15/4/2024

4NF

Multi Valued dependencies

eid	dept	hobby
1	CS	Cricket
	IT	Singing
	Mech	Cricket
2	CS	Singing
	IT	Dancing

{ eid → dept, eid → hobby } — Multivalued Dependencies

↓

eid	dept	hobby
1	CS	Cricket
1	IT	Singing
1	Mech	Cricket
2	CS	Singing
2	IT	Dancing

Formal Definition ⇒

(R)

$x \longrightarrow\!\!\!\!\!\rightarrow y$

tuples ⇒ $(t1, t2, t3, t4)$

$Z = R - (X \cup Y)$

$t_3[x] = t_4[x] = t_1[x] = t_2[x]$

$t_3[Y] = t_1[Y]$ & $t_4[Y] = t_2[Y]$

$t_3[Z] = t_2[Z]$ & $t_4[Z] = t_1[Z]$

x: eid	y: dept	z: hobby
1	CS	Cricket
1	IT	Singing
1	CS	Cricket
1	IT	Cricket

{ x → y, x → z }

OR

{ x → y | z }

4NF ⇒ — BCNF

— No-Non-trivial MVDs

An MVD is called as trivial MVD iff

① Y is subset of X

OR

② $X \cup Y = R$

→ A table having key of all attributes will always be in BCNF

Like in our example (eid, dept, hobby) will be in BCNF.

→ If a table only have Two attributes than the table will be in BCNF.

(Lowest)

→ A Relation that is not in 4NF due to non-trivial MVDs must be decomposed into set of relations in 4NF.

eg ⇒ {eid, dept} {eid, hobby}

5NF {PJNF}

{Project-Join Normal Form}

→ 4NF

→ No Join Dependency

If we decompose any Relation and Join them back again it must be lossless Join.

Transaction

→ Set of statements { which must ~~be~~ execute o together or NOT }

Properties of Transaction ⇒

ACID

- Atomicity
- Consistency.
- ~~Integrity~~ Isolation
- Durability.

17/4/2024

Levels of Isolation

(1) → Lost update ⇒

1. A reads row1
2. B reads row1
3. A updates row1
4. B updates row1

A	B
Salary = 5000	
	5000
debit = 500 → 4500	
	Credit = 1000
	6000

{ Lost update as A's updated value was not ~~used~~ by B, it used B's previous value

(2) Dirty Read ⇒

1. A inserted a row1
2. B read row1
3. A rolled back
4. B is having a row which is not available

A	B
insert row1	
	Read Row1
Rollback	

{ B here still has Row1 which doesn't exists

(3) Non-Repeatable Read ⇒

1. A read row of data
2. B modify & updates that row
3. A again reads that row & gets different result

A	B
read a row	
	modify that Row
Re-Read { Now Confused }	

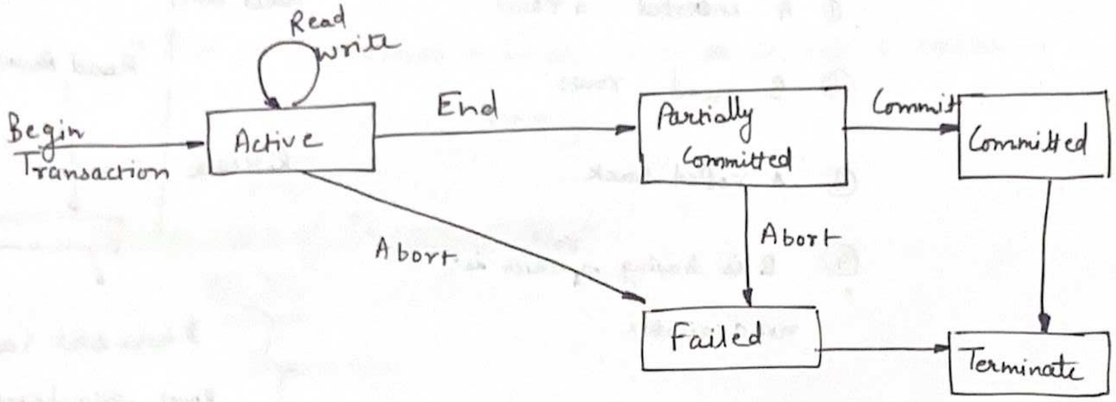
↓ Because value is different from previous.

(4) Phantoms ⇒

1. A reads 2 rows based on a query with where clause
2. B inserts a row which falls under above where clause
3. A ~~reads~~ runs the query again but gets different no. of rows

A	B
query and got 2 rows	
	Inserted 2 rows
Again, runs the query getting 3 rows	

States of Transactions



→ Active ⇒ The initial state, transaction will be in this state while execution.

→ Partially Committed = When final statement has been executed.

→ Failed = When discovered that normal execution can not proceed.

→ Committed = After Committing the transaction on successful Completion.

→ Terminated = After the transaction is completed due to Success or failure.

System Log =

Entries ⇒ Some sample entries ↓

[State_transaction , transaction_id]

[write_item, T_id , data_item , old_value , new_value]

[read_item, T_id, data_item]

[commit , T_id]

[abort, T_id]

⋮

Commit Point. ⇒ Point where we committed

Concurrent Execution ⇒ - When more than one transactions are executing simultaneously.
- Main cause of anomalies.

Concurrent Execution

→ Schedule ⇒ Sequence of statements of transaction.

- Serial Schedule
- Non-Serial Schedule.

→ Serial Sch. ⇒ All transaction are executing one by one no concurrent ~~tran~~ execution

Example ⇒

No anomalies due to concurrent execution

T_1	T_2
① R(x)	
② W(x)	
	③ R(x)
	④ W(x)

Non-Serial Schedule ⇒ where more than 1 transactions are interleaved.

T_1	T_2
① R(x)	
	② R(x)
③ W(x)	
	④ W(x)

Non-Serial Sch.

- Serializable
  - You can process the transaction in serial and result will not differ
- Non-Serializable
  - Cannot be executed in serial.

Non-Serial Schedule Categories

→ Recoverable Schedule ⇒

- Only reads are allowed before write operations on the same data.

eg① $R_1(x)\ W_1(x)\ R_2(x)\ R_1(y)\ R_2(y)\ W_2(x)\ W_1(y)\ C_1\ C_2$

Here $T_i \rightarrow T_j \Rightarrow C_1 \Rightarrow C_2$ Transaction T_1 is executed before T_2, hence no chance of conflicting operations.

$R_1(x)$ appears before $W_1(x)$ and T_1 is committed before T_2

i.e. Completion of 1st transaction performed the first update on data item x.

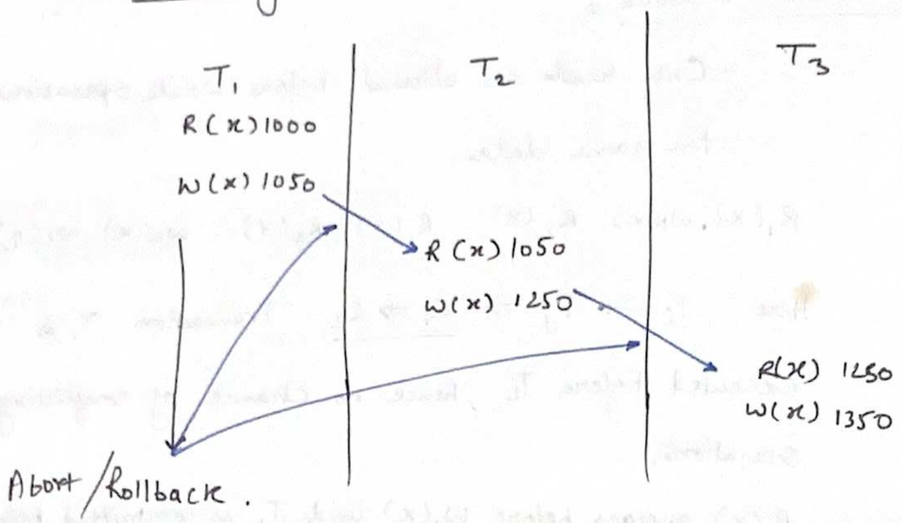
eg② ⇒ $R_1(x)\ R_2(x)\ R_1(z)\ R_3(x)\ R_3(y)\ W_1(x)\ W_3(y)\ R_2(y)\ W_2(z)\ W_2(y)\ C_1, C_2, C_3$

Unrecoverable ⇒ because write on y is done by 3rd transaction before 2nd, but Commit of 2nd is before 3rd.

** Sequence of Commit must be in order as sequence of writes

- We should check for writes from the end

Cascading Schedule ⇒



When failure one one transaction leads to rollback / abort other transaction.

Cascadeless

T_1	T_2	T_3
R(x)		
W(x)		
C		
	R(x)	
	W(x)	
	C	
		R(x)
		W(x)
		C
Abort		

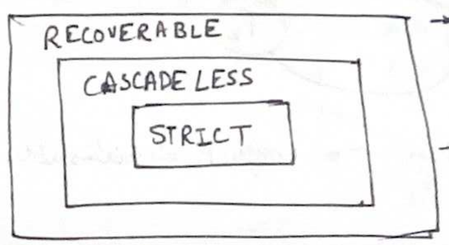
Can't Rollback after Commit

Strict Schedule

- Conflicting Operations ⇒ 2 Operations are set to be conflicting operations if all following conditions satisfy.-
  - They belong to two different transaction.
  - They operate on same data item.
  - Atleast one of them is a write operation.

- A schedule is said to be strict schedule if a write operation of T_i ~~prceted~~ proceeds a conflicting operation of T_j (Read or write), the commit or abort of T_i also proceeds that conflicting operations of T_j.

T_i	T_j
R(x)	
	R(x)
W(x)	
Commit	
	W(x)



→ All strict sch. are Cascadeless and Recoverable

→ All Cascadeless sch. are Recoverable.

Serializable

→ A schedule is known as serializable, if it is equivalent to some serial schedule of same transaction.

→ Conflict Serializability

→ View Serializability.

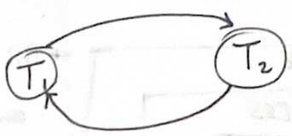
Conflict Serializability ⇒

↑ to check we make precedence Graph

→ Precedence Graph ⇒

- All the transactions are made nodes.
- Conflicting Operations are made edges.
  (Read → write)
  (write → Read)
  (write → Write)

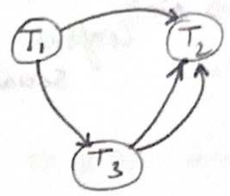
eg ⇒ S1 ⇒ $r_1(x)\ r_1(y)\ w_2(x)\ w_1(x)\ r_2(y)$



⇒ If cycle, then no conflict serialisable

S2 = $r_1(x), r_3(y), w_1(x), w_2(y), r_3(x), w_2(x)$

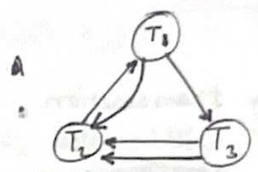
$r_1(x)$ will not be considered as $w_1(x)$ is there



No Loop

→ Conflict Serializable

S3 = $r_1(x), r_3(y), r_2(x), w_1(x), w_3(y), r_3(x), w_2(x)$

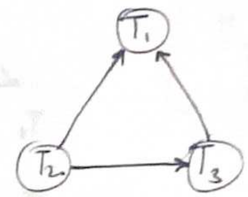


⇒ Not Conflict Serializable

S4 = $w_1(x), R_2(y), R_1(y), R_2(x)$

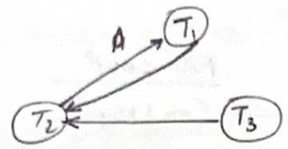
S5 =

T_1	T_2	T_3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		



⇒ Conflict Serializable

S6= $R_1(A), R_2(A), R_1(B), R_2(B), R_3(B), W_1(A), W_2(B)$



⇒ Not Conflict Serializable

19/4/2024

Conflict Equivalance

Swapping Non-conflicting transaction of any schedule, will be called those schedule are known as Conflict Equivalence Schedule.

→ If by swapping it becomes serial or transaction are in sequence then it is conflict serializable.

Example ⇒

$R_1(A)$. $W1(A)$ $R_2(A)$ $W_2(A)$ $R_1(B)$ $W_1(B)$ $R_2(B)$ $W_2(B)$

we want serial ⇒ $T_1 \rightarrow T_2$ OR $T_2 \rightarrow T_1$

Let's see what are the operation in each transaction

$T_1 \Rightarrow R_1(A)\ W1(A)\ R_1(B)\ W(B)$

$T_2 \Rightarrow R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$

S2 = $R_1(A)\ W_1(A)\ R_1(B)\ W_2(A)\ R_2(A)\ W_1(B)\ R_2(B)\ W_2(B)$

S3 = $R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)\ R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$

Conflicting Serializable

S1 & S2 are Conflict equivalence

S1 & S3 & S2 are Conflict equivalence.

S1= $R_2(A)\ W_2(A)\ R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)\ R_2(B)\ W_2(B)$

S2= $R_2(A)\ W_2(A)\ R_2(B)\ W_1(A)\ R_1(B)\ W_1(B)\ R_1(A)\ W_2(B)$

S3= $R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)\ R_1(B)\ W_1(B)\ R_1(A)\ W_1(A)$

(from S1) Same / Not Same

$T_1 = R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)$

$T_2 = R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$

S1 & S2 & S3 are Conflict equivalence

S1 is not Conflict Serializable.

S1= $R_1(A)\ W_1(A)\ R_2(A)\ W_2(A)\ R_1(B)\ W_1(B)\ R_2(B)\ W_2(B)$

① find Conflict equivalence Sch.

② If S is Conflict Serializable

$T_1 = R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)$

$T_2 = R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$

S2= $R_1(A)\ W_1(A)\ R_1(B)\ W_2(A)\ R_2(A)\ W_1(B)\ R_2(B)\ W_2(B)$

S3= $R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)\ R_2(A)\ W_2(A)\ R_2(B)\ W_2(B)$

T_1 T_2

S1 is Conflict Serializable & S1 & S2 & S3 are conflict equivalence.

Q⇒ Consider the following schedules involving 2 transactions. which one of following statement is true :→

S1= $R_1(x)\ R_1(y)\ R_2(x)\ R_2(y)\ W_2(y)\ W_1(x)$

S2= $R_1(x)\ R_2(x)\ R_2(y)\ W_2(y)\ R_1(y)\ W_1(x)$

(A) Both S1 & S2 are conflict serializable

(B) only S1 is CS

(C) Only S2 is CS

(D) None.

S1 ⇒ $T_1 \rightleftarrows T_2$

S2= $T_1 \rightleftarrows T_2$

Only S2 is conflict serializable.

⇒ If a schedule is conflict Serializable then it will be view serializable but if a schedule is view serializable it may or may not be conflict serializable

⇒ A schedule is view serializable if it is view equivalent to its serial schedule