedules into two things ⇒ serializability and recoverability.

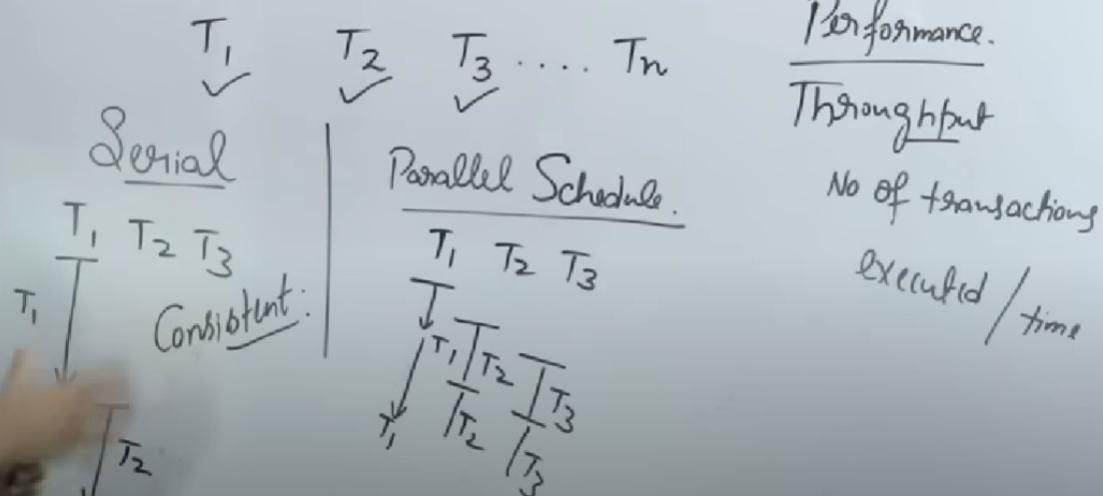
⇒ serial: the execution of a transaction will be started only after the completion of previous transaction.

its advantage is consistency. the result is always consistent. and disadvantage is waiting time of transactions and performance is degraded.

⇒ parallel: in this type of scheduling multiple transactions can come and start.

multiple transactions are executed at a single time. advantage : performance and throughput increases. disadvantage: inconsistency.

Schedule - it is chronological execution sequence of multiple transactions.



16 • Advantage of Parallel Schedule >

Concurrency : multiple transactions in a single time (Parallel scheduling).

All Concurrency Problems | Dirty Read | Incorrect Summary | Lost Update | Phantom Read

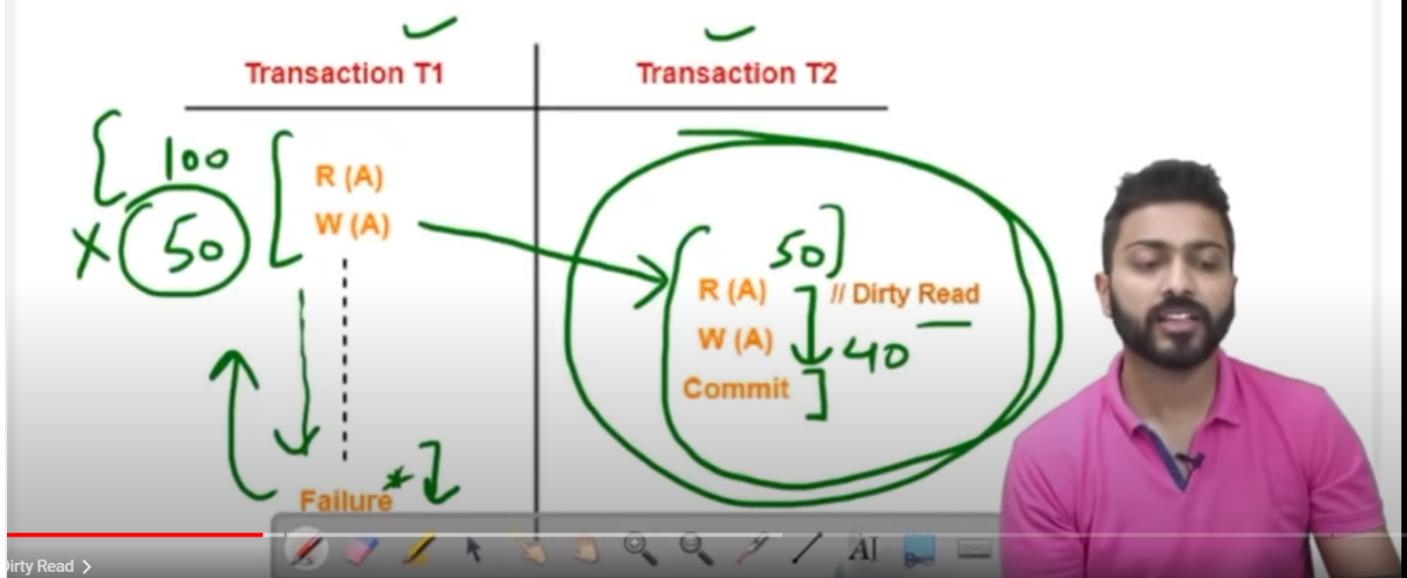
Types of problems in concurrency

- Dirty read
- Incorrect summary
- Lost update
- Unrepeatable read
- Phantom read

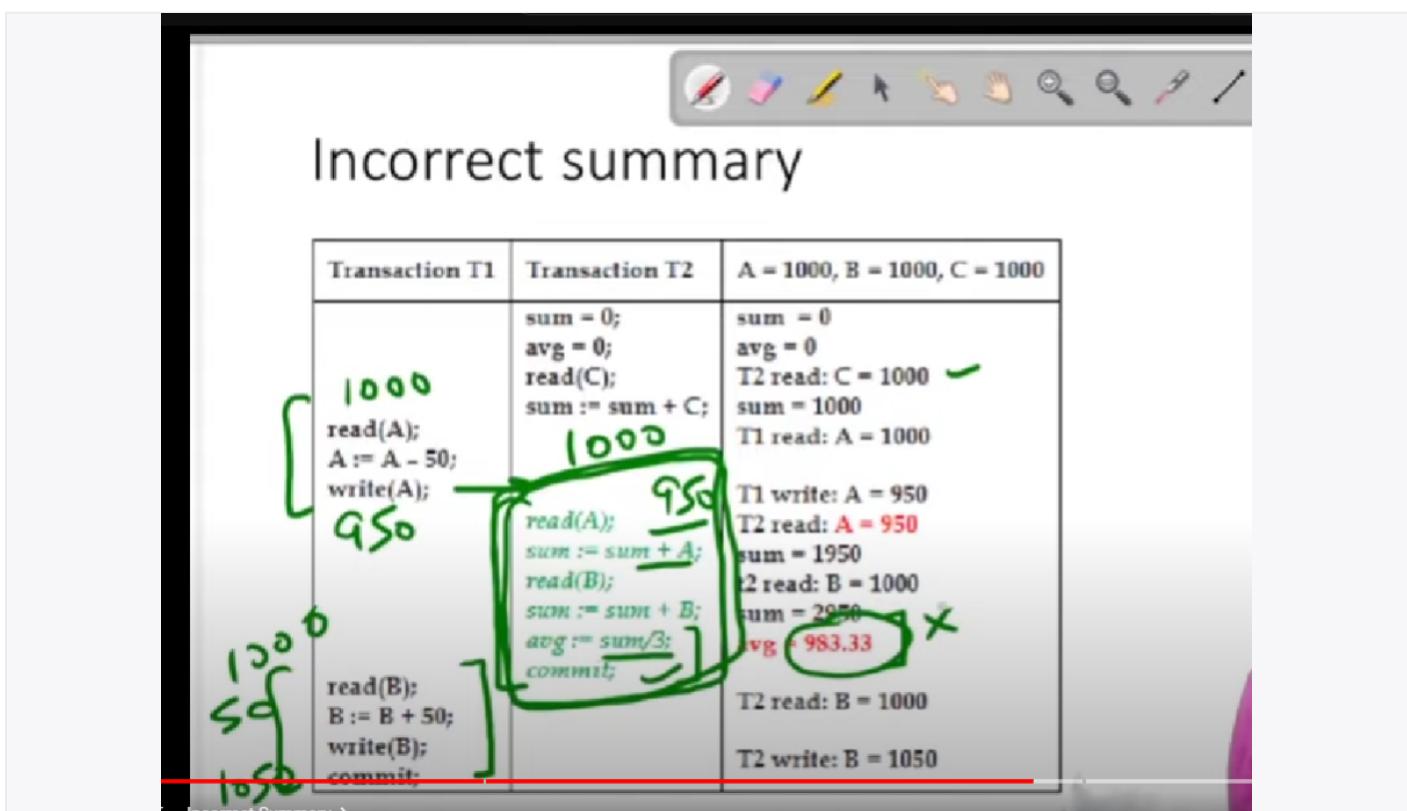
Multiple



Dirty Read or Uncommitted Read or RAW



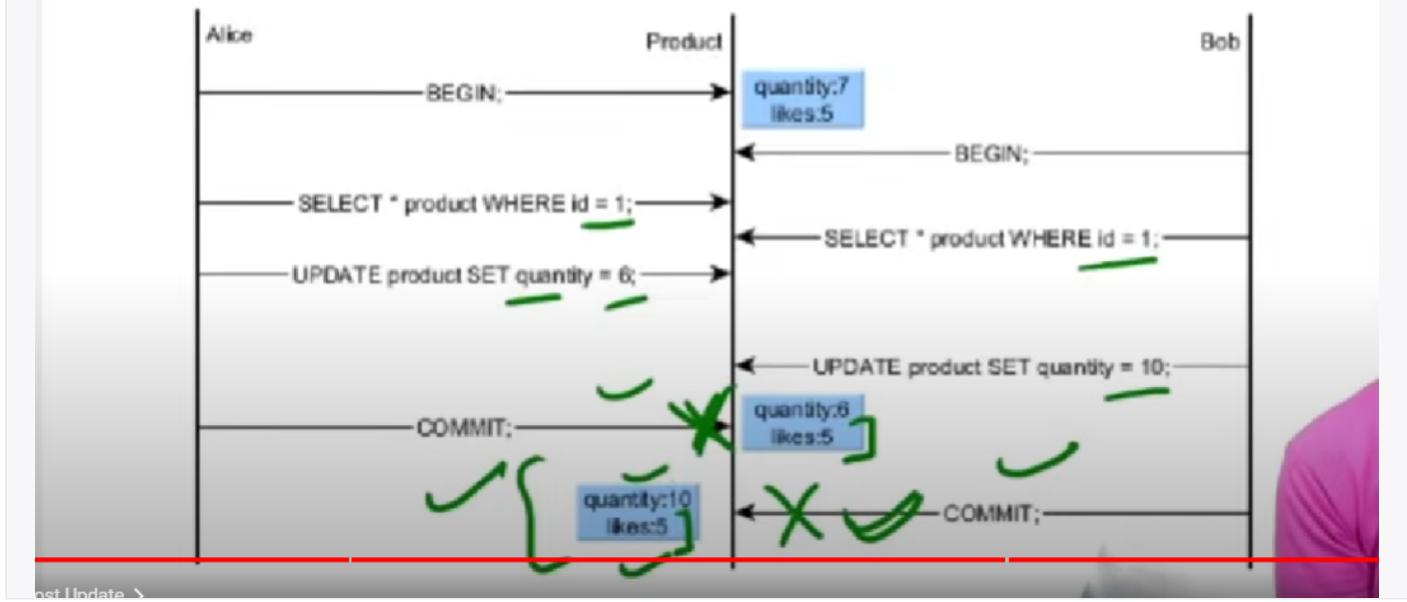
dirty read: a transaction t1 starts and did some changes and then t2 get it and did some changes and afterwards if t1 fails then it is roll back and t2 comes with dirty read as it make changes to a wrong value or variable.



incorrect summary: this problem generally appears when a transaction was executing and then another transaction comes and calculated something in between which was waste or wrong.

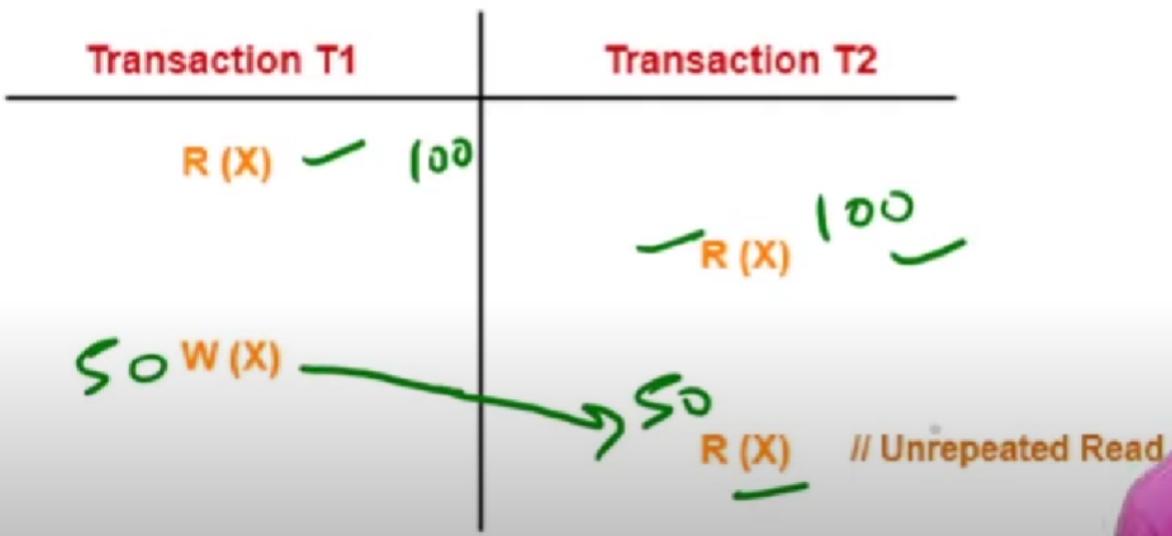
Lost Update

$id=1$ 10 1c

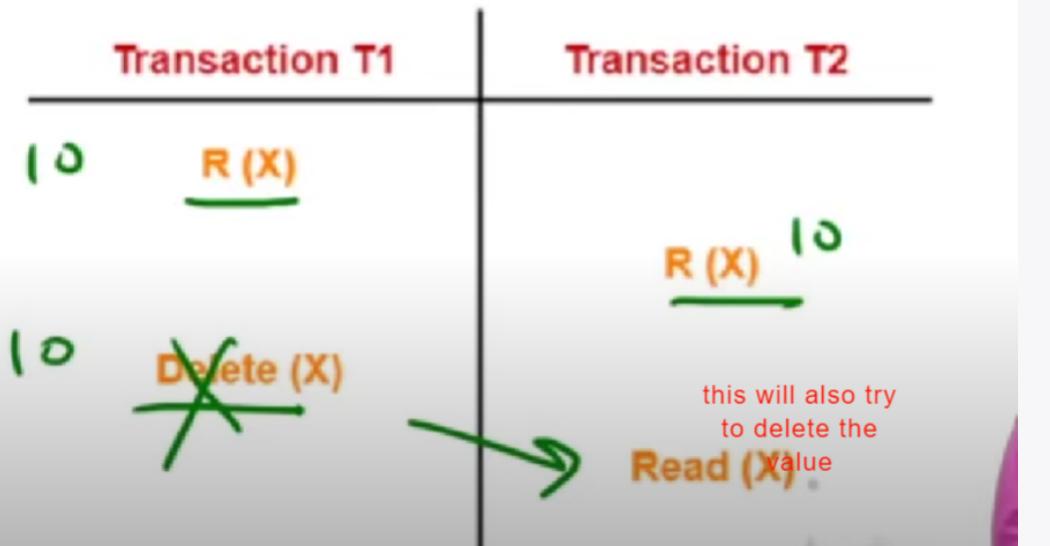


lost update: generally arises when someone updates the already committed transaction.

Unrepeatable read

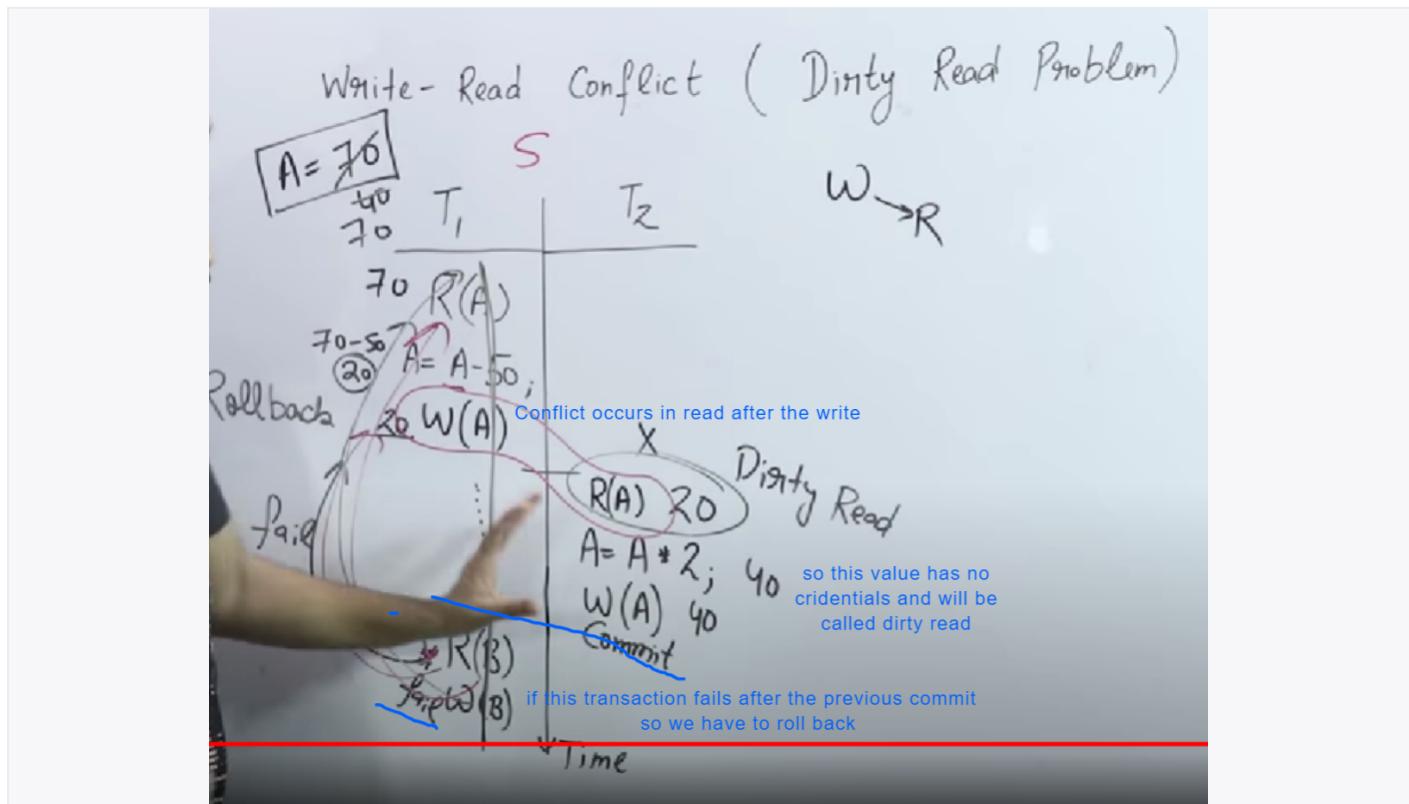


Phantom read



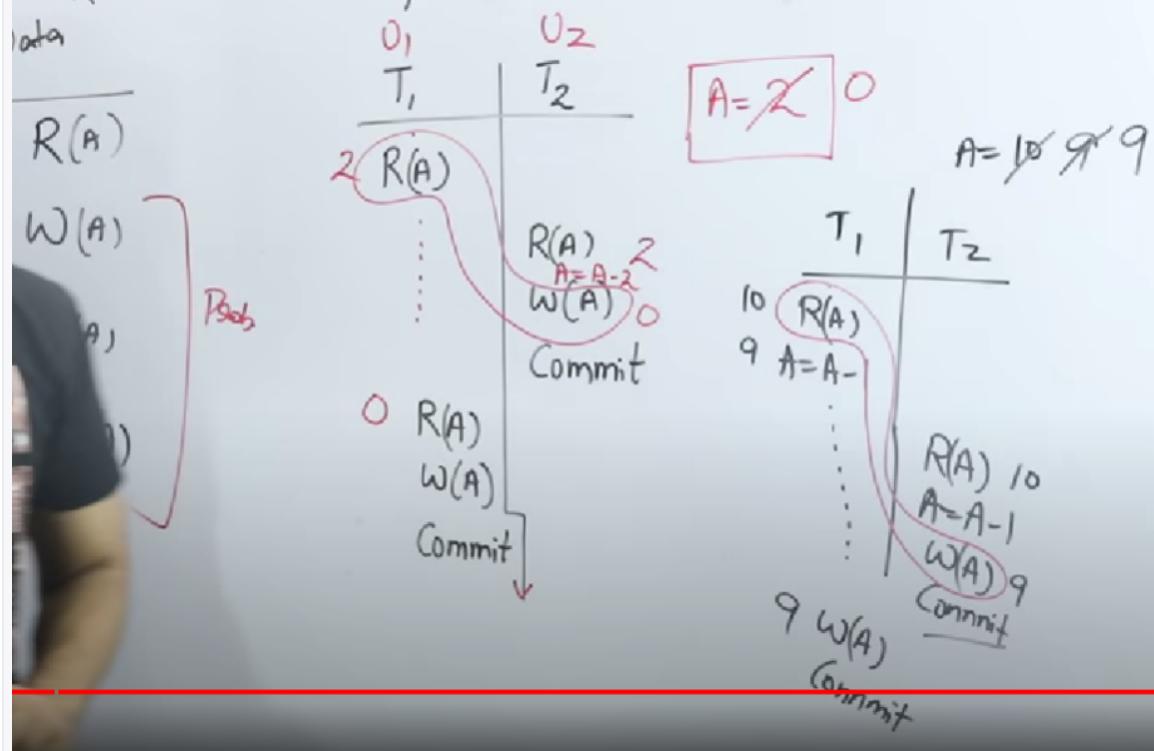
Write - read conflict

it occurs in parallel transactions.



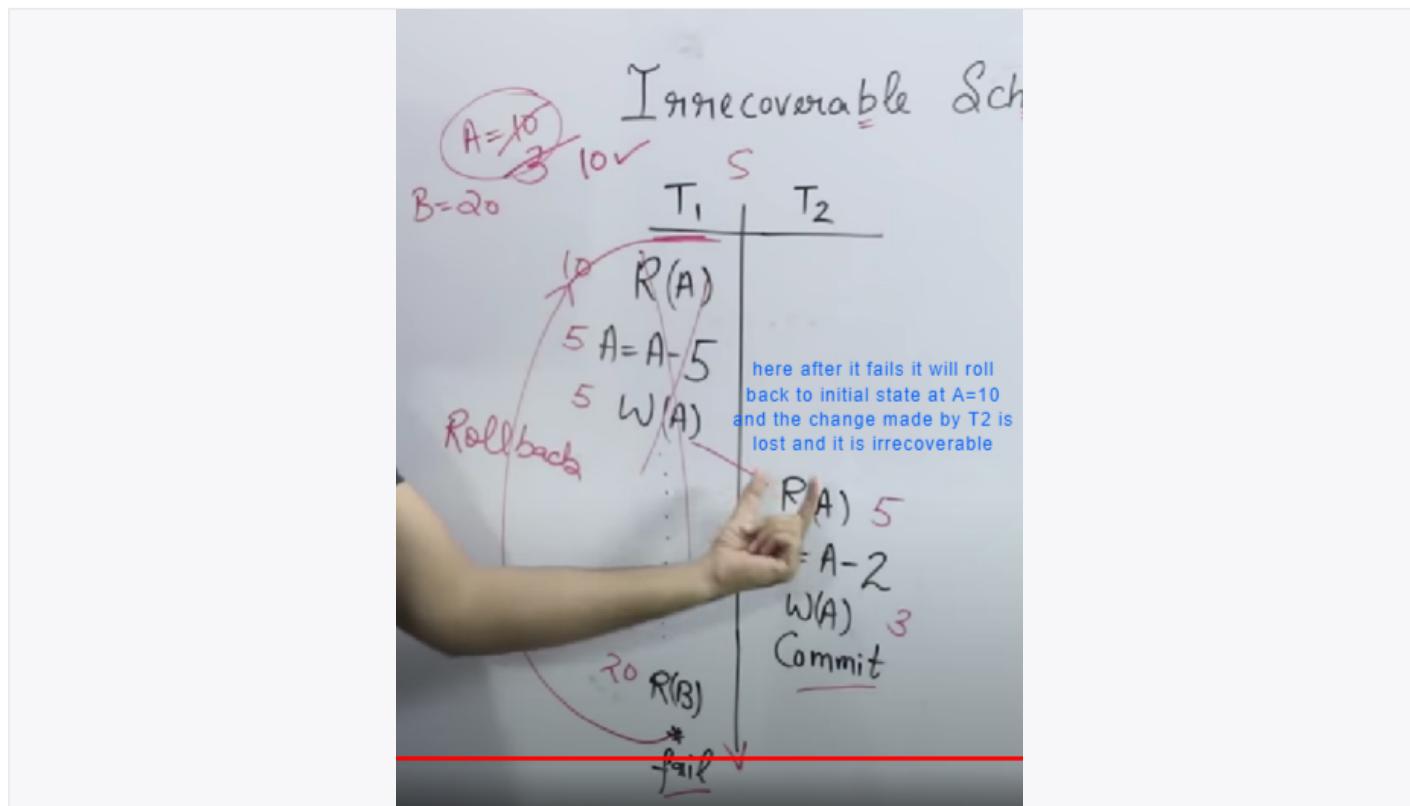
read-write conflict :

Read - Write Conflict or Unrepeadable Item



Recoverability

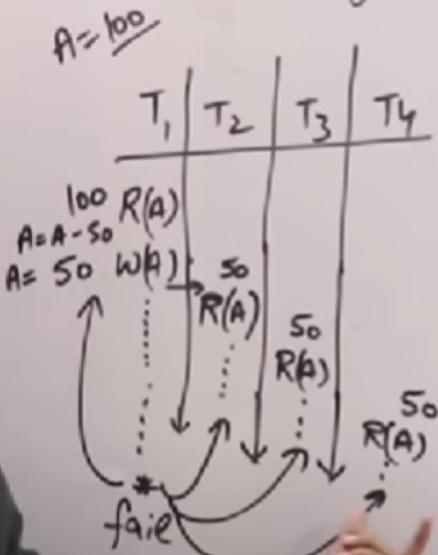
irrecoverable schedule: a schedule in which a value can not be recovered.



Cascading vs Cascadeless schedule

cascade: due to occurrence of one event multiple events are automatically occurring.

Cascading Schedule Vs Cascadeless Schedule

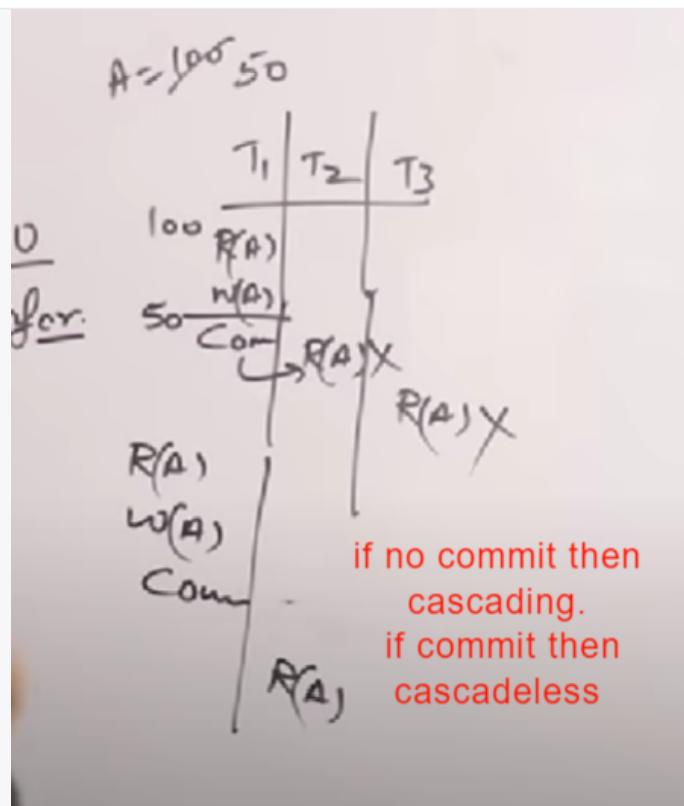


when transaction t1 fails it will be roll back to $A = 100$ and t2 t3 and t4 will be dirty read and have wrong value so as the t1 fails t2 t3 t4 will be automatically forced to stop and roll back

Disadvantage:
the CPU utilization is wasted and performance is degraded.

this is called cascading schedule

for cascadeless schedule: we will not allow the transactions to read from other transactions until it is not committed.

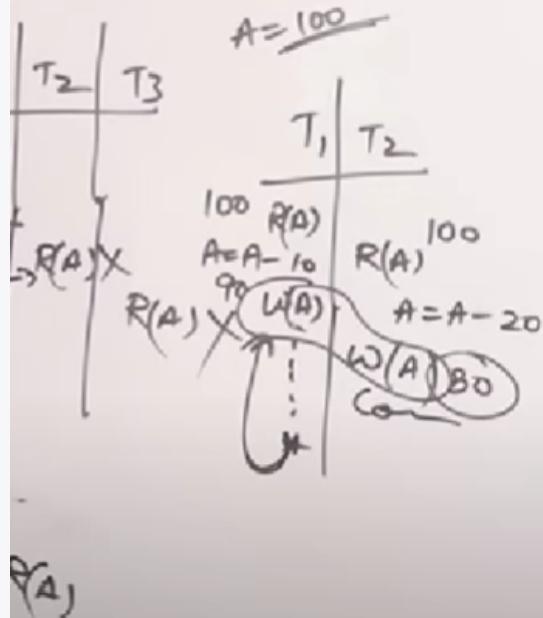


A transaction can not **read** before commit but can **write** before commit.

there cannot be read after write on the same data until it is not committed.

but write after write can also occur problem as if t1 gets fail and it will roll back and the write done by t2 will be lost. it will be cascadeless but its can be lost update if t1 fails

Cascadeless Schedule



so these problems are solved by strict cascading schedule.

Serializability

interleaving of process is not done.

serial schedule: in this scheduling, the next process will start only after the completion of the first.

the serial transaction cannot be serialized only parallel transactions cannot be serialized.

we will be given a parallel schedule and we have to check the serializability that there exists a clone of that schedule that is in serial form.

13

3 transactions
so 6 possibilities
as $3! = 6$

S	T_1	T_2	T_3
	$R(A)$		
		$R(A)$ $W(A)$	
			$W(A)$
	$R(B)$ $W(B)$		

$T_1 \rightarrow T_2 \rightarrow T_3$
 $T_1 \rightarrow T_3 \rightarrow T_2$
 $T_2 \rightarrow T_3 \rightarrow T_1$
 $T_2 \rightarrow T_1 \rightarrow T_3$
 $T_3 \rightarrow T_1 \rightarrow T_2$
 $T_3 \rightarrow T_2 \rightarrow T_1$

to convert this schedule to serial we have 6 possibilities

Serialization can be done in 2 ways: conflict equivalent and view equivalent .

to check whether s and s' are conflict equivalent or not, we will check the pairs in below example:

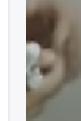
"Conflict Equivalent"

$R(A)$	$R(A)$	$\{$	Non Conflict Pairs
$R(A)$	$W(A)$	$\}$	
$W(A)$	$R(A)$	$\{$	Conflict Pairs
$W(A)$	$W(A)$	$\}$	
$R(B)$	$R(A)$	$\{$	
$W(B)$	$R(A)$	$\}$	
$R(B)$	$R(A)$	$\{$	Non Conflict
$W(B)$	$W(A)$	$\}$	
$R(B)$	$W(A)$	$\}$	
$W(B)$	$W(A)$	$\}$	

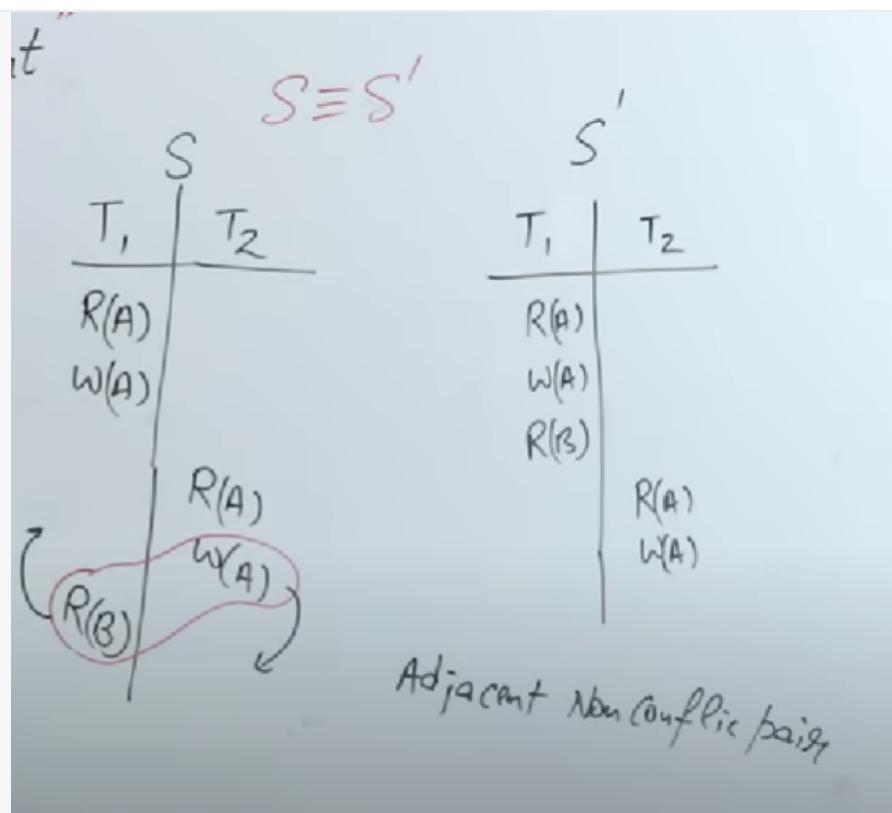
$s \equiv s'$

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

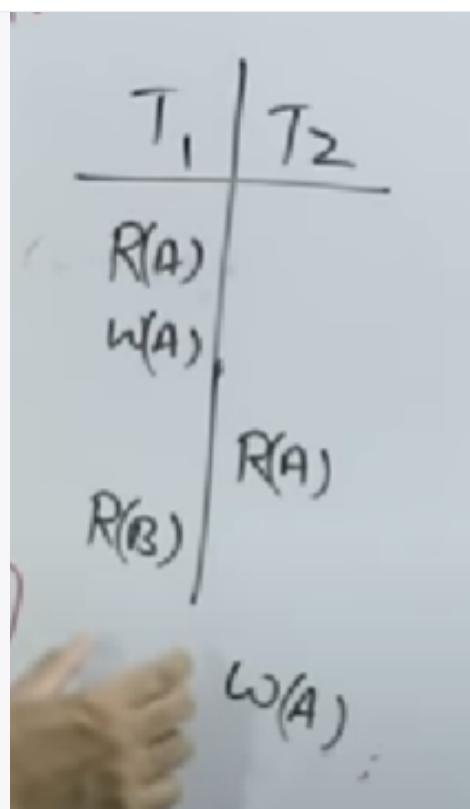
s'	T_1	T_2
	$R(A)$	
	$W(A)$	
	$R(B)$	
	$R(A)$	$W(A)$



we will check the adjacent pairs, if they are non-conflicting we can change their positions.



after changing the position the schedule will look like :



then again we will check the adjacent pairs.

as R(A) AND R(B) are non conflicting we will swap their positions.

now the schedule become

int

$$S \equiv S'$$

S		S'	
T_1	T_2	T_1	T_2
$R(A)$		$R(A)$	
$W(A)$		$W(A)$	
$R(B)$		$R(B)$	
	$R(A)$ $W(A)$		$R(A)$ $W(A)$

Adjancency

so S and S' are conflict equivalent schedules.

⇒ if there is a case of conflicting pairs like $R(A)$ and $W(A)$, there will be no swapping and no change in position.

⇒ for any schedule whose conflict equivalent exists, it is serializable.

Q.

Lec-78: Conflict Serializability | Precedence Graph | Transaction | DBMS

Conflict Serializability

$\equiv S$

Precedence graph

- Check Conflict pairs in other transactions and draw edges

Loop / Cycle

No loop / Cycle

Conflict Serializable

Serializable

Consistent

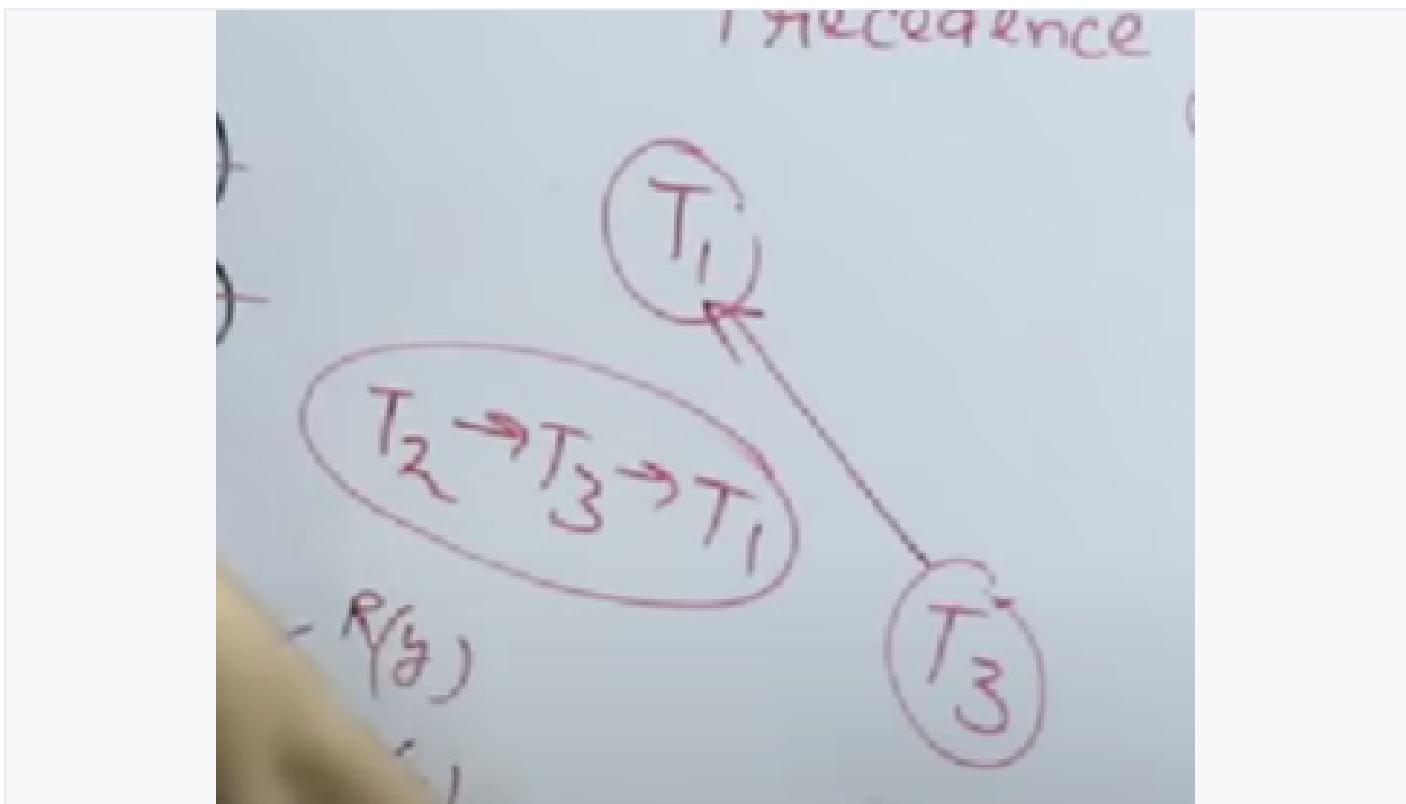
9:38 / 12:17

to know the sequence of the serial schedule we will take the graph and take the transaction with minimum indegree (number of arrows coming inward, starting with 0) and it will be the sequence. as in above example:

T_2 as indegree 0, it will be first and taken out of graph

now there will be two transaction left.

now,



T_3 has indegree = 0, it will be second

and last is T_1 .

so the sequence will be $T_2 \rightarrow T_3 \rightarrow T_1$.

⇒ if in the graph there is a loop, then it is not **conflict serializable**.

but we cannot say it is not serializable, so we use view serializable.

Q.

A handwritten note on a whiteboard. On the left, a student's face is visible. The note includes a transaction schedule table and some handwritten text.

$W_A < R(A)$

$A = 100$ S

	T_1	T_2	T_3
$R(A)$	100		
$W(A)$		$A = A - 40$ $W(A) - 60$	
$W(A)$	$A = A - 40$ $W(A) = 20$		
O		$A = A - 20$ $W(A) = 0$	

View Check whether Schedule is Conflict Serializable
Or not ?

$A = 100$

	T_1	T_2	T_3
$R(A)$	100		
$W(A)$		60	
$W(A)$		20	
O		0	

Loop
↓
Non Answerable + Serializable → \times Answers

in this example as the change of position of $W(A)$ of T_1 and T_2 does not affect the output

This is view serializability

if the output operation states is fixed we can interchange the position of in between operation in view serialization.

Concurrency control protocols

the basic aim is to achieve serializability and recoverability and make concurrent (parallel) schedules serial.

achieve by using **locking protocol**.

'Shared - Exclusive Locking'

→ Shared Lock (S) ⇒ if transaction locked data item in Shared mode then allowed to read only.
→ Exclusive Lock (X) ⇒ if transaction locked data item in Exclusive mode then allowed to Read and write both.

* Problems in S/X locking

- 1) May not sufficient to produce only Serializable schedule.
- 2) May not free from livelock.
- 3) May not free from deadlock.
- 4) May not free from starvation.

Request

S	X
Yes	No
No	No

T₁

S(A)
R(A)
U(A)

T₂

X(A)
R(A)
W(A)
U(A)

grant S (A)

R-R

R(A)

R(A), W(A)

R(A), W(A)

R(A)

Varun Singla //

Introduction >

2 phase locking

use to achieve serializability.

the transaction who is following 2PL is always serializable.

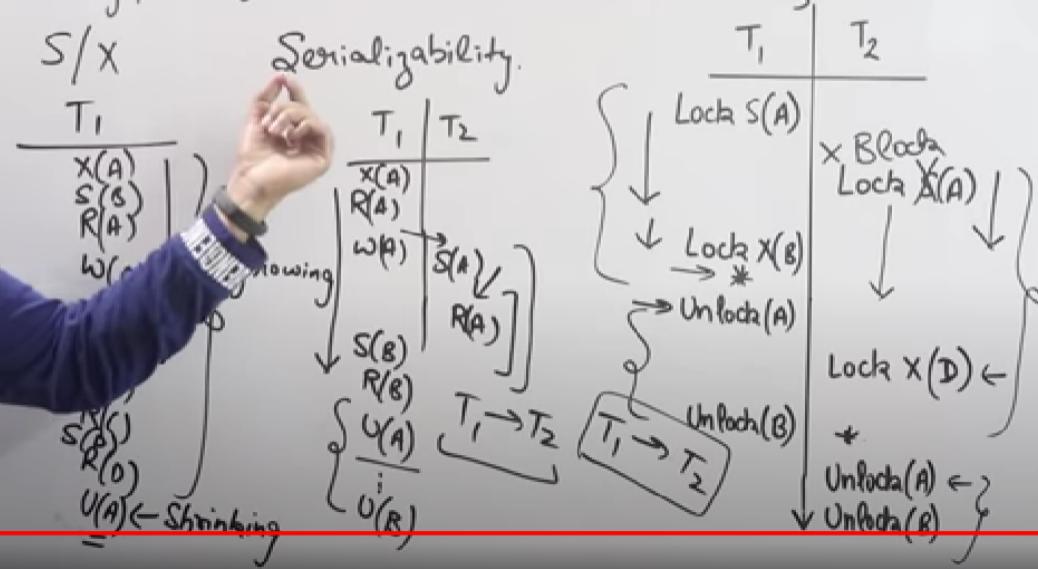
sequence can be given by lock point.

the point just before the first release of the lock in transaction. whose lock point comes first will start the sequence and the sequel will go on.

2-Phase Locking (2PL)

→ Growing phase: locks are acquired and no locks are released.

→ Shrinking phase: locks are released and no locks are acquired.



2PL (2 phase locking)

Advantages: Always ensures Serializability

Drawbacks: May not free from inconsistency

Not free from deadlocks

Not free from starvation

Not free from Cascading Rollback

Strict 2PL: It should satisfy the basic 2PL and all exclusive locks should hold until Commit / Abort

Rigorous 2PL: It should satisfy the basic 2PL and all shared, exclusive locks should hold until commit / Abort.

these two are recoverable and free from cascading rollback(will be always cascadeless).

conservative 2PL : in this a transaction will take all the locks for the operations. so it is free from starvation and deadlock.

it is not practical in real life.

