Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are –

- Insertion anomaly
- update anomaly
- deletion anomaly.

EmpId	Name	Job	Deptno	Dname	Dmgr
100	Amit	Sales	10	Sales	105
101	Ram	Manager	20	Account	106
102	Amar	Clerk	20	Account	106
103	Roshni	Clerk	30	Operations	108
104	Abhishek	Sales	10	Sales	105

Functional dependency in DBMS

The attributes of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.

For example: Suppose we have a student table with attributes: Stu_Id, Stu_Name, Stu_Age. Here Stu_Id attribute uniquely identifies the Stu_Name attribute of student table because if we know the student id we can tell the student name associated with it. This is known as functional dependency and can be written as Stu_Id->Stu_Name or in words we can say Stu_Name is functionally dependent on Stu_Id.

Formally:
If column A of a table uniquely identifies the column B of same table then it can represented as A->B (Attribute B is functionally dependent on attribute A)

A→ determinant

B→ Object of determinant

Types of Functional Dependencies

- **Compound Determinant**

  (internal marks , external marks → grades)

- **Full functional dependency**

  (internal marks , external marks → grades)

- **Trivial Functional Dependency**

  A ->B is trivial functional dependency if B is a subset of A.

  {Student_Id, Student_Name} -> Student_Id

- **non-trivial functional dependency**

  emp_id -> emp_name

- **Multivalued dependency**

  bike_model ->> color

- **Transitive dependency**

  $A \rightarrow B$

  $B \rightarrow C$

  $A \rightarrow C$

  Example

  Staff No →Designation

  Designation →Salary

  Staff No → Salary

- **Partial dependency**

Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.

<StudentProject>

StudentID	**ProjectNo**	**StudentName**	**ProjectName**
S01	199	Katie	Geo Location
S02	120	Ollie	Cluster Exploration

Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y.

If $X \supseteq Y$ 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 augmentation, if X determines Y, then XZ determines YZ for any Z.

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

Example:

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

Proof:

X $\rightarrow$ Y (given)
X $\rightarrow$ Z (given)
X $\rightarrow$ XY (using IR_2 on 1 by augmentation with X. Where XX = X)
XY $\rightarrow$ YZ (using IR_2 on 2 by augmentation with Y)
X $\rightarrow$ YZ (using IR_3 on 3 and 4)

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

Proof:

X $\rightarrow$ YZ (given)
YZ $\rightarrow$ Y (using IR_1 Rule)
X $\rightarrow$ Y (using IR_3 on 1 and 2)

6. Pseudo transitive Rule (IR_6)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If X $\rightarrow$ Y and YZ $\rightarrow$ W then XZ $\rightarrow$ W

Proof:

X $\rightarrow$ Y (given)
WY $\rightarrow$ Z (given)
WX $\rightarrow$ WY (using IR_2 on 1 by augmenting with W)
WX $\rightarrow$ Z (using IR_3 on 3 and 2)

Steps to Find the Attribute Closure of A

Q. Given FD set of a Relation R, The attribute closure set S be the set of A

- Add A to S.
- Recursively add attributes which can be functionally determined from attributes of the set S until done.

Q. Find the attribute closures of given FDs R(ABCDE) = {AB->C, B->D, C->E, D->A} To find $(B)^+$,we will add attribute in set using various FD which has been shown in table below.

Attributes Added in Closure	FD used
{B}	Triviality
{B,D}	B->D
{B,D,A}	D->A
{B,D,A,C}	AB->C
{B,D,A,C,E}	C->E

- We can find $(C, D)^+$ by adding C and D into the set (triviality) and then E using(C->E) and then A using (D->A) and set becomes.

  $(C,D)^+ = \{C,D,E,A\}$
- Similarly we can find $(B,C)^+$ by adding B and C into the set (triviality) and then D using (B->D) and then E using (C->E) and then A using (D->A) and set becomes

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

How to check whether an FD can be derived from a given FD set?

To check whether an FD A->B can be derived from an FD set F,

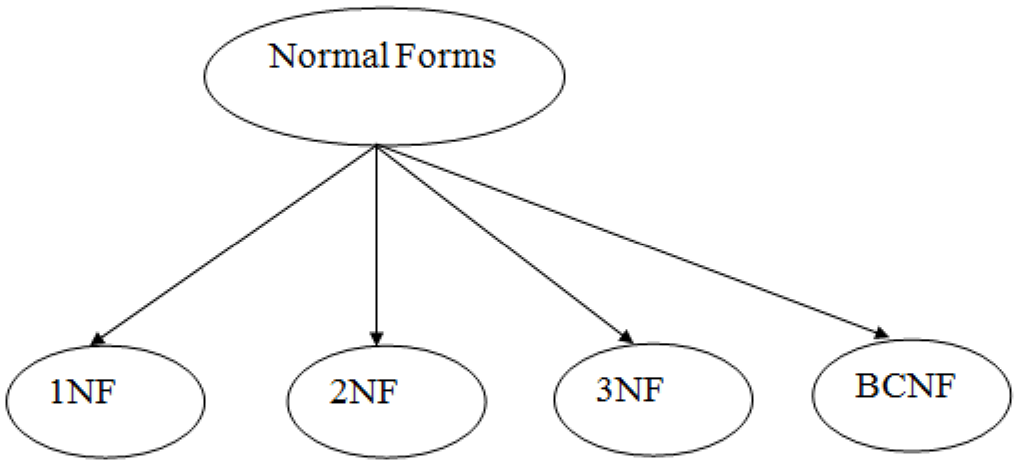
1. Find (A)+ using FD set F.
2. If B is subset of (A)+, then A->B is true else not true.

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 also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Types of Normal Forms

There are the four types of normal forms:

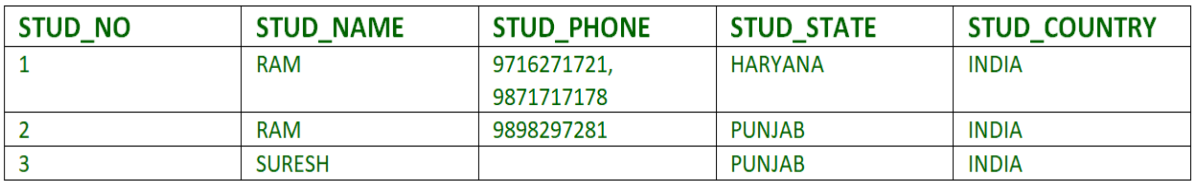


Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

1. First Normal Form –

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

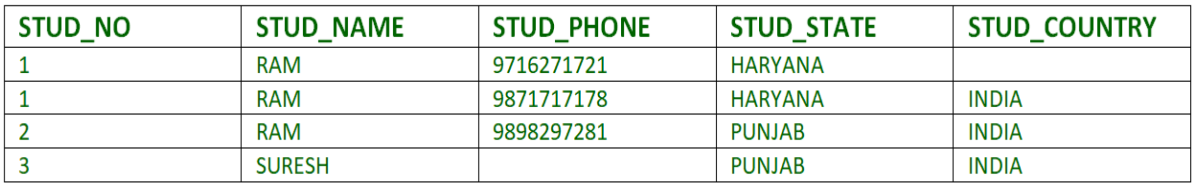
- **Example 1 –** Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.



STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

$\downarrow$ Conversion to first normal form



STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

- **Example 2 –**

```

ID   Name   Courses
------------------
1    A      c1, c2
2    E      c3
3    M      C2, c3
```

In the above table Course is a multi valued attribute so it is not in 1NF.

Below Table is in 1NF as there is no multi valued attribute

```
ID   Name   Course
------------------
1    A       c1
1    A       c2
2    E       c3
3    M       c2
```

```
3    M       c3
```

. Second Normal Form –

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has **No Partial Dependency,** i.e.**,** no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.
Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

Example :

E_ID	E_NAME	P_ID	P_NAME	TOTAL_TIME

{ E_ID, P_ID} → TOTAL_TIME

E_ID → E_NAME

P_ID → P_NAME

Prime Attributes are those attributes , which are part of candidate key, but not a candidate key.

3. Third Normal Form –

A relation is in third normal form, if there is **no transitive dependency** for non-prime attributes as well as it is in second normal form.
A relation is in 3NF if **at least one of the following condition holds** in every non-trivial function dependency X –> Y

1. X is a super key.
2. Y is a prime attribute (each element of Y is part of some candidate key).

SSN	ENAME	BDATE	ADD	DNUMBER	DNAME	DMGR

SSN → ENAME,BDATE,ADD,DNUMBER

DNUMBER → DNAME,DMANAGER

SSN	ENAME	BDATE	ADD	DNUMBER

DNUMBER	DNAME	DMGR

4. Boyce-Codd Normal Form (BCNF)

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key.

IT IS STRICTER FORM OF 3NF.

STUDENT	COURSE	INSTRUCTOR

STUDENT , COURSE → INSTRUCTOR

INSTRUCTOR → STUDENT

LOSSLESS/ LOSSY DECOMPOSITION

Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.

The following are the types −

Lossless Decomposition

Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. This is the preferred choice. The information will not lose from the relation when decomposed. The join would result in the same original relation.

Let us see an example −

<EmpInfo>

Emp_ID	**Emp_Name**	**Emp_Age**	**Emp_Location**	**Dept_ID**	**Dept_Name**
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Decompose the above table into two tables:

<EmpDetails>

Emp_ID	**Emp_Name**	**Emp_Age**	**Emp_Location**
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

<DeptDetails>

Dept_ID	**Emp_ID**	**Dept_Name**
Dpt1	E001	Operations
Dpt2	E002	HR
Dpt3	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	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Therefore, the above relation had lossless decomposition i.e. no loss of information.

Lossy Decomposition

As the name suggests, when a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.

Let us see an example –

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Decompose the above table into two tables –

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

<DeptDetails>

Dept_ID	Dept_Name

Dpt1	Operations
Dpt2	HR
Dpt3	Finance

Now, you won't be able to join the above tables, since **Emp_ID** isn't part of the **DeptDetails** relation. Therefore, the above relation has lossy decomposition.

Example-1:
Example to check whether given Decomposition Lossless Join Decomposition.
Let there be a relational schema R(A, B, C). R1(A, B) and R2(B, C) be it's decompositions.

A	**B**	**C**
a1	b1	c1
a2	b1	c1
a1	b2	c2

R

A	**B**
a1	b1
a2	b1
a1	b2

A	B

R1

B	C
b1	c1
b1	c1
b2	c2

R2

Now for the decomposition to be lossless,

R1 ⋈ R2 = R then, R1 ⋈ R2 is

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2

R1 ⋈ R2

As, R1 ⋈ R2 = R,

This decomposition is Lossless.

Example-2:
Example to check whether given Decomposition Lossy Join Decomposition.
Let there be a relational schema R(A, B, C). R1(A, B) and R2(A, C) be it’s decompositions.

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2
a1	b3	c3

R

A	B
a1	b1
a2	b1
a1	b2
a1	b3

R1

A	C
a1	c1
a2	c1
a1	c2
a1	c3
R2	

Now for the decomposition to be lossy,

R ⊂ R1 ⋈ R2 then, R1 ⋈ R2 is

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a1	b2	c2
a1	b2	c1
a1	b3	c3
a1	b3	c1

A	B	C

R1 ⋈ R2

As, R ⊂ R1 ⋈ R2,

This decomposition is Lossy.

Multivalued Dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

1. BIKE_MODEL → → MANUF_YEAR
2. BIKE_MODEL → → COLOR

This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

A table with multivalued dependency violates the 4NF.

Example

Let us see an example &mins;

<Student>

StudentName	**CourseDiscipline**	**Activities**
Amit	Mathematics	Singing
Amit	Mathematics	Dancing
Yuvraj	Computers	Cricket
Akash	Literature	Dancing
Akash	Literature	Cricket
Akash	Literature	Singing

In the above table, we can see Students **Amit** and **Akash** have interest in more than one activity.

This is multivalued dependency because **CourseDiscipline** of a student are independent of Activities, but are dependent on the student.

Therefore, multivalued dependency −

```
StudentName ->-> CourseDiscipline
StudentName ->-> Activities
```

The above relation violates Fourth Normal Form in Normalization.

To correct it, divide the table into two separate tables and break Multivalued Dependency −

<StudentCourse>

StudentName	**CourseDiscipline**
Amit	Mathematics

Amit	Mathematics
Yuvraj	Computers
Akash	Literature
Akash	Literature
Akash	Literature

<StudentActivities>

StudentName	**Activities**
Amit	Singing
Amit	Dancing
Yuvraj	Cricket
Akash	Dancing
Akash	Cricket
Akash	Singing

This breaks the multivalued dependency and now we have two functional dependencies −

StudentName -> CourseDiscipline
StudentName - > Activities

Rules for 4th Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1. It should be in the **Boyce-Codd Normal Form**.
2. And, the table should not have any **Multi-valued Dependency**.

Join Dependency

- Join decomposition is a further generalization of Multivalued dependencies.
- If the join of R1 and R2 over C is equal to relation R, then we can say that a join dependency (JD) exists.
- Where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D).
- Alternatively, R1 and R2 are a lossless decomposition of R.
- A JD ⋈ {R1, R2,..., Rn} is said to hold over a relation R if R1, R2,....., Rn is a lossless-join decomposition.
- The *(A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R.
- Here, *(R1, R2, R3) is used to indicate that relation R1, R2, R3 and so on are a JD of R.

What is Join Dependency?

If a table can be recreated by joining multiple tables and each of this table have a subset of the attributes of the table, then the table is in Join Dependency. It is a generalization of Multivalued Dependency

Join Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it cannot be decomposed further.

Example

<Employee>

EmpName	**EmpSkills**	**EmpJob** (Assigned Work)
Tom	Networking	EJ001

Harry	Web Development	EJ002
Katie	Programming	EJ002

The above table can be decomposed into the following three tables; therefore it is not in 5NF:

<EmployeeSkills>

EmpName	**EmpSkills**
Tom	Networking
Harry	Web Development
Katie	Programming

<EmployeeJob>

EmpName	**EmpJob**
Tom	EJ001
Harry	EJ002
Katie	EJ002

<JobSkills>

EmpSkills	**EmpJob**
Networking	EJ001

Web Development	EJ002
Programming	EJ002

Our Join Dependency −

{(EmpName, EmpSkills), (EmpName, EmpJob), (EmpSkills, EmpJob)}

The above relations have join dependency, so they are not in 5NF. That would mean that a join relation of the above three relations is equal to our original relation **<Employee>**.

Fifth Normal Form / Projected Normal Form (5NF):

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated, when relations are reunited through a natural join.

Properties – A relation R is in 5NF if and only if it satisfies following conditions:

1. R should be already in 4NF.
2. It cannot be further non loss decomposed (join dependency)

Example – Consider the above schema, with a case as "if a company makes a product and an agent is an agent for that company, then he always sells that product for the company". Under these circumstances, the ACP table is shown as:

Table – ACP

AGENT	COMPANY	PRODUCT
A1	PQR	Nut
A1	PQR	Bolt

AGENT	COMPANY	PRODUCT
A1	XYZ	Nut
A1	XYZ	Bolt
A2	PQR	Nut

The relation ACP is again decompose into 3 relations. Now, the natural Join of all the three relations will be shown as:

Table – R1

AGENT	COMPANY
A1	PQR
A1	XYZ
A2	PQR

Table – R2

AGENT	PRODUCT
A1	Nut

AGENT	PRODUCT
A1	Bolt
A2	Nut

Table – R3

COMPANY	PRODUCT
PQR	Nut
PQR	Bolt
XYZ	Nut
XYZ	Bolt

Result of Natural Join of R1 and R3 over 'Company' and then Natural Join of R13 and R2 over 'Agent'and 'Product' will be table **ACP**.
Hence, in this example, all the redundancies are eliminated, and the decomposition of ACP is a lossless join decomposition. Therefore, the relation is in 5NF as it does not violate the property of lossless join.

Inclusion Dependency

- Multivalued dependency and join dependency can be used to guide database design although they both are less common than functional dependencies.
- Inclusion dependencies are quite common. They typically show little influence on designing of the database.
- The inclusion dependency is a statement in which some columns of a relation are contained in other columns.
- The example of inclusion dependency is a foreign key. In one relation, the referring relation is contained in the primary key column(s) of the referenced relation.
- Suppose we have two relations R and S which was obtained by translating two entity sets such that every R entity is also an S entity.
- Inclusion dependency would be happen if projecting R on its key attributes yields a relation that is contained in the relation obtained by projecting S on its key attributes.
- In inclusion dependency, we should not split groups of attributes that participate in an inclusion dependency.
  - In practice, most inclusion dependencies are key-based that is involved only keys