$f^* \rightarrow \{ A \rightarrow A, A \rightarrow B, B \rightarrow C, A \rightarrow C, C \rightarrow D, A \rightarrow D,$

$B \rightarrow D \ldots\ldots \}$

→ **Attribute closure :—**

It defines all the attributes that can be determine using an attribute.

For escample :—

$A^* \longrightarrow B\ C\ D$

$B^* \longrightarrow C\ D$

* Normalization :—

It is the process of determining how much redundancy exist in a table and it gives us techniques to reduce it. It will help us to characterise the level of redundency that how much redundancy there and it will provide mechanisms to remove all those redundencies.

There are multiple normal forms - normal forms actually helps us to understand what level of redundancy you have and gives us technique, every normal form is a technique to actually reduce the redundancy at some position.

There are multiple normal forms, such as :—

- → 1NF
- → 2NF
- → 3NF
- → BCNF

- Every normal form is dependent on other. For example 2NF is dependent on 1NF, 3NF is dependent on 2NF, BCNF is dependent on 3NF and so on.

1. First Normal form (1NF) :—

It is the most simplest form. It says that - Any attribute must only contain atomic values (indivisible)

For example :—

s_id	s_name	s_course
1	Sachin	DBMS, OS, DSA, ED
2	Raju	DSA, SE

→ Student Table

This is very bad design because there exists redundancy. We can see that "DSA" course is placed at multiple places.

- If you want to rename the course "DSA" to "Data structure & Algo", so the problem is you have to rename it at multiple places.

- There exist even bigger problem that most probably we will store data like an array. So you have to go all student and read through the whole array. Is this "DSA" yes ya No, many time, If yes then update otherwise No. Means, we have to go every single subject of every student in order to update it, that will take hell of time. So, this is a bad design.

So, first normal formal says that you should only have atomic values.

So, now we will distribute the table again.

s_id	s_name	s_course
1	Sachin	DBMS
1	Sachin	OS
1	Sachin	DSA
2	Raju	DSA
⋮	⋮	⋮

→ It is not the best design because still there is redundency but it is a better design because now we don't have to go every single subject because of array.

- We can now easily filter the data. We can say that update only those rows where the course name is "DSA".

- At any point of time, you have a column which have multivalued attribute, 1NF discard them.

2. Second Normal Form (2NF) :—

Second normal form says that the table should not have partial dependencies, and the table should be already 1NF.

For example :—

Table_Purchase_Detail

customer_id	stored_id	Location
1	1	Los Angeles
1	3	San Franciso
2	1	Los Angeles
3	2	New York
4	3	San Franciso

This table has a composite primary key [customer ID, store ID]. The non key attribute is [Purchas Location]. In this case, [Location] only depends on [store ID], which is only part of the primary key. Therefore, this table does not satisfy second normal form.

To bring this table to second form, we break the table into two tables.

Table_Purchase

customer_id	store_id
1	1
1	3
2	1
3	2
4	3

Table_Store

store_id	Location
1	Los Angeles
2	New York
3	San Franciso

Now we have remove the partial dependency that we initially had. Now in the table [Table_Store], the column [Location] is fully dependent on the primary key of that table, which is [Store_id].

3. Third Normal Form (3NF) :-

- The table should be in 2NF
- It should not have transitive dependency

STUDENT TABLE

S_no	S_name	s_state	s_country	s_age
1	A	Haryana	INDIA	20
2	A	Punjab	INDIA	19
3	B	Punjab	INDIA	21

For this table, s-no → s-state and s-state → s-country are true. s- country is transitively dependent on s-no. It violates the third normal form.

To convert it in third normal form, we will decompose the relation :-

student (s-no, s-name, s-phone, s-state, s-country, s-age) as:

student (s-no, s-name, s-phone, s-state, s-age)

s-country (state, country)