# Normalization

## Functional Dependency

- The functional Dependency is a relationship that Exists between two attributes.
- It typically Exists between the primary key and non-key attributes within a table.

$$X \longrightarrow y$$

The Left Side of FD is known as a determinant, The right Side of the production is known as a dependent.

For Example ⇒ Assume we have an Employee table with attributes: Emp-Id, Emp-Name, Emp-Address.

Here Emp-Id attribute Can uniquely identify the Emp-Name attribute of Employee table because if we known the Emp-Id, we Can tell that Employee name associated with it.

functional dependency Can be written as:

Emp-Id ⟶ Emp-Name

We Can Say that Emp-Name is functionally dependent on Emp-Id.

$$\begin{tikzcd}
 & \boxed{\text{Types of function dependency}} \arrow[dl] \arrow[dr] & \\
\boxed{\text{Trivial functional Dependency}} & & \boxed{\text{Non-trivial functional dependency}}
\end{tikzcd}$$

## # Trivial functional dependency ⇒

- A → B has trivial functional dependency if B is a Subset of A.

- The following dependencies are also trivial like: A → A , B → B.

Ex ⇒ Consider a table with two columns Employee-id and Employee-Name.
{ Employee-id, Employee-Name } → Employee-id is a trivial functional dependency as Employee-id is a Subset of { Employee-id, Employee-Name }
Also, Employee-id → Employee-id and Employee-Name → Employee-Name are trivial dependencies too.

## # Non-Trivial functional Dependency ⇒

- A → B has a non-trivial functional dependency if B is not a Subset of A.

- When A intersection B is NULL, then A → B is called as Complete non-trivial

Ex ⇒ Consider a table with three Column Emp-id, Emp-Name and Emp-age.
Emp-id → Emp-Name is a Non trivial functional dependency because Emp-Name is not a Subset of Emp-id.
Similarly
Emp-id, Emp-Name ⟶ Emp-Age is non-trivial because Emp-age is not a Subset of Emp-id, Emp-Name

# Closure of a Set of functional Dependencies

## Inference Rule (IR)

- The inference rule is a type of assertion. It can apply a set of FD (functional dependency) to derive other FD.
- Using the inference Rule, we can derive additional functional dependency from the initial set.

1) Reflexive Rule ($IR_1$) ⇒ In the Reflexive Rule, if Y is a subset of X, then X determines Y

if $Y \subseteq X$ then $X \rightarrow Y$

Example ⇒

$X = \{a, b, c, d, e\}$
$Y = \{a, b, c\}$

2) Augmentation Rule ($IR_2$) ⇒ The augmentation is also called as a partial dependency. In ~~go~~ augmentation, if X determines Y, then XZ determines YZ for any Z.

if $X \rightarrow Y$ then $XZ \rightarrow YZ$

Ex ⇒

for R(ABCD)

if $A \rightarrow B$ then

$AC \rightarrow BC$

3) Transitive Rule ($IR_3$) ⇒ In the transitive Rule, if X determines Y and Y determine Z, then X must also determine Z.

If $X \rightarrow Y$ and $Y \rightarrow Z$ then
$X \rightarrow Z$

4) Union Rule ($IR_4$) ⇒ Union rule Says, if X determines Y and X determines Z, then X must also determine Y and Z.

If $X \rightarrow Y$ and $X \rightarrow Z$ then

$X \rightarrow YZ$

5) DeComposition Rule ($IR_5$) ⇒ Decomposition Rule is also known as project rule. It is the reverse of union rule.
This Rule Says, if X determines Y and Z, then X determines Y and X determines Z Separately.

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

6) Pseudo transitive Rule ($IR_6$) ⇒ In Pseudo transitive Rule, if X determine Y and YZ determine W, then XZ determines W.

if $X \rightarrow Y$ and $YZ \rightarrow W$ then
$XZ \rightarrow W$.

Q→ find the Candidate key of Relation R(A, B, C, D, E, F) with functional dependency

$A \rightarrow C$
$C \rightarrow D$
$D \rightarrow B$
$E \rightarrow F$

(set of attributes whose closure contains all attributes of given relation)

Soln→

S.K — $A B \cancel{C} D E F^+ = \{A, B, C, D, E, F\}$

$A \rightarrow C$

SK — $A B \cancel{D} E F^+ = \{A, B, C, D, E, F\}$

$A \rightarrow C \quad C \rightarrow D$
$A \rightarrow D$ {transitive dependency}

S.k — $A \cancel{B} E F^+ = \{A, B, C, D, E, F\}$

$A \rightarrow C \rightarrow C \rightarrow D \quad D \rightarrow B$
$A \rightarrow B$ transitive dependency

S.K — $A E F^+ = \{A, B, C, D, E, F\}$

$E \rightarrow F$

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

Candidate key

Prime attribute = (A, E) if no prime attribute available to Right hand side of any function dependency so it has only one candidate key ($AE^+$)

Ex ⇒ find the possible Candidate key of the Relation R(A, B, C, D) with functional dependency A → B, B → C, C → A

Sol$^n$ ⇒ S.K $ABCD^+ \longrightarrow \{A, B, C, D\}$

S.K $ACD^+ \longrightarrow \{A, B, C, D\}$

S.K. $AD^+ \longrightarrow \{A, B, C, D\}$

AD is Candidate key

AD prime attribute A, D, prime attribute
A available in Right side of function dependency C → A so another candidate key $\underline{\underline{CD}}$

again C is available in Righ side of functional dependency B → C so another Candidate key

BD.

so Candidatkey = AD, CD, BD

# Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations
- It is used to eliminate undesirable Characteristics like insertion, Update and Deletion Anomalies
- Normalization divides the Larger table into Smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

When we normalize the database, we have four goals:

1) Arranging data into Logical groupings Such that Each group describes a small part of the whole.
2) minimizing the amount of duplicate data, Called redundancy, stored in a database.
3) organizing the data such that, when you modify it, you make the changes only in one place.
4) Building a database in which you can access and manipulate the data quickly and efficiently without Compromissing the integrity of the data in storage.

$$\begin{tikzcd}
 & & \boxed{\text{Types of Normal forms}} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] \arrow[drrr] & & & \\
\boxed{\text{1NF}} & \boxed{\text{2NF}} & \boxed{\text{3NF}} & \boxed{\text{BCNF}} & \boxed{\text{4NF}} & \boxed{\text{5NF}}
\end{tikzcd}$$

Camlin

1NF (first Normal form) ⇒ • A relation is in 1NF if it contains an atomic values.
• It Eliminate Repeating Groups.

2NF (second Normal form) ⇒ • A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
• It Eliminate Partial functional dependency.

3NF (Third Normal form) ⇒ • A Relation will be in 3NF if it is in 2NF and no transitive dependency Exists.
• It Eliminate Transitive dependency.

BCNF (Boyce Codd's normal form) ⇒ • A Stronger Definition of 3NF is known as Boyce Codd's normal form.
• It is also Called 3.5 NF.

4NF (fourth Normal form) ⇒ • A relation will be in 4NF if it is in Boyce Codd's Normal form and has no multi-valued dependency
• Eliminate multi-Values Dependency.

5NF (fifth Normal form) ⇒ • A relation is in 5NF if It is in 4NF and does not Contain any join dependency, joining should be Lossless.
• Eliminate Join Dependency.

# Advantages & Disadvantages of Normalization

## Advantages of Normalization.

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- much more flexible database design.
- fewer indexes per table ensure faster maintenance tasks (index rebuilds).
- Relizes the option of Joining only the tables that are needed.
- A better handle on database security.
- Increase storage efficiency.
- Speed up data access.

## Disadvantages of Normalization.

- You cannot start building the database before knowing what the user needs.
- Requires much CPU, memory and I/O process thus normalized data gives reduced database performance
- Requires more joins to get the desired result. A poorly-written query can bring the database down.
- It is very time-consuming and Difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

## First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First Normal form disallow the multi-valued attribute, Composite attribute and their Combinations.

Ex ⇒ Relation Student is not in 1NF because of multi-Valued attribute ~~Std~~-Phone.

Student table:

| Stud_id | Stud_name | Stud-Phone | Stud_Branch |
|---|---|---|---|
| 11 | Kamal | 7276854823, 3214172829 | IT |
| 12 | Karan | 8762547890 | CS |
| 13 | Ravi | 32456278, 34563578 | ES |
| 14 | Ram | 8176345630 | ME |
| 15 | Komal | 3456873200 | IT |

The Decomposition of the Student table into 1NF as:

| Stud_id | Stud_name | Stud-Phone | Stud_Branch |
|---|---|---|---|
| 11 | Kamal | 7276854823 | IT |
| 11 | Kamal | 3214172829 | IT |
| 12 | Karan | 8762547890 | CS |
| 13 | Ravi | 32456278 | ES |
| 13 | Ravi | 34563578 | ES |
| 14 | Ram | 8176345630 | ME |
| 15 | Komal | 3456873200 | IT |

# Second Normal form (2NF)

- In the 2NF, relation must be in 1NF.
- No attributes of the table should be functionally dependent on only one part of a concatenated primary key.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key.

Ex ⇒ 2NF is based on the concept of full functional Dependency $\boxed{X \longrightarrow Y}$ is a fully function Dependency (FFD) if removal of any attribute (A) from X means that the dependency does not hold any more.

In the Book-order table such as

| order No. | title | Qty | Unit Price |
|---|---|---|---|
| 1 | Computer Network | 1 | 250 |
| 1 | Java | 1 | 275 |
| 1 | DBMS | 2 | 295 |
| 2 | multimedia | 1 | 300 |
| 2 | Data Structure | 1 | 190 |
| 3 | DBMS | 1 | 295 |
| 3 | multimedia | 2 | 300 |
| 3 | Computer Network | 5 | 250 |

It is not in 2NF because it hold Partial function Dependency and fully function dependency.

title ⟶ Unit Price

P.F.D.

and

$$\left.\text{OrderNo, Title} \longrightarrow \text{Qty}\right\} \text{F.F.D}$$

$$\text{OrderNo} \xrightarrow{\times} \text{Qty}$$

$$\text{Title} \xrightarrow{\times} \text{Qty}$$

Now we will convert the given table in 2NF to decompose the given table into two sub table such as:

Order_master table

| Order No. | Title | Qty |
|---|---|---|
| 1 | Computer Network | 1 |
| 1 | Java | 1 |
| 1 | DBMS | 2 |
| 2 | Multimedia | 1 |
| 2 | Data Structure | 1 |
| 3 | Multimedia | 2 |
| 3 | Computer Network | 5 |

Book_Master table

| Title | Unit Price |
|---|---|
| Computer Network | 250 |
| Java | 275 |
| DBMS | 295 |
| Multimedia | 300 |
| Data Structure | 190 |

# Third Normal form (3NF)

- A relation is in 3NF, if it is in 2NF and no non-prime attribute functionally dependent on other non-prime attributes.
- A relation is in 3NF, if it is in 2NF and no attributes of the table should be transitively functionally dependent on the primary key.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for Every non-trivial function dependency X → Y.

1→ X is Super key

2→ Y is a prime attribute suchas Each Element of Y is part of some Candidate key.

Ex ⇒ Employee_Department_Location table.

| EmpNo | E_Name | Sal | Depart_Name | Depart_Location |
|---|---|---|---|---|
| 1001 | Vishal | 7500 | Accounts | 102 |
| 1002 | Amit | 5000 | Sales | 104 |
| 1003 | Anuj | 10000 | Accounts | 102 |
| 1004 | Vikas | 4500 | Sales | 104 |
| 1005 | Sumit | 6500 | Store | 106 |

Table Not in 3NF because it hold the transitive dependency.

$\text{EmpNo} \longrightarrow \text{Dept-Name} \longrightarrow \text{Dept-Location}$

$\text{EmpNo} \longrightarrow \text{Dept-Location}$

To make it in 3NF we decompose and remove the transitive dependency. so we convert the given table in 3NF decompose two sub table Such as :-

(Table1) Employee - Depatment

| EmpNo | E-Name | Sal | Dep_Name |
|---|---|---|---|
| 1001 | Vishal | 7500 | ACCount |
| 1002 | Amit | 5000 | Sales |
| 1003 | Anuj | 10000 | ACCounts |
| 1004 | Vikas | 4500 | Sales |
| 1005 | SUmit | 6500 | Store |

Table 2 Department- Location

| Dept_Name | 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.

Example:- Let's assume there is a Company where Employees 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-type~~, 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.

Emp_Country table:

| 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:

| Emp_id | Emp_Dept |
|---|---|
| D394 | 283 |
| D394 | 300 |
| D283 | 232 |
| D283 | 549 |

functional dependencies:

Emp_id → Emp_Country
Emp_Dept → {Dept_type, Emp_Dept_No}

Candidate keys:

for the first table: Emp_id
for the Second table: Emp_Dept
for the third table: {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

$$X \longrightarrow \longrightarrow Y$$

It will be readers "~~X~~ 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 with faculty name Kailash contains three Subject DBMS, Java and C, and two Committee placement and Scholarship. so there is a multi-valued dependency on faculty name, which leads to unnecessary repetition of data.

$$\text{Kailash} \longrightarrow \longrightarrow \text{Placement}$$
$$\text{Kailash} \longrightarrow \rightarrow \text{Scholarship.}$$

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 |

# 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)

| 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 |

Table 2 faculty - Committee

| faculty | Committee |
|---|---|
| Kailash | Placement |
| Kailash | Scholarship |

To Convert it in 5NF, we join both table1 and table2 if it give the result same as original table (faculty) then it's in 5NF otherwise not in 5 NF.

Table 1 + Table 2

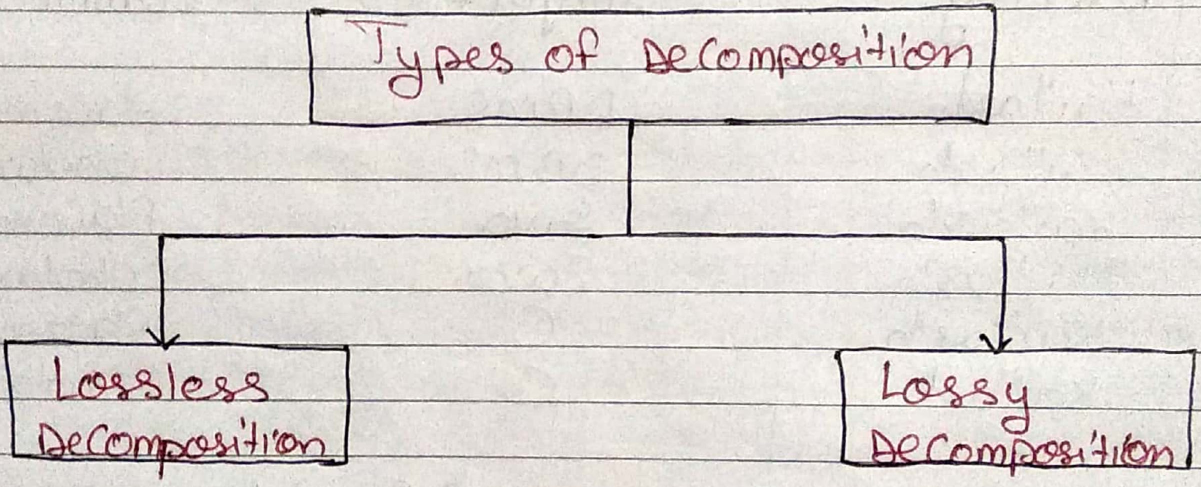
| faculty | Subject | Committee |
|---|---|---|
| Kailash | DBMS | Placement |
| Kailash | DBMS | Scholarship |
| Kailash | Java | Placement |
| Kailash | Java | Scholarship |
| Kailash | C | Placement |
| Kailash | C | Scholarship |

is Equal to original tab

So it is in 5NF

# Relational Decomposition

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, 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 guarantees that the join of relations 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.

Ex ⇒ Emp_Info

| Emp_ID | Emp_Name | Emp_Age | Emp_Location | Dept_ID | Dept_Na |
|---|---|---|---|---|---|
| E001 | Kamal | 29 | Haridwar | Dpt1 | Operation |
| E002 | Karan | 32 | Dehradun | Dpt2 | HR |
| E003 | Ravi | 22 | Delhi | Dpt3 | finance |

Decompose the above table into two tables:

1) EmpDetails

| Emp_ID | Emp_Name | Emp_Age | Emp_Location |
|---|---|---|---|
| E001 | Kamal | 29 | Haridwar |
| E002 | Karan | 32 | Dehradun |
| E003 | Ravi | 22 | Delhi |

2) Dept Details

| Dept_ID | Emp_ID | Dept_Name |
|---|---|---|
| Dpt 1 | E001 | Operation |
| Dpt 2 | E002 | HR |
| Dpt 3 | E003 | finance |

Now, Natural Join is applied on the above two tables :
The result will be

| Emp-ID | Emp-Name | Emp-Age | Emp-Location | Dept-ID | Dept-Name |
|---|---|---|---|---|---|
| E001 | Kamal | 29 | Haridwar | Dpt1 | operations |
| E002 | Karan | 32 | Dehradun | Dpt2 | HR |
| E003 | Ravi | 22 | Delhi | Dpt3 | finance |

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

Lossy Decomposition ⇒ when a relation is decomposed into two or more relational Schemas, the Loss of information unavoidable when the original relation is retrieved.
Let us see an Example -
Ex ⇒
EmpInfo

| Emp-ID | Emp-Name | Emp-Age | Emp-Location | Dept-ID | Dept-Name |
|---|---|---|---|---|---|
| E001 | Kamal | 29 | Haridwar | Dpt1 | operations |
| E002 | Karan | 32 | Dehradun | Dpt2 | HR |
| E003 | Ravi | 22 | Delhi | Dpt3 | finance |

Decompose the table into two tables -

~~EmpInfo~~ <EmpDetails>

| Emp-ID | Emp-Name | Emp-Age | Emp-Location |
|---|---|---|---|
| E001 | Kamal | 29 | Haridwar |
| E002 | Karan | 32 | Dehradun |
| E003 | Ravi | 22 | Delhi |

<Dept Details>

| Dept-ID | Dept-Name |
|---|---|
| Dpt1 | Operations |
| Dpt2 | HR |
| Dpt 3 | finance |

Now, you won't be able to join the above tables, Since Emp-ID is not part of DeptDetails relation.
Therefore, the above relation has Lossy Decomposition.

Q.> Consider the Schema $S = (V, W, X, Y, Z)$ suppose the following F.D hold:

$Z \rightarrow V$
$W \rightarrow Y$
$XY \rightarrow Z$
$V \rightarrow WX$

State whether the following decomposition Scheme S is lossless join decomposition or Loss decomposition.

i) $S1 = (V, W, X)$
$S2 = (V, Y, Z)$

ii) $S1 = (V, W, X)$
$S_2 = (X Y Z)$

Ans=> $R_1$ & $R_2$ are two relation is Lossless decomposition if

$R_1 \cap R_2 \longrightarrow R_1$

$R_1 \cap R_2 \longrightarrow R_2$

i) $S_1 = (V, W, X)$

$S_2 = (V, Y, Z)$

$S_1 \cap S_2 = V \longrightarrow S_1$

$V \rightarrow S_2$

$V \rightarrow V$
$V \rightarrow WX$ }

$V \rightarrow VWX$

$V \rightarrow S_1$

so the decomposition

$S_1 = (V, W, X)$

$S_2 = V, Y, Z)$ is Lossless.

ii) $S_1 = (V, W, X)$

$S_2 = (X, Y, Z)$

$S_1 \cap S_2 \rightarrow X \rightarrow S_1 \rightarrow VWX$

$X \rightarrow S_2 \rightarrow XYZ$

$X \rightarrow X$

$X \xrightarrow{\times} W$

$\xrightarrow{\times} Y$

$\xrightarrow{\times} Z$

Lossy decomposition because

X not determine V, WX or XYZ

$X \xrightarrow{\times} VWX$

$X \xrightarrow{\times} XYZ$ Lossy decompositio

# 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 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 : X=1000 Y=1000

| | T1 | T2 |
|---|---|---|
| 1) R(X) ; | read(X) | read(Y) |
| 2) X = X-500 ; | X = X-500 | Y=Y+500 |
| 3) W(X) | write(X) | write(Y) |
| | (X = 500) | Y= 1500 |

Let's assume the value of X before Starting of trasaction is 4000.

- The first operation read X's value from database 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'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.

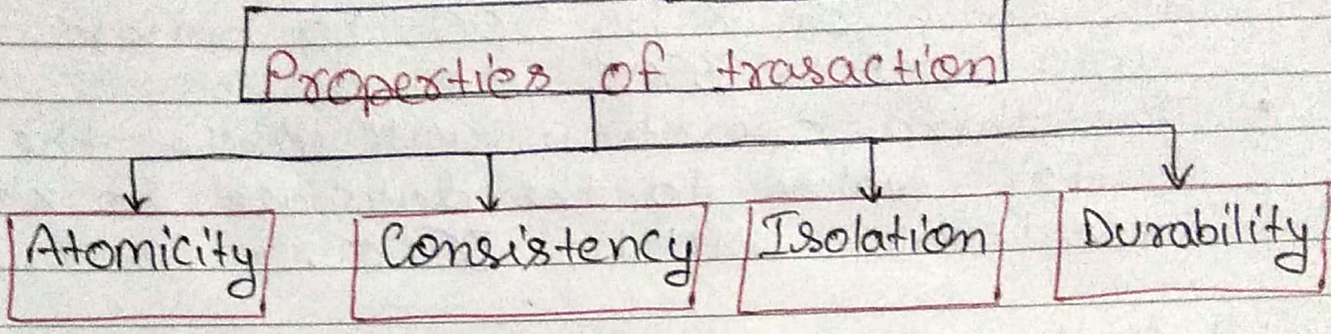
To solve this problem, we have two important operations:

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

2) Rollback:⇒ It is used to undo the work done.

# 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.

• There is no midway suchas 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 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 T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure Correctness of database State, the transaction must be Executed in entirety.

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 T1 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 T1 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 T1 ends.

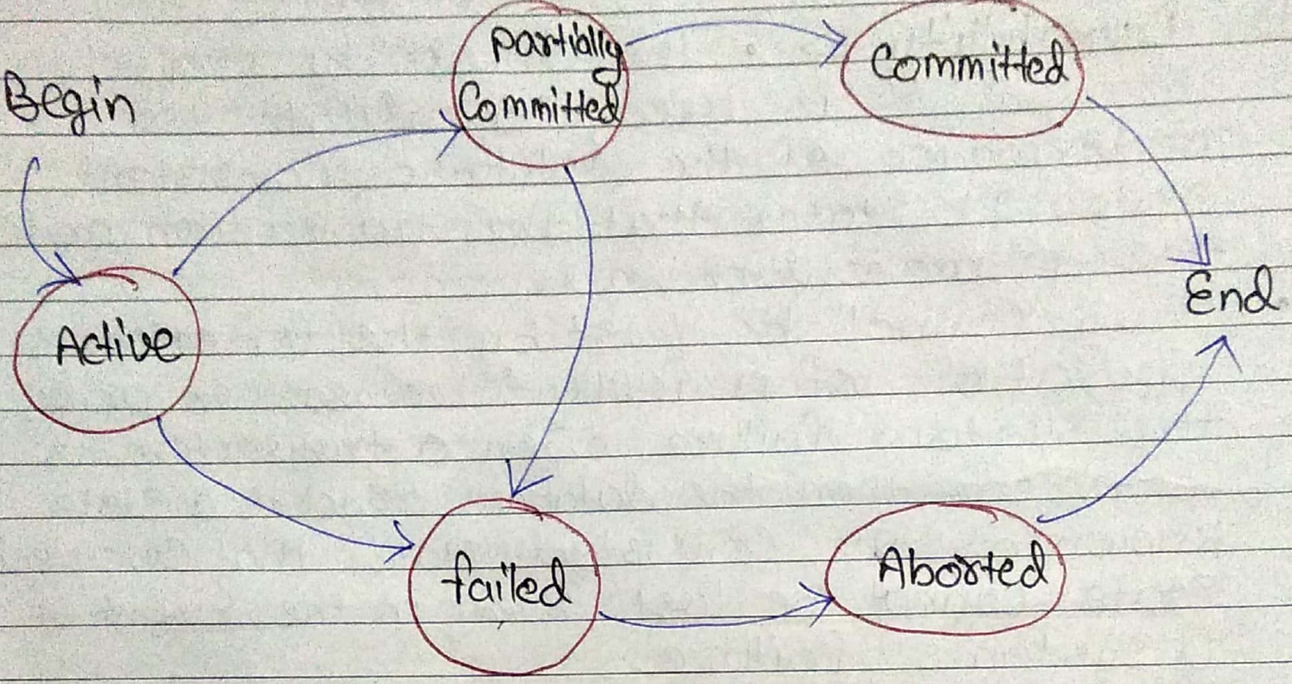
- The Concurrency Control subsystem of the DBMS enfored the isolation property.

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 resposibility of Durability property

# States of Transaction

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



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.

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 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
2) Kill the transaction

# Serializability of Schedules

# Schedule ⇒ When several transaction are Executing Concurrently then the order of Execution of various instruction is known as a <u>schedule</u>.
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.

$$\begin{tikzcd}
 & & \boxed{\text{Types of Schedules}} \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drr] & & \\
\boxed{\text{Serial Schedules}} & \boxed{\text{Non-Serial Schedules}} & & \boxed{\text{Conflict Schedules}} & \boxed{\text{View Schedules}}
\end{tikzcd}$$

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 |
|---|---|
| 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 Schedule if the operations of Each transaction are Executed non-consecutively with interleaved operations from the other transaction.

| T1 | T2 |
|---|---|
| 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

| $T_1$ | $T_2$ |
|---|---|
| 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 $\Longrightarrow$

| $T_1$ | $T_2$ |
|---|---|
| | 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 A, then in S2, transaction T1 should also read A

| T1 | T2 |
|---|---|
| read(A) | |
| | write(A) |

Schedule S1.

| T1 | T2 |
|---|---|
| | 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

by $T_j$.

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| Write(A) | | |
| | write(A) | |
| | | Read(A) |

Schedule S1

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| | write(A) | |
| write(A) | | |
| | | Read(A) |

Schedule S2

Above two Schedules are not view Equal because, in S1, T3 is reading A updated 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.