4

Emp No → Dept_Nam → Dept_location

empNo → Dept_location

To make it in 3NF we decompose and remove the transitive dependency so we convert the given table in 3NF decompose into sub table such as

Table 1 employee_depatment

emp_No	E_Nam	Sal	Dep Name
1001	Vishal	7800	Accounts
1002	Amit	8000	Sales
1003	Anuj	10000	Accounts
1004	Vikas	4500	Sales
1005	Rohit	6500	Store

Table 2 Depatment_location Remove Redundancy.

Dept Nam	Dept location
Accounts	102
Sales	104
Store	106

Table converted in 3NF.

Boyce codd Normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every Functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF the table should be in 3NF and for every FD LHS is super key.

Eg: Let's assume there is a company where employee work in more than one department.

Employee table

emp-id	emp-country	Emp-Dept	Dept-type	emp-dept No.
264	India	designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows

Emp-Id → Emp-country

Emp-Dept → {Dept-type, emp Dept No}

Candidate key : {emp-id, emp-Dept}

The table is not in BCNF, because neither emp-dept nor emp-id alone are keys. To convert the given table into BCNF, we decompose it into three tables.

4

emp_country table — Remove Redundancy.

emp_id	emp_country
264	India
364	UK

emp_dept table

emp_dept	dept type	emp_dept No.
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549.

emp_Dept mapping table: — BCNF.

emp id	type	emp_Dept No.
D394		283
D394		300
D283		232
D283		549.

Functional Dependencies

emp_id → emp_country

emp_dept → {dept_type, emp_dept No}.

Candidate key.

For the first table : emp id ✓

" " 2nd " : emp_dept ✓

" " Third " : {emp id, emp_dept}.

Now this is in BCNF because left side part of both the functional dependencies is a key.

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

$$X \longrightarrow\!\!\!\!\rightarrow Y.$$

It will be read as "there is a multi valued dependency of Y" or "X multi-determine Y".

Example - Faculty.

Faculty	subject	committee
Aman	DBMS	Placement
Aman	Java	"
Aman	C	"
Aman	DBMS	Scholarship
Aman	Java	"
Aman	C	"

The given Faculty table is in 3NF, but the subject and committee are two independent entity Here there is no relationship b/w subject and committee.

In the Faculty relation, a faculty with Faculty name Aman contains three subject DBMS, Java and C and two committe placement and scholarship

4- Is so there is multivalued dependency on faculty name which leads to unnecessary repetition of data.

Aman → Placement
Aman → Scholarship.

so to make the above table into 4NF, we can decompose it into two tables.

Tab 1 Faculty - Course — Remove Redundancy.

Faculty	Subject
Aman	DBMS
Aman	Java
Aman	C

same name value

N of

Table 2 Faculty — committee → 4NF.

Faculty	committee
Aman	Placement
Aman	Scholarship.

Fifth Normal Form (5NF)

- 5th is also known as Project join Normal Form.
- A relation is in Fifth Normal Form if it is in 4NF and won't have to 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)

Faculty	Subject	Committe.
Aman	DBMS	Placement
Aman	Java	"
Aman	C	"
Aman	DBMS	Scholarship
Aman	Java	"
Aman	C	"

The given table is not in 4NF and 5NF First we convert it in 4NF with converting it two sub tables

4. Table 1 Faculty - Subject

Faculty	Subject
Aman	DBMS
Aman	Java
Aman	C

Table 2 Faculty - committee

Faculty	committee
Aman	Placement
Aman	Scholarship

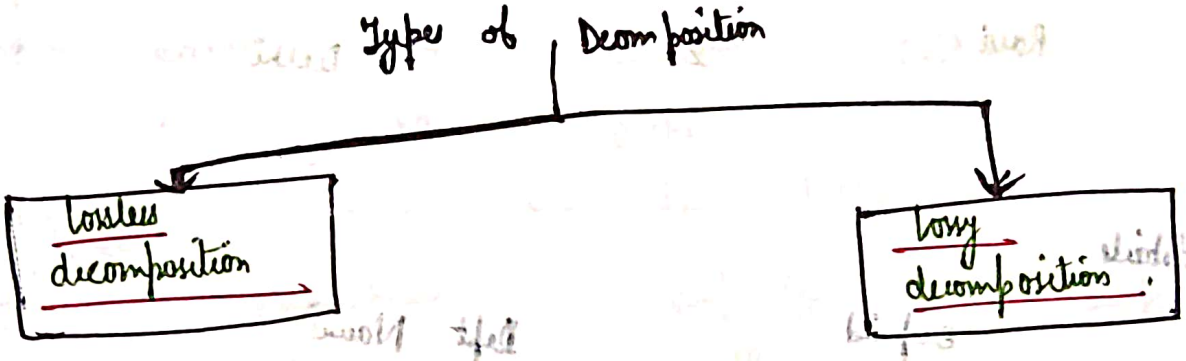
To convert it in 5NF we join both Table 1 and Table 2 if it give the result same as original table (Faculty) then it's in 5NF Other wise Not in 5NF.

Faculty	Subject	committee
Aman	DBMS	Placement
Aman	DBMS	Scholarship
Aman	Java	Placement
Aman	Java	Scholarship
Aman	C	Placement
Aman	C	Scholarship

So it is in 5NF is equal to the original table.

Relational Database Decomposition

- When a relation in the relational model is not in appropriate normal form the decomposition of a relation is required.
- In a DB, it breaks the table into multiple tables.
- If the relation has no proper decomposition then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies and redundancy.



lossless decomposition

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantee that the join of relation will result in the same relation as it was decomposed
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

4. Ex – Emp – Info

emp_Id	Emp_Name	Emp_Age	Emp location	Dept Id	Dept No.
E001	Komal	29	Haridwar	Dep1	Operations
E002	Karan	32	Deheradun	Dept2	HR
E003	Ravi	22	Delhi	Dept3	Finance

Decompose the above details into two tables.

(i) Emp details

Emp Id	Emp Name	Emp Age	Emp location
E001	Komal	29	Haridwar
E002	Karan	32	Deheradun
E003	Ravi	22	Delhi

2. Dept details

Dept Id	emp Id	Dept Name
Dept 1	E001	operation
Dept 2	E002	HR
Dept 3	E003	Finance

Now, Natural Join is applied on the above two tables.

The result will be (jab combine karenge to original table mil jayega).

Emp Id	Emp Name	Emp Age	Emp location	Dep Id	Dep Name
E001	Komal	29	Haridwar	Dep1	Operations
E002	Karan	32	Deheradun	Dep2	HR
E003	Ravi	22	Delhi	Dep3	Finance

Therfore the above relation had lossless decomposition there is no loss of information if we join all Decompose tables.

Lossy Decomposition

↳ When a Relation is decomped into two or more relational schemas, the loss of information unavailable when the original relation is retrieved.

Let us see an example

Emp-Id	Emp Name	Emp - Age	Emp location	Dept Id	Dept Name
E001	Kanal	29	Haridwar	Dept 1	Operations
E002	Karan	32	Dehradun	Dept 2	HR
E003	Ravi	22	Delhi	Dept 3	Finance

Decompose the table into two tables.

<Emp details>

Emp Id	Emp - Name	Emp Age	Emp location
E001	Kanal	29	Haridwar
E002	Karan	32	Dehradun
E003	Ravi	22	Delhi

<Dept Details>

Dept Id	Dept Name
Dept 1	operation
Dept 2	HR
Dept 3	Finance

PK is not present.

Now you won't be able to join the above tables since Emp-Id (PK) is not part of dept details relation.

Therfore the above relation has lossy decomposition

Transaction Processing Concepts...

Transaction system.

- Collection of operations that form a single logical unit of work are called transaction.
- A transaction is a unit of Program execution that accesses and possibly updates various data items.
- Transaction is defined as a logical unit of DB processing that includes one or more DB access operations.

Transaction Access data using two operations.

(i) Read (X)
- Read operation is used to read the value of Account X from the DB and stores it in a buffer in main memory.

(ii) Write (X)
- Write operation is used to write the value back to the DB from the buffer.

Let us take an example to debit transaction from an account which consists of following operations: X=1000

		X=1000	Y=1000
		T_1	T_2
1	R(X) :	read (X)	Read(Y)
2	X = X-500	X = X-500	Y = Y+1000
3	W (X)	Write (X) (X=500)	Write Y (Y=1000)

Let us Assum the Value of X before starting of transaction is 4000.

↳ The First operation read X is value from DB and stores it in a buffer

↳ The second operation will decrease the value of X by 500 so buffer will contain 3500

↳ The third operation will write the buffer value to the DB, 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 transaction may fail before finished all the operation in the set.

For example: If the given transaction the debit transaction fails after Executing operation 2 then X's value will remain 400 in the DB which is not acceptable by the bank. (Rollback ho jayega)

To solve this Problem we have two important operation

1. commit
   ↳ It is used to save the work done permanently.
2. Rollback
   ↳ It is used to undo the work done.

ACID Properties of Transaction

- Transaction have 4 basic properties which are called ACID properties.
- These are used to maintain consistency in a DB before and after the transaction.

Properties of transaction

Atomicity Consistency Isolation Durability

(i) 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.

Ex:- Let us assume that the 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.

T_1	T_2
Read (A)	Read (B)
A = A - 100	B = B + 100
Write (A)	Write (B)

After completion of the transaction A consists of Rs 500 and B consists of Rs 400.

If the transaction T fails after the completion of transaction T_1 but before completion T_2, then the amount will be deducted from A but not added to B. In order to ensure correctness of DB state to ensure Correctness of DB state the transaction must be executed in entirely.

2. Consistency (data loss nahi hona chaiye).

- The integrity constraints are maintained so that the DB is consistent before and after the transaction.
- The execution of a transaction will leave a DB in either its prior stable state or a new stable state.
- The consistency property of DB states that every transaction sees a consistent DB instance.
- The transaction is used to transform the DB from one consistent state to another consistent state.

For ex: Lets 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.

4.

T_1	T_2
Read (A)	Read(B)
A = A - 100	B = B + 100
Write (A)	Write (B)

The total amount must be maintains before or after the transaction.

Total before T occurs = 600 + 300 = 900
Total after T occurs = 500 + 400 = 900
] No change

Therefore the DB is consistent. In the case when T_1 is completed but T_2 fails then inconsistency will occur.

3 Isolation (does not depend on any state).

It means 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 T_2 until the transaction T_1 ends.

The concurrency control subsystem of the DBMS enforced the isolation property.

4. Durability

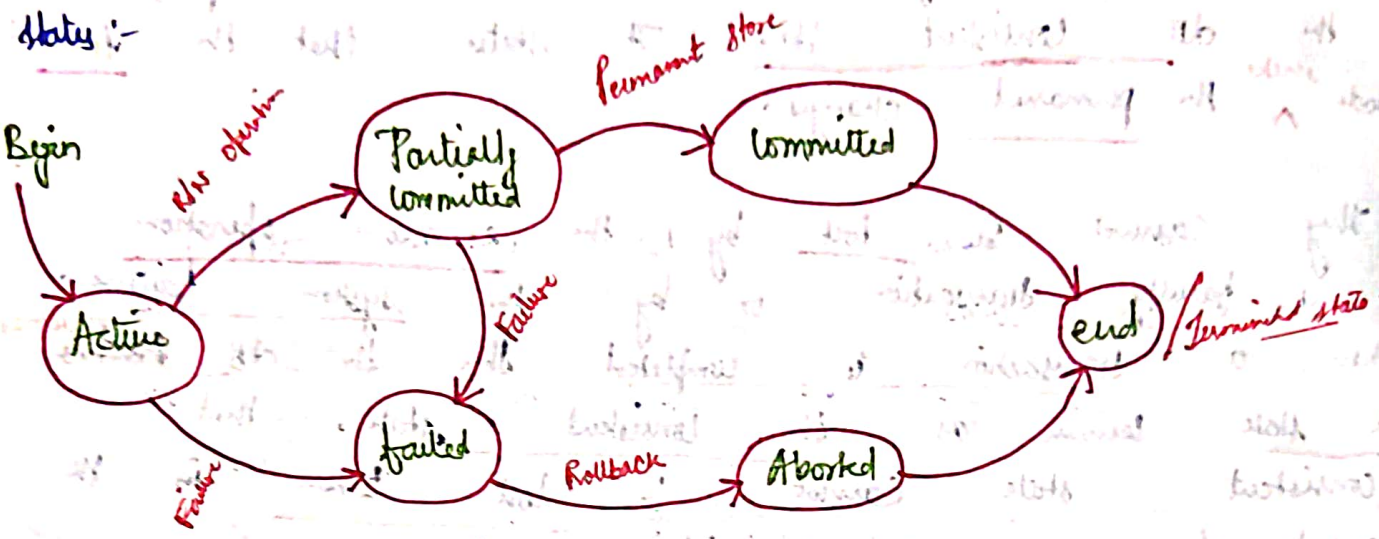
Is The durability property is used to indicate the performance of the dbs consistent state. It states that the transaction make the permanent changes.

Is They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed then the dbs reaches a state known as the consistent state. That consistent state cannot be lost Even in the event of a system's failure.

Is The recovery subsystem of the DBMS has the responsibility of durability property.

States of Transaction

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



1. Active State (initial state)

↳ It is the first state of every transaction. In this state, the transaction is being executed.

For eg:- Insertion or deletion or updating a record is done here : But all the records are still not saved to the DB.

2. Partially committed (final state mein aa gaya hai but execute nhi hua hai).

↳ In the Partially committed state, a transaction executes its final operation, but the data is still not saved to the DB.

↳ In the total Mark Calculation example a final display of the total Marks step is executed in this state.

3 committed (data is executed successfully).

↳ A transaction is said to be in committed state if it executes all its operations successfully.

↳ In this state all the effects are now permanently saved on the db system.

4. Failed state (Kabi bhi fail ho sakta to fail) network error, or any thing

↳ If any of the checks made by the db recovery system fails, then the transaction is said to be in the failed state.

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

5. Aborted (Check how to load bhi to fail state main ho so aborted)

↳ If any of the checks fails and the transaction has reached a failed state then the db recovery system will make sure that the db is in its previous consistent state. If not then it will abort or rollback the transaction to bring the db into a consistent state.

↳ After aborting the transaction, the db recovery module will select one of the two operations

(i) Re-start the transaction
or
(ii) kill the transaction.

} After aborting

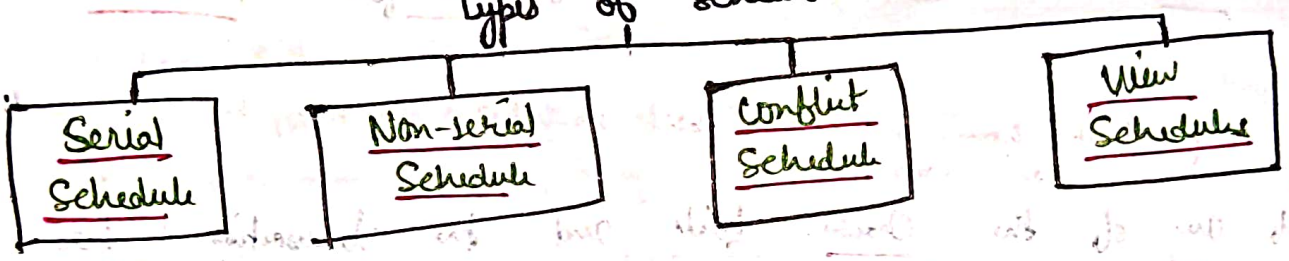
Serializability of Schedules

Schedule

↳ When several transaction are executy concurrently 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 correct when transaction executions have interleaving of their operation in the schedules.

Types of Schedules:



1. Serial Schedule

↳ It is 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

This schedule A is called Serial schedule entire transaction are perform in serial order T_1 and then T_2 or T_3 and then T_1

2. Non- Serial Schedule

↳ If interleaving of operation is allowed, then there will be non- serial schedule. (Parallel executing).

↳ A Schedule 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 Schedule

↳ A Schedule is called conflict serializability if after swapping of non- conflicting operations it can ~~swapping of non conflicting~~ transform into a serial Schedule.