



# SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)  
Approved by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai



## Department of MCA

### DBMS Introduction

**Course Name : 19CAT609 - DATA BASE MANAGEMENT SYSTEM**

**Class : I Year / II Semester**

**Unit IV - Normalization**





# Normalization



1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (BCNF)
5. Fourth Normal Form (4NF)
6. Fifth Normal Form (5NF)



# Normalization



If a table has data redundancy and is not properly normalized, then it will be difficult to handle and update the database, without facing data loss. It will also eat up extra memory space and Insertion, Update and Deletion *Anomalies are very frequent if database is not normalized.*

[Normalization](#) is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.



# First Normal Form (1NF)



## First Normal Form (1NF):

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the 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.

A table is in 1 NF if:

- There are only Single Valued Attributes.
- Attribute Domain does not change.
- There is a unique name for every Attribute/Column.
- The order in which data is stored does not matter.



# First Normal Form (1NF)



Consider the examples given below.

## 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

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



# First Normal Form (1NF)



1 NF

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.



# First Normal Form (1NF)



## First Normal Form (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

**Note:** A database design is considered as bad if it is not even in the First Normal Form (1NF).



# Second Normal Form (2NF)



**Second Normal Form (2NF):** Second Normal Form (2NF) is based on the concept of full functional dependency. Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes. A relation with a single-attribute primary key is automatically in at least 2NF. A relation that is not in 2NF may suffer from the update anomalies. 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. In other words,





# Second Normal Form (2NF)



A relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key, then the relation is in Second Normal Form (2NF).

**Note** – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency. The normalization of 1NF relations to 2NF involves the **removal of partial dependencies**. If a partial dependency exists, we remove the partially dependent attribute(s) from the relation by placing them in a new relation along with a copy of their determinant. Consider the examples given below.



# Second Normal Form (2NF)



Consider table as following below.

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000



## Second Normal Form (2NF)



{Note that, there are many courses having the same course fee. } Here, COURSE\_FEE cannot alone decide the value of COURSE\_NO or STUD\_NO; COURSE\_FEE together with STUD\_NO cannot decide the value of COURSE\_NO; COURSE\_FEE together with COURSE\_NO cannot decide the value of STUD\_NO; Hence, COURSE\_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD\_NO, COURSE\_NO} ; But, COURSE\_NO  $\rightarrow$  COURSE\_FEE, i.e., COURSE\_FEE is dependent on COURSE\_NO, which is a proper subset of the candidate key. Non-prime attribute COURSE\_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF.



# Second Normal Form (2NF)



To convert the above relation to 2NF, we need to split the table into two tables such as :

Table 1: STUD\_NO, COURSE\_NO Table 2: COURSE\_NO, COURSE\_FEE

Table 1		Table 2	
STUD_NO	COURSE_NO	COURSE_NO	COURSE_FEE
1	C1	C1	1000
2	C2	C2	1500
1	C4	C3	1000
4	C3	C4	2000
4	C1	C5	2000

**Note** – 2NF tries to reduce the redundant data getting stored in memory. For instance, if there are 100 students taking C1 course, we dont need to store its Fee as 1000 for all the 100 records, instead once we can store it in the second table as the course fee for C1 is 1000.



# Second Normal Form (2NF)



**Example-2:** Consider following functional dependencies in relation

$R(A, B, C, D)$

$AB \rightarrow C$  [A and B together determine C]

$BC \rightarrow D$  [B and C together determine D]

In this case, we can see that the relation R has a composite candidate key  $\{A, B\}$  as  $AB \rightarrow C$ .

Therefore, A and B together uniquely determine the value of C.

Similarly,  $BC \rightarrow D$  shows that B and C together uniquely determine the value of D.

The relation R is already in 1NF because it does not have any repeating groups or nested relations.

However, we can see that the non-prime attribute D is functionally dependent on only part of a candidate key, BC. This violates the 2NF condition.



# Third Normal Form (3NF)



## Third Normal Form (3NF):

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  $X \rightarrow Y$ :

X is a super key.

Y is a prime attribute (each element of Y is part of some candidate key).

A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

**Note** – If  $A \rightarrow B$  and  $B \rightarrow C$  are two FDs then  $A \rightarrow C$  is called transitive dependency.



# Third Normal Form (3NF)



The [normalization](#) of 2NF relations to 3NF involves the removal of transitive dependencies. If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.

Consider the examples given below.

## Example-1:

In relation STUDENT given in Table 4,

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

**Table 4**



# Third Normal Form (3NF)



FD set:

{STUD\_NO -> STUD\_NAME, STUD\_NO -> STUD\_STATE, STUD\_STATE -> STUD\_COUNTRY,  
STUD\_NO -> STUD\_AGE}

Candidate Key:

{STUD\_NO}

For this relation in table 4, STUD\_NO -> STUD\_STATE and STUD\_STATE -> STUD\_COUNTRY are true. So STUD\_COUNTRY is transitively dependent on STUD\_NO. It violates the third normal form. To convert it in third normal form, we will decompose the relation STUDENT (STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_COUNTRY, STUD\_AGE) as:

STUDENT (STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_AGE)  
STATE\_COUNTRY (STATE, COUNTRY)





# Third Normal Form (3NF)



## Example-2:

Consider relation R(A, B, C, D, E)

A  $\rightarrow$  BC,

CD  $\rightarrow$  E,

B  $\rightarrow$  D,

E  $\rightarrow$  A

All possible candidate keys in above relation are {A, E, CD, BC} All attribute are on right sides of all functional dependencies are prime.

## Note –

Third Normal Form (3NF) is considered *adequate* for normal relational database design because most of the 3NF tables are free of insertion, update, and deletion anomalies. Moreover, 3NF *always ensures functional dependency preserving and lossless*.



# Boyce-Codd Normal Form (BCNF)



## Boyce-Codd Normal Form (BCNF)

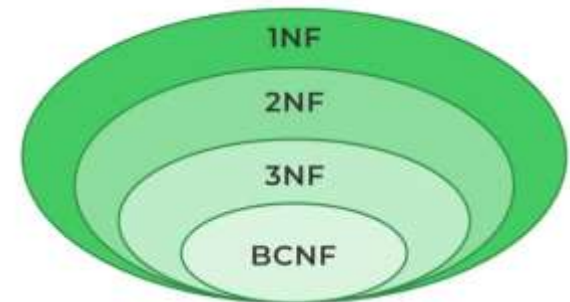
Boyce–Codd Normal Form (BCNF) is based on [functional dependencies](#) that take into account all candidate keys in a relation; however, BCNF also has additional constraints compared with the general definition of 3NF.

### Rules for BCNF

**Rule 1:** The table should be in the 3rd Normal Form.

**Rule 2:** X should be a superkey for every functional dependency (FD)  $X \rightarrow Y$  in a given relation.

**Note:** To test whether a relation is in BCNF, we identify all the determinants and make sure that they are candidate keys.





# Boyce-Codd Normal Form (BCNF)



You came across a similar hierarchy known as the **Chomsky Normal Form** in the Theory of Computation. Now, carefully study the hierarchy above. It can be inferred that **every relation in BCNF is also in 3NF**. To put it another way, a relation in 3NF need not be in BCNF.

To determine the highest normal form of a given relation R with functional dependencies, the first step is to check whether the BCNF condition holds. If R is found to be in BCNF, it can be safely deduced that the relation is also in [3NF](#), [2NF](#), and [1NF](#) as the hierarchy shows. The 1NF has the least restrictive constraint – it only requires a relation R to have atomic values in each tuple. The 2NF has a slightly more restrictive constraint.

The 3NF has a more restrictive constraint than the first two normal forms but is less restrictive than the BCNF. In this manner, the restriction increases as we traverse down the hierarchy.



# Boyce-Codd Normal Form (BCNF)



## Examples

Here, we are going to discuss some basic examples which let you understand the properties of BCNF. We will discuss multiple examples here.

### Example 1

Let us consider the student database, in which data of the student are mentioned.

Stu_ID	Stu_Branch	Stu_Course	Branch_Number	Stu_Course_No
101	Computer Science & Engineering	DBMS	B_001	201
101	Computer Science & Engineering	Computer Networks	B_001	202
102	Electronics & Communication Engineering	VLSI Technology	B_003	401
102	Electronics & Communication Engineering	Mobile Communication	B_003	402



# Boyce-Codd Normal Form (BCNF)



Functional Dependency of the above is as mentioned:

$\text{Stu\_ID} \rightarrow \text{Stu\_Branch}$   $\text{Stu\_Course} \rightarrow \{\text{Branch\_Number}, \text{Stu\_Course\_No}\}$  Candidate Keys of the above table are: **{Stu\_ID, Stu\_Course}**

## Why this Table is Not in BCNF?

The table present above is not in BCNF, because as we can see that neither Stu\_ID nor Stu\_Course is a Super Key. As the rules mentioned above clearly tell that for a table to be in BCNF, it must follow the property that for functional dependency  $X \rightarrow Y$ , X must be in Super Key and here this property fails, that's why this table is not in BCNF.



# Boyce-Codd Normal Form (BCNF)



## How to Satisfy BCNF?

For satisfying this table in BCNF, we have to decompose it into further tables. Here is the full procedure through which we transform this table into BCNF. Let us first divide this main table into two tables **Stu\_Branch** and **Stu\_Course** Table.

### Stu\_Branch Table

Stu_ID	Stu_Branch
101	Computer Science & Engineering
102	Electronics & Communication Engineering

Candidate Key for this table: **Stu\_ID**.



# Boyce-Codd Normal Form (BCNF)



## Stu\_Course Table

Stu_Course	Branch_Number	Stu_Course_No
DBMS	B_001	201
Computer Networks	B_001	202
VLSI Technology	B_003	401
Mobile Communication	B_003	402

Candidate Key for this table: **Stu\_Course**.

## Stu\_ID to Stu\_Course\_No Table

Stu_ID	Stu_Course_No
101	201
101	202
102	401
102	402

Candidate Key for this table:  
{**Stu\_ID, Stu\_Course\_No**}.

After decomposing into further tables, now it is in BCNF, as it is passing the condition of Super Key, that in functional dependency  $X \rightarrow Y$ , X is a **Super Key**.



# Boyce-Codd Normal Form (BCNF)



## Example 2

Find the highest normal form of a relation  $R(A, B, C, D, E)$  with FD set as:  
{  $BC \rightarrow D$ ,  $AC \rightarrow BE$ ,  $B \rightarrow E$  }

### Explanation:

**Step-1:** As we can see,  $(AC)^+ = \{A, C, B, E, D\}$  but none of its subsets can determine all attributes of the relation, So AC will be the candidate key. A or C can't be derived from any other attribute of the relation, so there will be only 1 candidate key {AC}.

**Step-2:** Prime attributes are those attributes that are part of candidate key {A, C} in this example and others will be non-prime {B, D, E} in this example.

**Step-3:** The relation R is in 1st normal form as a relational DBMS does not allow multi-valued or composite attributes.





# Boyce-Codd Normal Form (BCNF)



The relation is in 2nd normal form because  $BC \rightarrow D$  is in 2nd normal form ( $BC$  is not a proper subset of candidate key  $AC$ ) and

$AC \rightarrow BE$  is in 2nd normal form ( $AC$  is candidate key) and  $B \rightarrow E$  is in 2nd normal form ( $B$  is not a proper subset of candidate key  $AC$ ).

The relation is **not** in 3rd normal form because in  $BC \rightarrow D$  (neither  $BC$  is a super key nor  $D$  is a prime attribute) and in  $B \rightarrow E$  (neither  $B$  is a super key nor  $E$  is a prime attribute) but to satisfy 3rd normal form, either LHS of an FD should be super key or RHS should be a prime attribute. So the highest normal form of relation will be the 2nd Normal form.

**Note:** A prime attribute cannot be transitively dependent on a key in BCNF relation.



# Boyce-Codd Normal Form (BCNF)



**Consider these functional dependencies of some relation R**

$AB \rightarrow C$

$C \rightarrow B$

$AB \rightarrow B$

Suppose, it is known that the only candidate key of R is AB. A careful observation is required to conclude that the above dependency is a Transitive Dependency as the prime attribute B transitively depends on the key AB through C. Now, the first and the third FD are in BCNF as they both contain the candidate key (or simply KEY) on their left sides. The second dependency, however, is not in BCNF but is definitely in 3NF due to the presence of the prime attribute on the right side. So, the highest normal form of R is 3NF as all three FDs satisfy the necessary conditions to be in 3NF.



# Boyce-Codd Normal Form (BCNF)



## Example 3

For example consider relation  $R(A, B, C)$

$A \rightarrow BC$ ,

$B \rightarrow A$

A and B both are super keys so the above relation is in BCNF.

**Note:** BCNF decomposition may always not be possible with [dependency preserving](#), however, it always satisfies the [lossless join](#) condition.

For example, relation  $R(V, W, X, Y, Z)$ ,  
with functional dependencies:

$V, W \rightarrow X$

$Y, Z \rightarrow X$

$W \rightarrow Y$



# Boyce-Codd Normal Form (BCNF)



It would not satisfy dependency preserving BCNF decomposition.

**Note:** Redundancies are sometimes still present in a BCNF relation as it is not always possible to eliminate them completely.

There are also some higher-order normal forms, like the 4th Normal Form and the 5th Normal Form.



## 4NF

- It should satisfy BCNF
- It should not have multivalued dependency

In **2NF** we removed **Partial Dependency**.

and, In **3NF** we removed **Transitive Dependency**.

- Multivalued Dependency



## 4NF

Any dependency:

$A \rightarrow B$ , is Multi-Valued Dependency





# 4NF



## 4NF

Table

Table

✗

✓

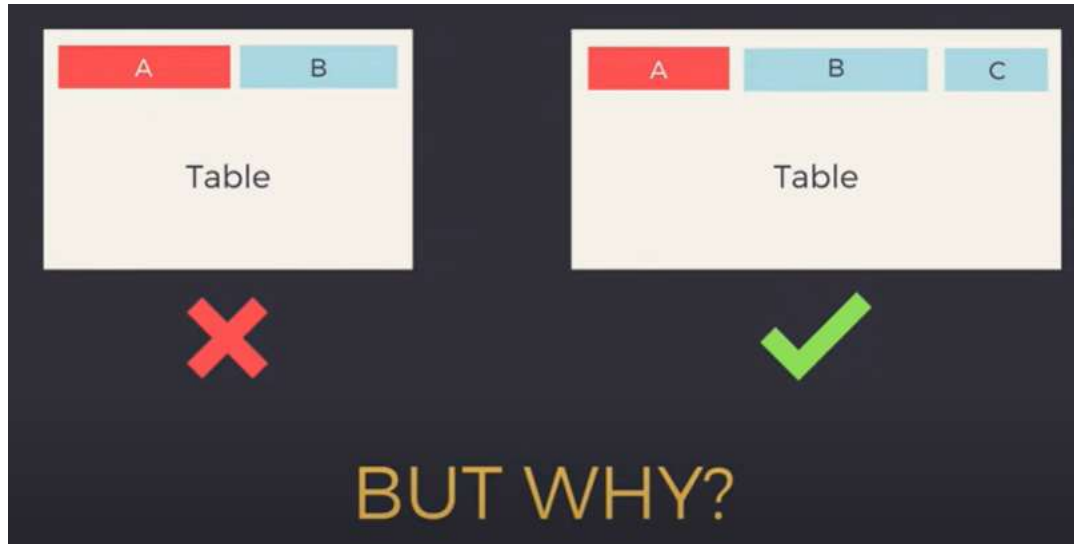
A table should have at least 3 columns to have Multi-valued Dependency



# 4NF



## 4NF



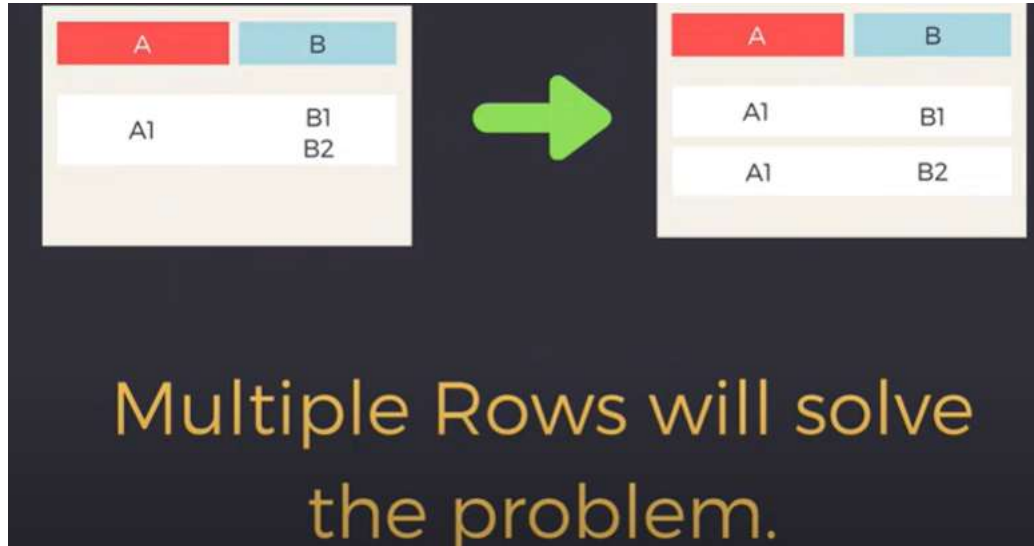




# 4NF



## 4NF





## 4NF

For a table with A, B, C columns

$A \twoheadrightarrow B$ , is Multi-Valued Dependency

Then B and C should be  
independent of each other.



4NF

## Three conditions for Multi-Valued Dependency

- $A \twoheadrightarrow B$ , for a single value of  $A$ , more than one value of  $B$  exist.
- Table should have at-least 3 columns.
- For this table with  $A, B, C$  columns,  $B$  and  $C$  should be independent.



# 4NF



4NF

If all these are true, then we can say that the table **may** have **Multi-valued Dependency**.

A	B	C
A1	B1	C1
	B2	C2

Multi-valued dependency  
between  $A \twoheadrightarrow B$  and  $A \twoheadrightarrow C$



# 4NF



## 4NF

ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey

ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey



# 4NF



## 4NF

ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket



# 4NF



## 4NF

ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

No relationship



# 4NF



## 4NF

ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

**BAD DESIGN**





## 4NF

### Student Enrollment Table



**CourseOpted Table + Hobbies Table**

(s\_id & course)

(s\_id & hobby)



# 4NF



## 4NF

CourseOpted TABLE		Hobbies TABLE	
s_id	course	s_id	hobby
1	Science	1	Cricket
1	Maths	1	Hockey
2	C#	2	Cricket
2	Php	2	Hockey

**2 independent tables**



# 4NF



## 4NF

A table can have both **Functional Dependency** and **Multi-Valued Dependency**.

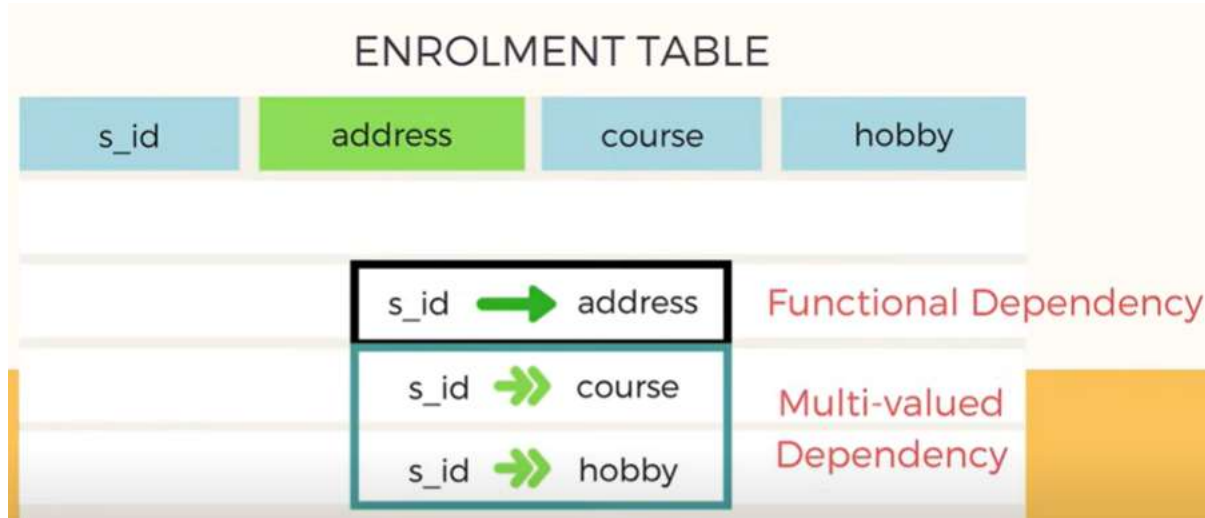
Let's add a new column **Address** to our original **Enrolment** Table.



# 4NF



## 4NF





## 4NF

### Student Enrollment Table



**CourseOpted Table + Hobbies Table + Address Table**

(s\_id & course)

(s\_id & hobby)

(s\_id & address)

Multi-valued Dependency occurs  
due to **bad** database design.




# 4NF



## 4NF

Student Table		Course Table	
s_id	s_name	c_id	c_name
S1	A	C1	C
S2	B	C2	D

 Good Database Design



# 4NF



## 4NF

ENROLMENT TABLE

s_id	s_name	c_id	c_name
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

**Now Multi-valued Dependency exist**

So design your Database carefully.



# Fourth Normal Form (4NF)



## Fourth Normal Form (4NF)

The [Fourth Normal Form \(4NF\)](#) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF, and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

## Properties

A relation R is in 4NF if and only if the following conditions are satisfied:

1. It should be in the Boyce-Codd Normal Form (BCNF).
2. The table should not have any Multi-valued Dependency.





# Fourth Normal Form (4NF)



A table with a multivalued dependency violates the normalization standard of the Fourth Normal Form (4NF) because it creates unnecessary redundancies and can contribute to inconsistent data. To bring this up to 4NF, it is necessary to break this information into two tables.

**Example:** Consider the database table of a class that has two relations R1 contains student ID(SID) and student name (SNAME) and R2 contains course id(CID) and course name (CNAME).

When their cross-product is done it resulted in multivalued dependencies.

Table R1

SID	SNAME
S1	A
S2	B

Table R2

CID	CNAME
C1	C
C2	D

Multivalued dependencies (MVD) are:

SID  $\twoheadrightarrow$  CID; SID  $\twoheadrightarrow$  CNAME;  
SNAME  $\twoheadrightarrow$  CNAME

Table R1 X R2

SID	SNAME	CID	CNAME
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D



# Fourth Normal Form (4NF)



## Joint Dependency

Join decomposition is a further generalization of Multivalued dependencies. If the join of  $R_1$  and  $R_2$  over  $C$  is equal to relation  $R$  then we can say that a join dependency (JD) exists, where  $R_1$  and  $R_2$  are the decomposition  $R_1(A, B, C)$  and  $R_2(C, D)$  of a given relations  $R(A, B, C, D)$ . Alternatively,  $R_1$  and  $R_2$  are a lossless decomposition of  $R$ . A JD  $\bowtie \{R_1, R_2, \dots, R_n\}$  is said to hold over a relation  $R$  if  $R_1, R_2, \dots, R_n$  is a lossless-join decomposition. The  $\bowtie(A, B, C, D), (C, D)$  will be a JD of  $R$  if the join of joins attribute is equal to the relation  $R$ . Here,  $\bowtie(R_1, R_2, R_3)$  is used to indicate that relation  $R_1, R_2, R_3$  and so on are a JD of  $R$ . Let  $R$  is a relation schema  $R_1, R_2, R_3, \dots, R_n$  be the decomposition of  $R$ .  $r(R)$  is said to satisfy join dependency if and only if

$$\bowtie_{i=1}^n \Pi_{R_i}(r) = r,$$



# Fourth Normal Form (4NF)



## Joint Dependency

Example:

Table R1

Company	Product
C1	Pendrive
C1	mic
C2	speaker
C2	speaker

Company ->>->Product

Table R2

Agent	Company
Aman	C1
Aman	C2
Mohan	C1

Agent ->>->Company

Table R3

Agent	Product
Aman	Pendrive
Aman	Mic
Aman	speaker
Mohan	speaker

Agent ->>->Product

Table R1⋈R2⋈R3

Company	Product	Agent
C1	Pendrive	Aman
C1	mic	Aman
C2	speaker	speaker
C1	speaker	Aman

Agent ->>->Product



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



5NF also known as PJNF

## 5th Normal Form

- Should be in 4NF
- It should not have  
Join Dependency.

• PJNF



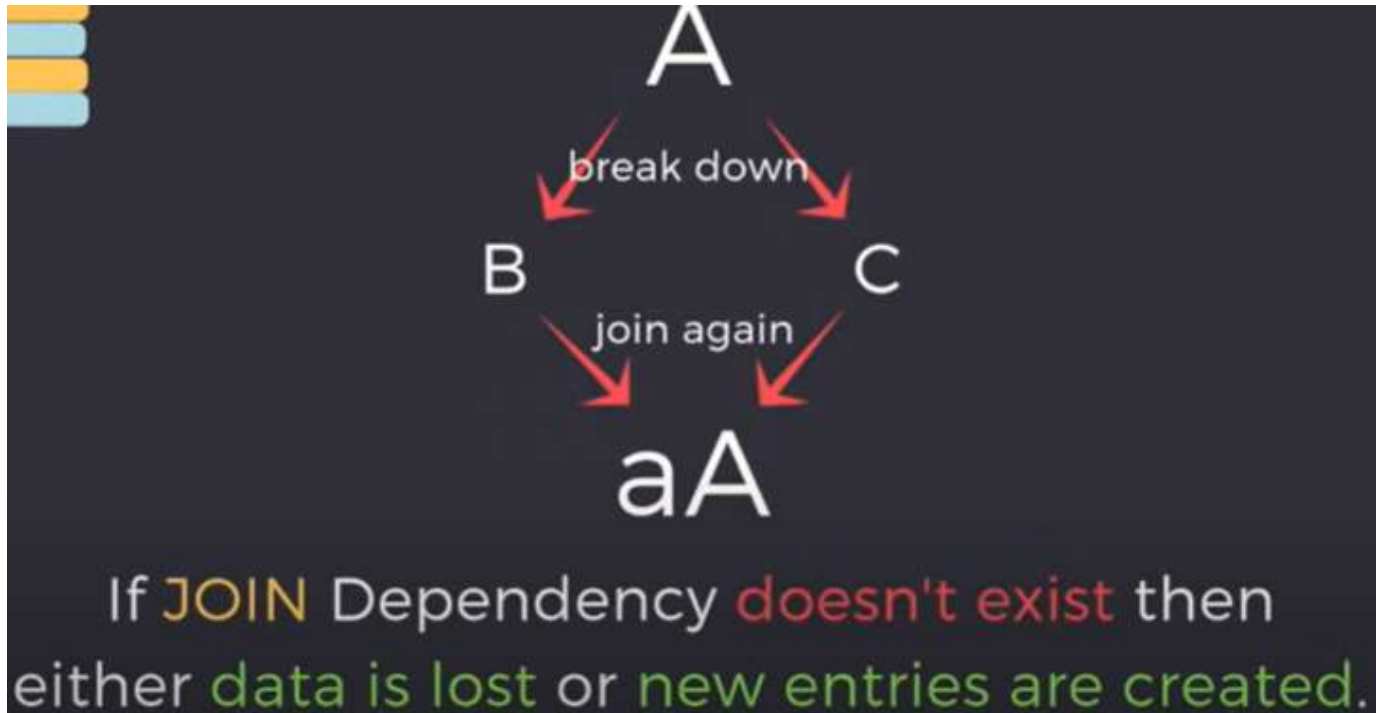
Project Join  
Normal Form



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



5NF

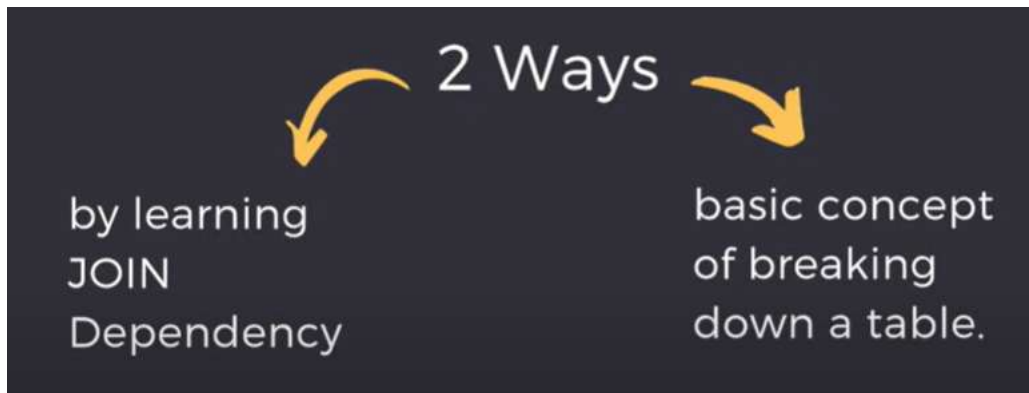




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

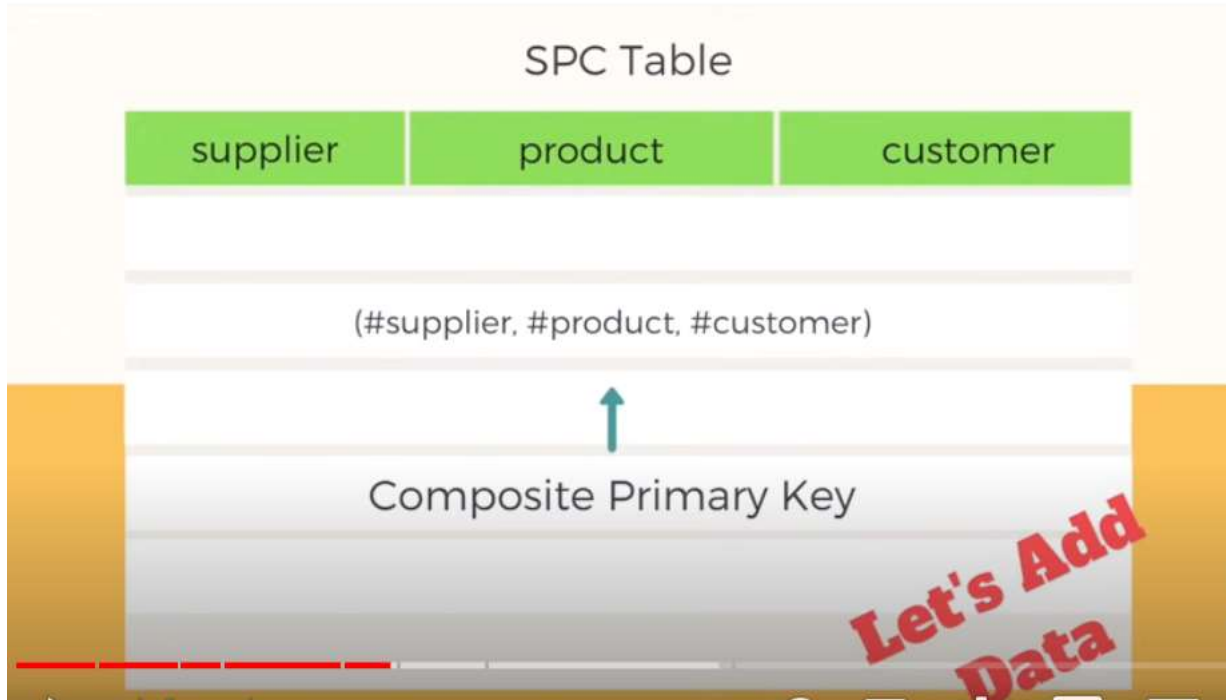




# Fifth Normal Form / Projected Normal Form (5NF)



5NF





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



5NF

SPC Table

supplier	product	customer
ACME	72X SW	FORD
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	GM
ALWAT	OBD II	MERCEDES





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



5NF

- Table should satisfy the 4th Normal Form
- If JOIN Dependency exist, decompose the table.

SPC Table

supplier	product	customer
ACME	72X SW	FORD
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	GM
ALWAT	OBD II	MERCEDES



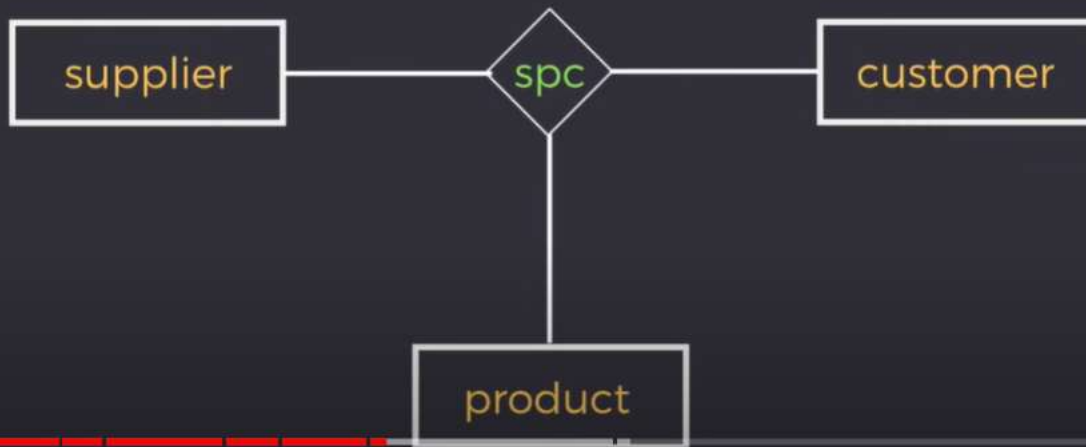
# Fifth Normal Form / Projected Normal Form (5NF)



5NF

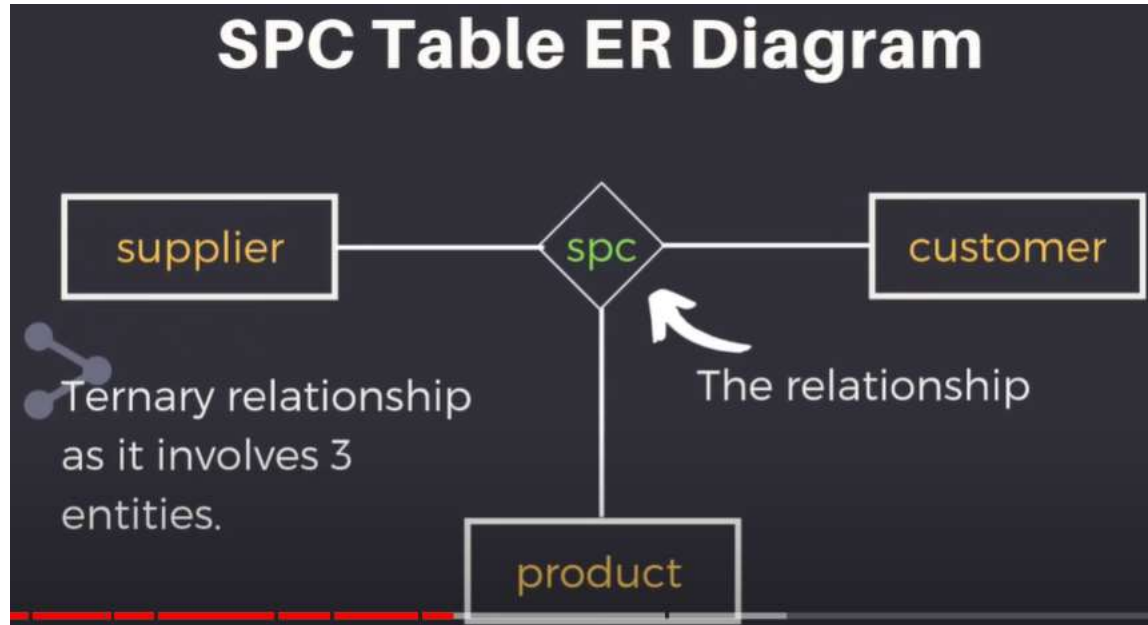
Let's create an ER Diagram...

## SPC Table ER Diagram





5NF

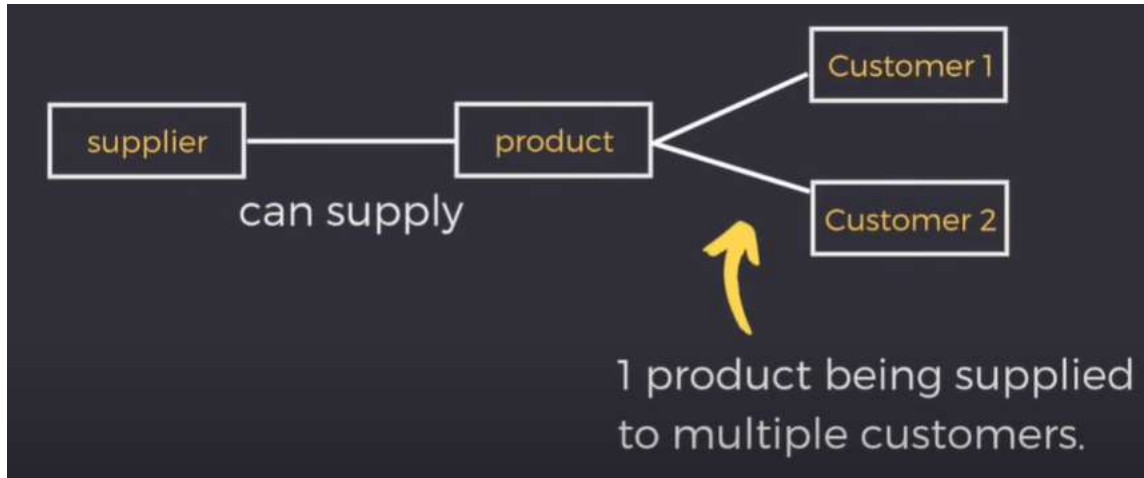




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

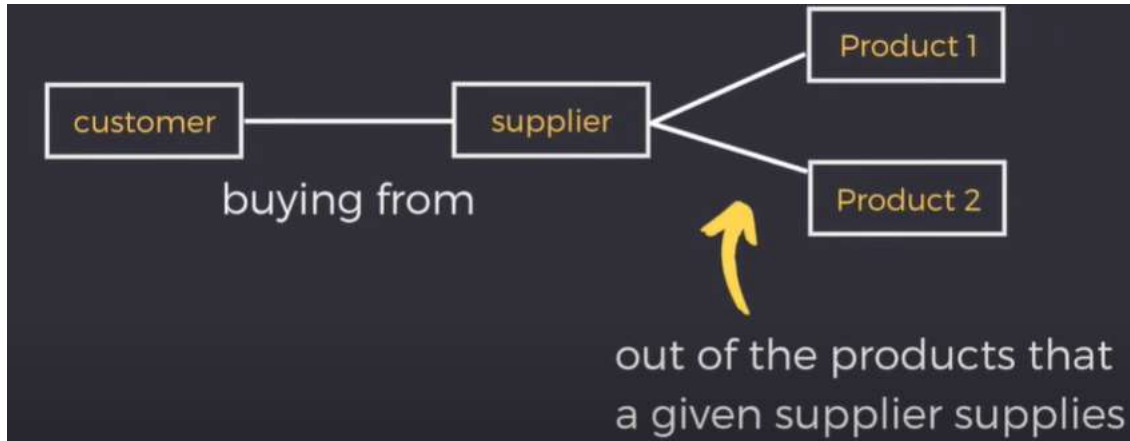




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

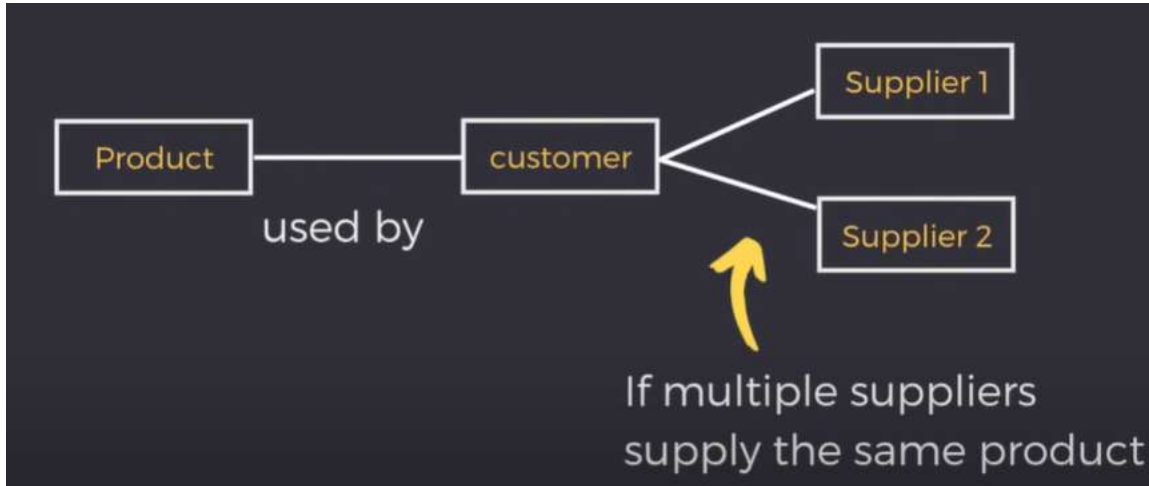




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

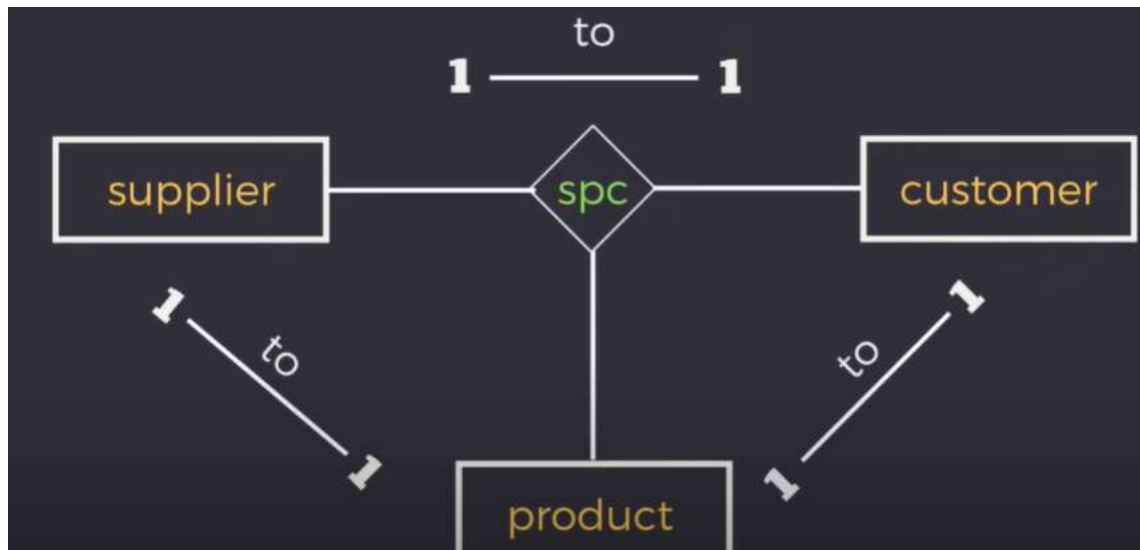




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

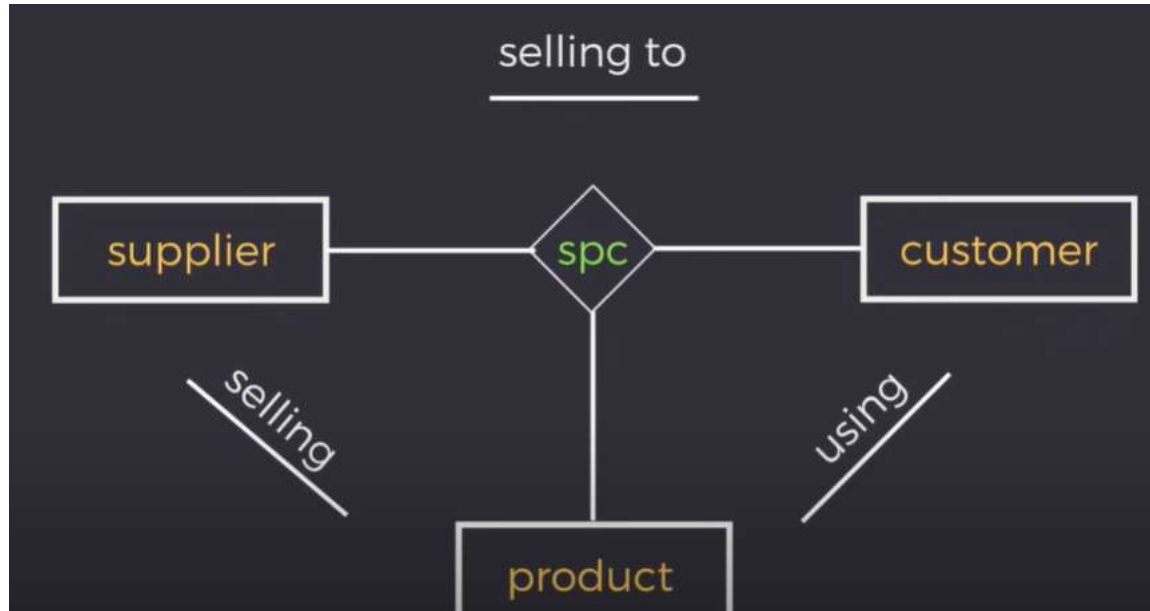




# Fifth Normal Form / Projected Normal Form (5NF)



5NF



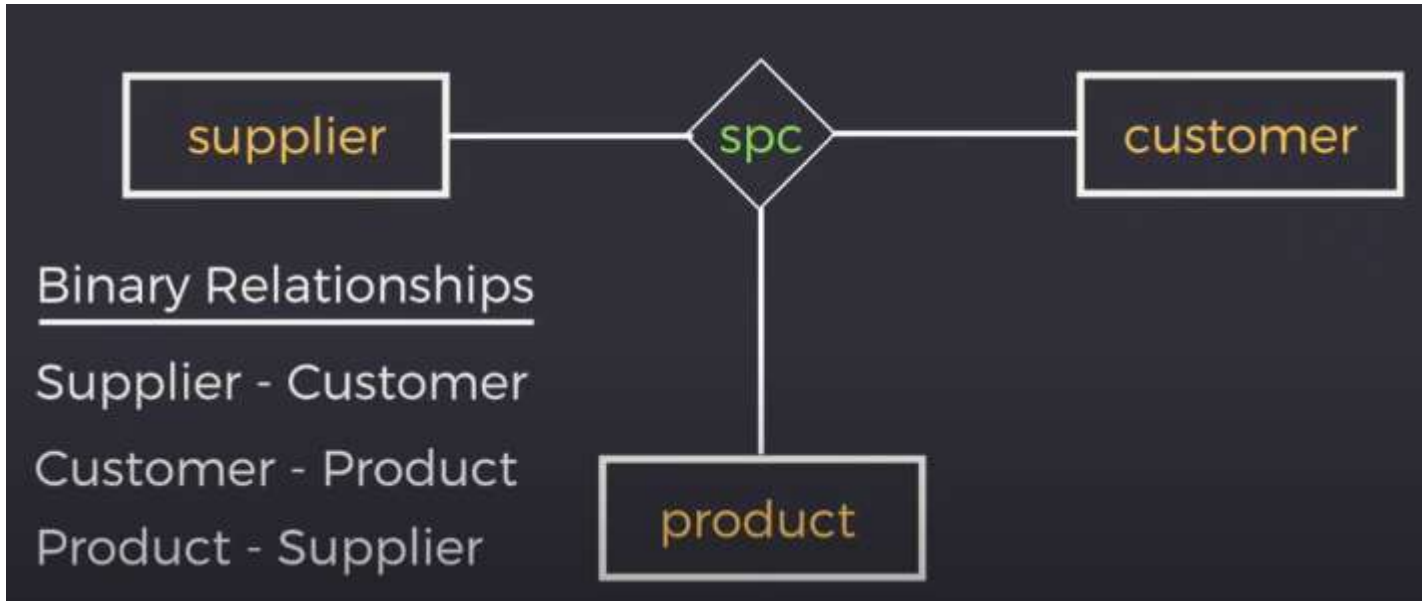




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

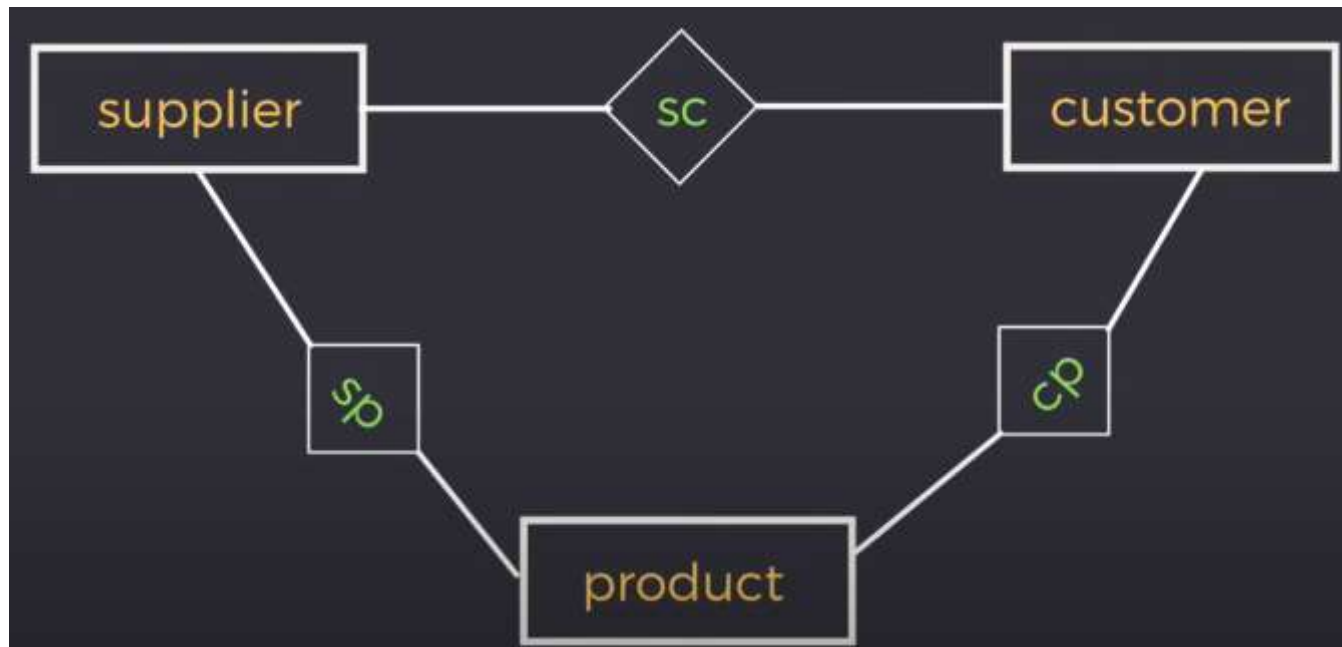




# Fifth Normal Form / Projected Normal Form (5NF)



5NF

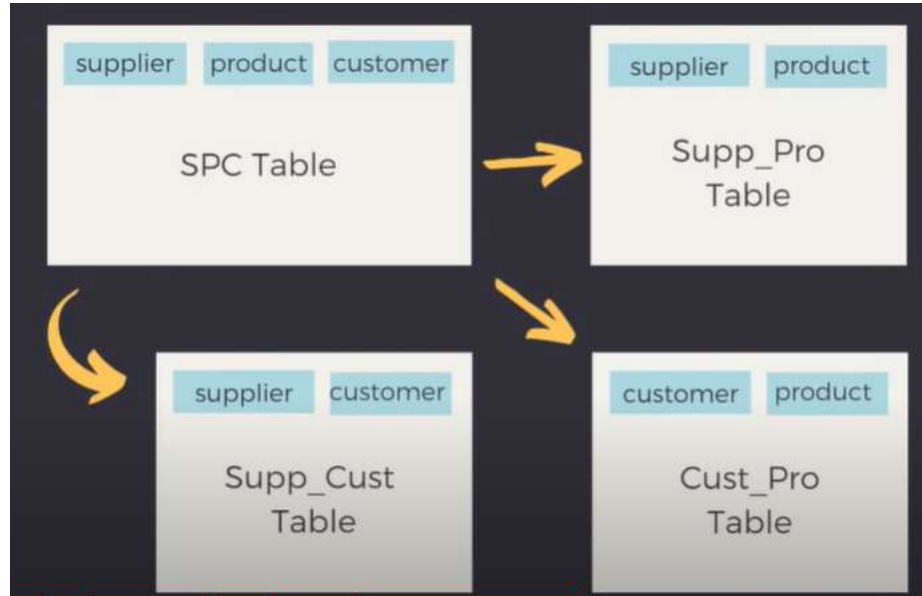




# Fifth Normal Form / Projected Normal Form (5NF)



5NF





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



5NF

SPC Table

supplier	product	customer
ACME	72X SW	FORD
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	GM
ALWAT	OBD II	MERCEDES



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



5NF

SPC Table

supplier	product	customer
ACME	72X SW	FORD
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	GM
ALWAT	OBD II	MERCEDES
ALWAT	GEAR L	MERCEDES

SUPP\_PRO TABLE

supplier	product
ACME	72X SW

SUPP\_CUST TABLE

supplier	customer
ACME	FORD

CUST\_PRO TABLE

customer	product
FORD	72X SW

*SELLS*

*SUPPLIES TO*

*USES*



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



5NF

{  
ACME sells 72X SW  
FORD uses 72X SW  
ACME supplies to FORD  
}

ACME sells ~~X~~ SW to FORD

SUPP\_PRO TABLE

supplier	product
ACME	72X SW

SUPP\_CUST TABLE

supplier	customer
ACME	FORD

CUST\_PRO TABLE

customer	product
FORD	72X SW

SPC Table

supplier	product	customer
ACME	72X SW	FORD



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



5NF

SPC Table

supplier	product	customer
ACME	72X SW	FORD

But it's **true**

as per the **original** table

**ACME** does sell **72X SW** to **FORD**



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



5NF

SPC Table

supplier	product	customer
ACME	72X SW	<del>FORD</del>
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	
ALWAT	OBD II	MERCEDES

Ford is still using the 72X SW





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



5NF

SUPP_PRO TABLE	
supplier	product
ACME	<b>sells</b> 72X SW

SUPP_CUST TABLE	
supplier	customer
ACME	FORD

CUST_PRO TABLE	
customer	product
FORD	<b>uses</b> 72X SW

SPC Table

supplier	product	customer
ACME	72X SW	<del>FORD</del>
ACME	GEAR L	
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	
ALWAT	OBD II	MERCEDES

*Ford might be buying GEAR L from ACME*



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



5NF

ACME  
sells 72X SW  
to FORD

supplier	product
ACME	72X SW

supplier	customer
ACME	FORD

customer	product
FORD	72X SW

SPC Table

supplier	product	customer
ACME	72X SW	FORD
ACME	GEAR L	GM
ROBUSTO	E SWITCH	FORD
ROBUSTO	OBD II	MERCEDES
ALWAT	72X SW	GM
ALWAT	OBD II	MERCEDES



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



5NF

ACME  
sells 72X SW  
to FORD



>// LOSS OF INFORMATION...

SUPP\_PRO TABLE

supplier	product
ACME	72X SW

SUPP\_CUST TABLE

supplier	customer
ACME	FORD

CUST\_PRO TABLE

customer	product
FORD	72X SW

Is this table in 5th Normal Form?



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



5NF

SPC Table

supplier	product	customer
ACME	72X SW	FORD

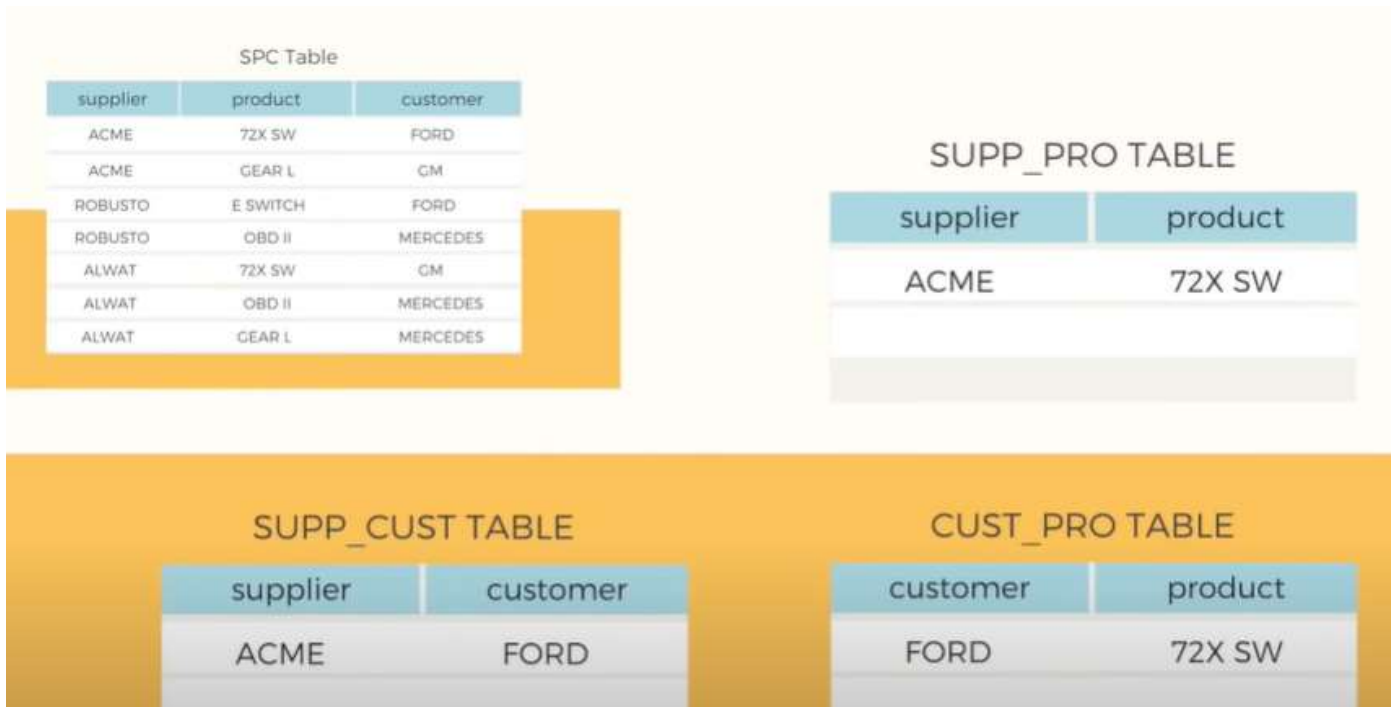
This table does satisfy the  
**5th Normal Form**



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



5NF





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



5NF

isn't that what  
**JOIN Dependency**  
says?

Business Logic  
+  
Data collection



Additional information is  
created or information is lost.



5NF

If decomposing table  
may lead to  
Information Loss or creation

Stick to the original table



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



5NF

But if breaking down the table  
doesn't lead to Information loss  
then decompose the table

And that's how 5NF works





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



A relation R is in [Fifth Normal Form](#) if and only if every functional dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have [lossless 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 the 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:



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



## 5NF

Table ACP

Agent	Company	Product
A1	PQR	Nut
A1	PQR	Bolt
A1	XYZ	Nut
A1	XYZ	Bolt
A2	PQR	Nut



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



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

Table R1

Agent	Company
A1	PQR
A1	XYZ
A2	PQR

Table R2

Agent	Product
A1	Nut
A1	Bolt
A2	Nut

Table R3

Company	Product
PQR	Nut
PQR	Bolt
XYZ	Nut
XYZ	Bolt

The result of the Natural Join of R1 and R3 over 'Company' and then the [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](#).



# References



<https://www.javatpoint.com/dbms-first-normal-form>

<https://www.geeksforgeeks.org/first-normal-form-1nf/>

<https://www.studytonight.com/dbms/second-normal-form.php>