

Database Management System

NORMALIZATION



The Problem of redundancy in Database

- **Redundancy** means having multiple copies of same data in the database.
- This problem arises when a database is not normalized.
- Suppose a table of student details attributes are:
- student Id, student name, college name, college rank, course opted.

Redundancy?

Student_ID	Name	Contact	College	Course	Rank
100	Himanshu	7300934851	GEU	Btech	1
101	Ankit	7900734858	GEU	Btech	1
102	Aysuh	7300936759	GEU	Btech	1
103	Ravi	7300901556	GEU	Btech	1



Insertion Anomaly

- If a student detail has to be inserted whose course is not being decided yet then insertion will not be possible till the time course is decided for student.



Deletion Anomaly

If the details of students in this table is deleted then the details of college will also get deleted which should not occur by common sense.

This anomaly happens when deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.

It is not possible to delete some information without losing some other information in the table as well.



Update Anomaly

Suppose if the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.

Functional Dependency

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

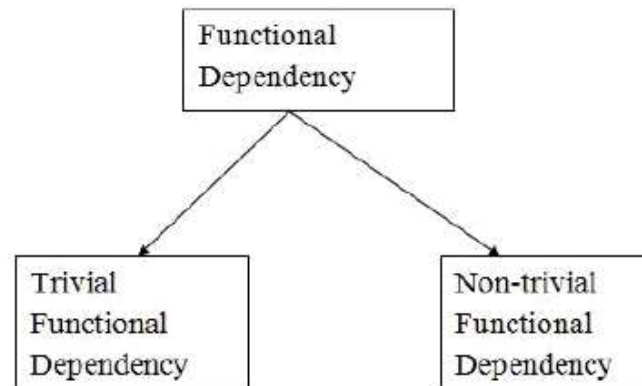
$$X \rightarrow Y$$


- The left side of FD is known as a determinant, the right side of the production is known as a dependent.

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 know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as:
Emp_Id \rightarrow Emp_Name

Types of Functional dependency



- 
- $A \rightarrow B$ has trivial functional dependency if B is a subset of A .
The following dependencies are also trivial like:

$A \rightarrow A, B \rightarrow B$

Example

- Consider a table with two columns Employee_Id and Employee_Name.
- $\{\text{Employee_id}, \text{Employee_Name}\} \rightarrow \text{Employee_Id}$ is a trivial functional dependency as Employee_Id is a subset of $\{\text{Employee_Id}, \text{Employee_Name}\}$.

Non-trivial functional dependency

- $A \rightarrow B$ has a non-trivial functional dependency if B is not a subset of A .
- When $A \cap B$ is NULL, then $A \rightarrow B$ is called as complete non-trivial.

Example:

$ID \rightarrow Name,$

$Name \rightarrow DOB$



Normalization of Database

- Database Normalization is a technique of organizing the data in the database.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like
- Insertion, Update and Deletion anomalies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.



Normalization is used for mainly two purposes,

- Eliminating redundant(useless) data.❏
- Ensuring data dependencies make sense i.e data is logically stored.❏



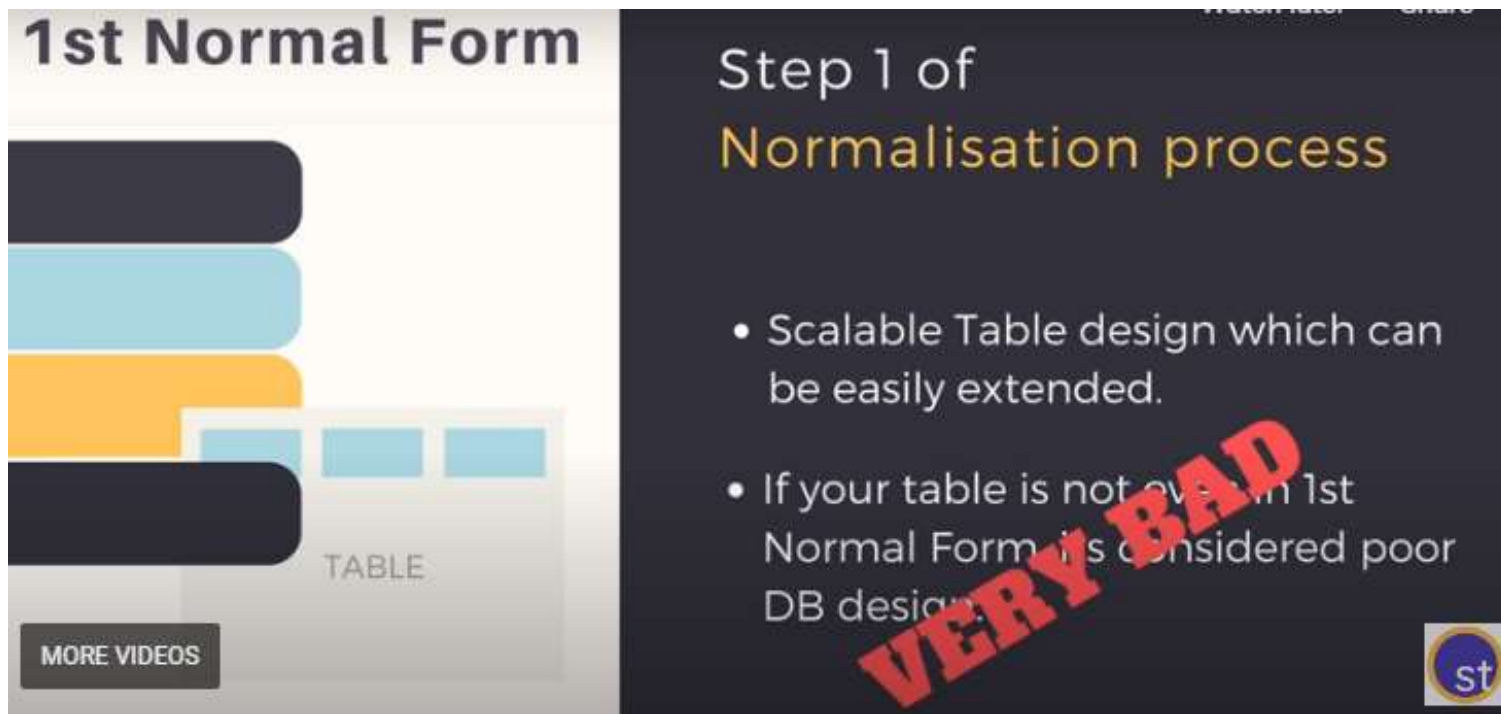
Normalization Rule

- Normalization rules are divided into the following normal forms:
- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

First Normal Form (1NF)

- For a table to be in the First Normal Form, it should follow the following 4 rules:
 1. It should only have single(atomic) valued attributes/columns.
 2. Values stored in a column should be of the same domain
 3. All the columns in a table should have unique names.
 4. And the order in which data is stored, does not matter.

1st Normal Form



1st Normal Form

Step 1 of
Normalisation process


- Scalable Table design which can be easily extended.
- If your table is not over in 1st Normal Form it is considered poor DB design.

VERY BAD

MORE VIDEOS

TABLE

st

- 
- Every table in the Database should atleast follow the 1st Normal form, always.

Time for an Example

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

1st Normal Form Conti..

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

- 
- Using the First Normal Form, data redundancy increases, but each row as a whole will be unique.




Second Normal Form (2NF)

For a table to be in the Second Normal Form,


- It should be in the First Normal form.
- And, it should not have Partial Dependency.

What is Dependency?

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat



subject_id	subject_name
1	Java
2	C++
3	Php



score_id	student_id	subject_id	marks	teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher



subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

And our Score table is now in the second normal form, with no partial dependency.

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80



Quick Recap

- For a table to be in the Second Normal form, it should be in the First Normal form and it should not have Partial Dependency.
- Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.
- To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.



Third Normal Form (3NF)

- A table is said to be in the Third Normal Form when,
- It is in the Second Normal form.
- And, it doesn't have Transitive Dependency.

Student Table


student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat
12	Bkon	09-WY	IT	Rajasthan


Subject Table

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

Score Table

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

- 
- In the Score table, we need to store some more information
 - which is the exam name and total marks
 - let's add 2 more columns to the Score table.



score_id	student_id	subject_id	marks	exam_name	total_marks



Requirements for Third Normal Form

- For a table to be in the third normal form,
- It should be in the Second Normal form.
- And it should not have Transitive Dependency.



Transitive Dependency?

- When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

How to remove Transitive Dependency?

Score Table: In 3rd Normal Form

score_id	student_id	subject_id	marks	exam_id

The new Exam table

exam_id	exam_name	total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30



Advantage of removing Transitive Dependency

- The advantage of removing transitive dependency is,
- Amount of data duplication is reduced.
- Data integrity achieved.

Boyce and Codd Normal Form


- It is a higher version of the Third Normal form.
- This form deals with certain type of anomaly that is not handled by 3NF.
- A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.
- For a table to be in BCNF, following conditions must be satisfied:
- R must be in 3rd Normal Form
- and, for each functional dependency ($X \rightarrow Y$), X should be a super Key.

Rules for BCNF

- For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:
- It should be in the **Third Normal Form**.
- And, for any dependency $A \rightarrow B$, A should be a **super key**.
- The second point sounds a bit tricky, right?
- In simple words, it means, that for a dependency $A \rightarrow B$, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

Below we have a college enrolment table with columns `student_id`, `subject` and `professor`.


<code>student_id</code>	<code>subject</code>	<code>professor</code>
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- 
- One student can enrol for multiple subjects. For example, student with **student_id** 101, has opted for subjects - Java & C++
 - For each subject, a professor is assigned to the student.
 - And, there can be multiple professors teaching one subject like we have for Java.


What do you think should be the **Primary Key**?

Well, in the table above **student_id**, **subject** together form the primary key, because using **student_id** and **subject**, we can find all the columns of the table.

One more important point to note here is, one professor teaches only one subject, but one subject may have two different professors.



Hence, there is a dependency between subject and professor here, where subject depends on the professor name.



But, there is one more dependency, **professor** \rightarrow **subject**.
And while **subject** is a prime attribute, **professor** is a non-prime attribute, which is not allowed by BCNF.



How to satisfy BCNF?

- To make this relation(table) satisfy BCNF, we will decompose this table into two tables, **student** table and **professor** table.

Student Table

student_id	p_id
101	1
101	2
and so on...	

And, **Professor Table**

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++



Rules for 4th Normal Form

- For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:
- It should be in the **Boyce-Codd Normal Form**.
- And, the table should not have any **Multi-valued Dependency**.

What is Multi-valued Dependency?


A table is said to have multi-valued dependency, if the following conditions are true,

1. For a dependency $A \twoheadrightarrow B$, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
3. And, for a relation $R(A,B,C)$, if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

Time for an Example

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

- 
- student with **s_id 1** has opted for two courses, **Science** and **Maths**, and has two hobbies, **Cricket** and **Hockey**.
 - Well the two records for student with **s_id 1**, will give rise to two more records, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified.

There is no relationship between the columns course and hobby. They are independent of each other.

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

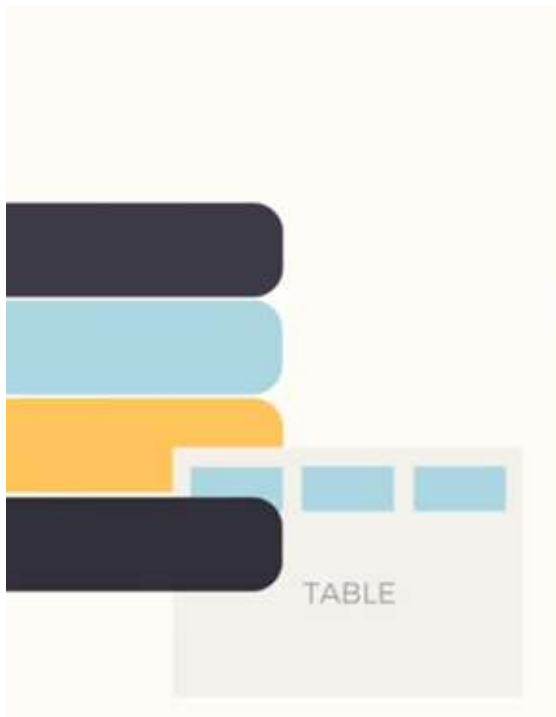
How to satisfy 4th Normal Form?

CourseOpted Table

s_id	course
1	Science
1	Maths
2	C#
2	Php

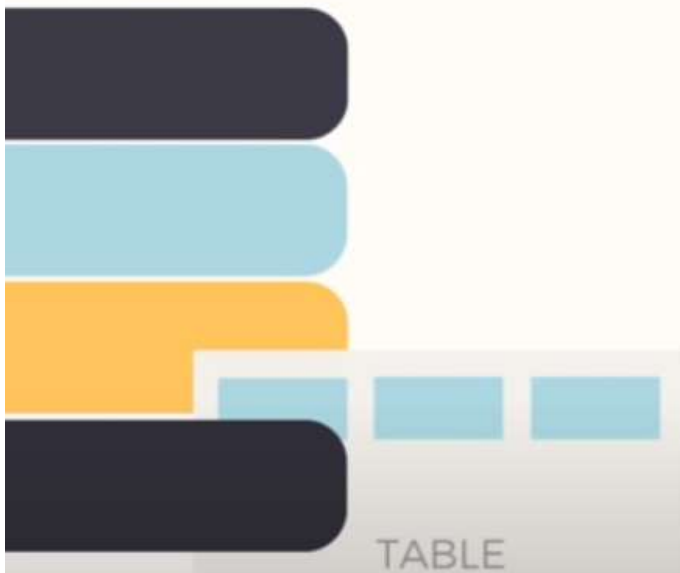
And, **Hobbies Table,**

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey



5th Normal Form

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



5th Normal Form

- PJNF



**Project Join
Normal Form**



A a relation with JOIN dependency

break down

B

C

join again

A

get the same relation again.





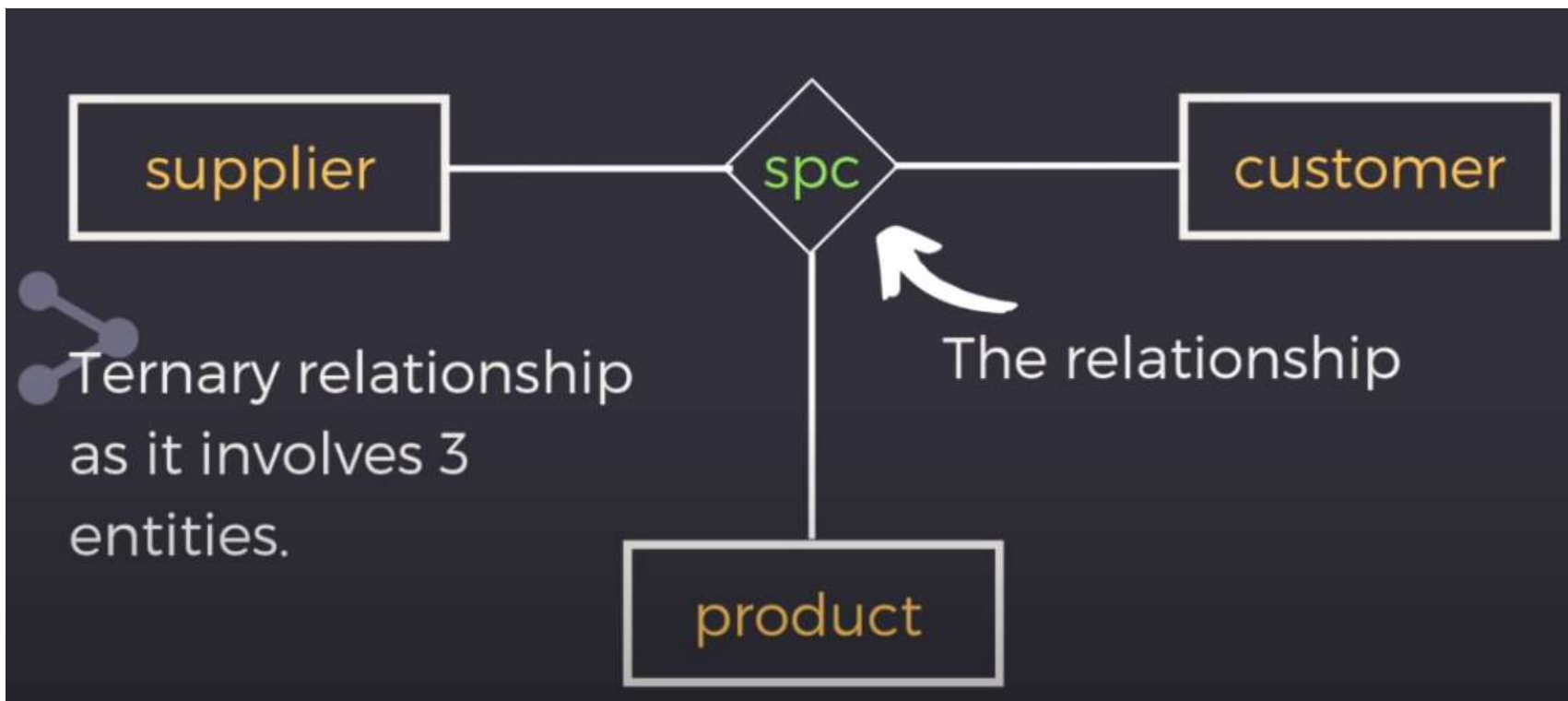
Watch later



If JOIN Dependency **doesn't exist** then
either **data is lost** or **new entries are created**.



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



st 5th Normal Form (5NF) | Join Dependency | Database Normalization

Watch later Share

```
graph LR; supplier[supplier] ---|can supply| product[product]; product --- Customer1[Customer 1]; product --- Customer2[Customer 2];
```

1 product being supplied to multiple customers.

st

3:17 / 8:08 YouTube



1 product being supplied to multiple customers.

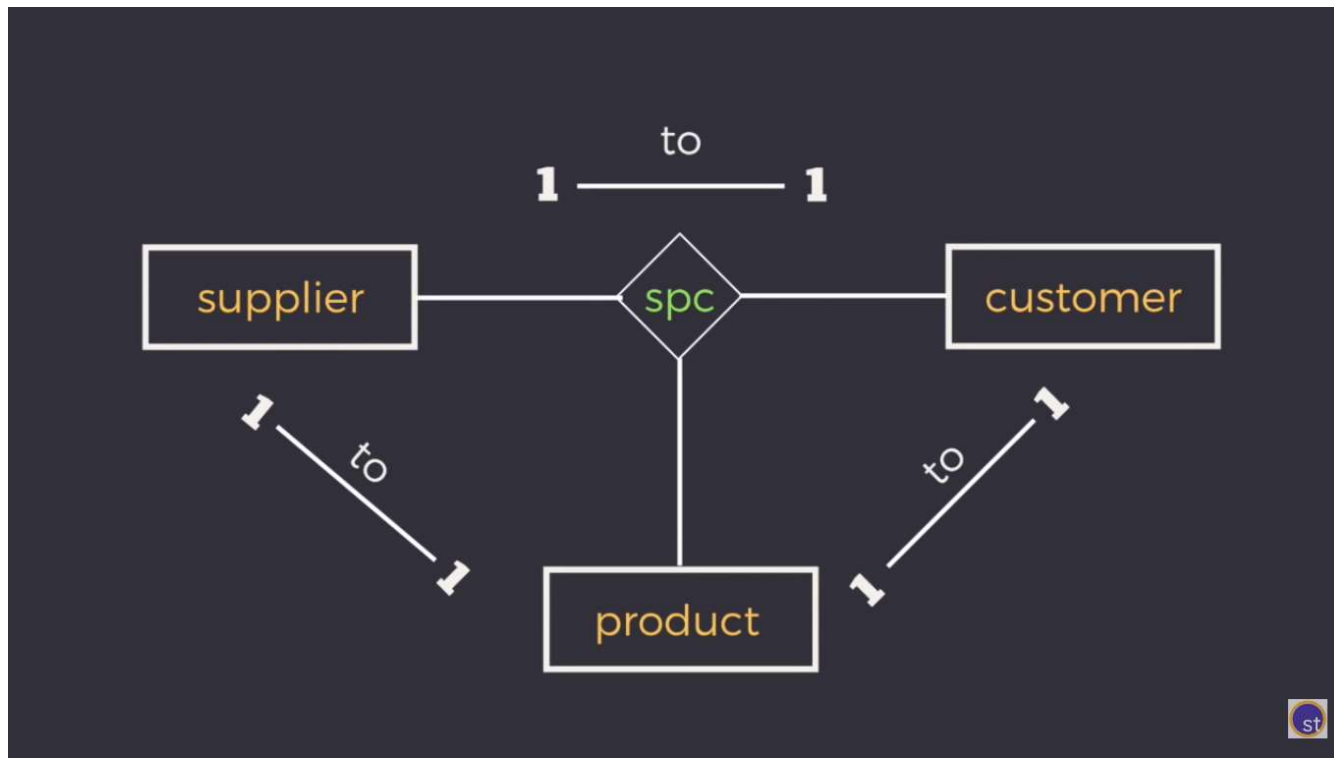


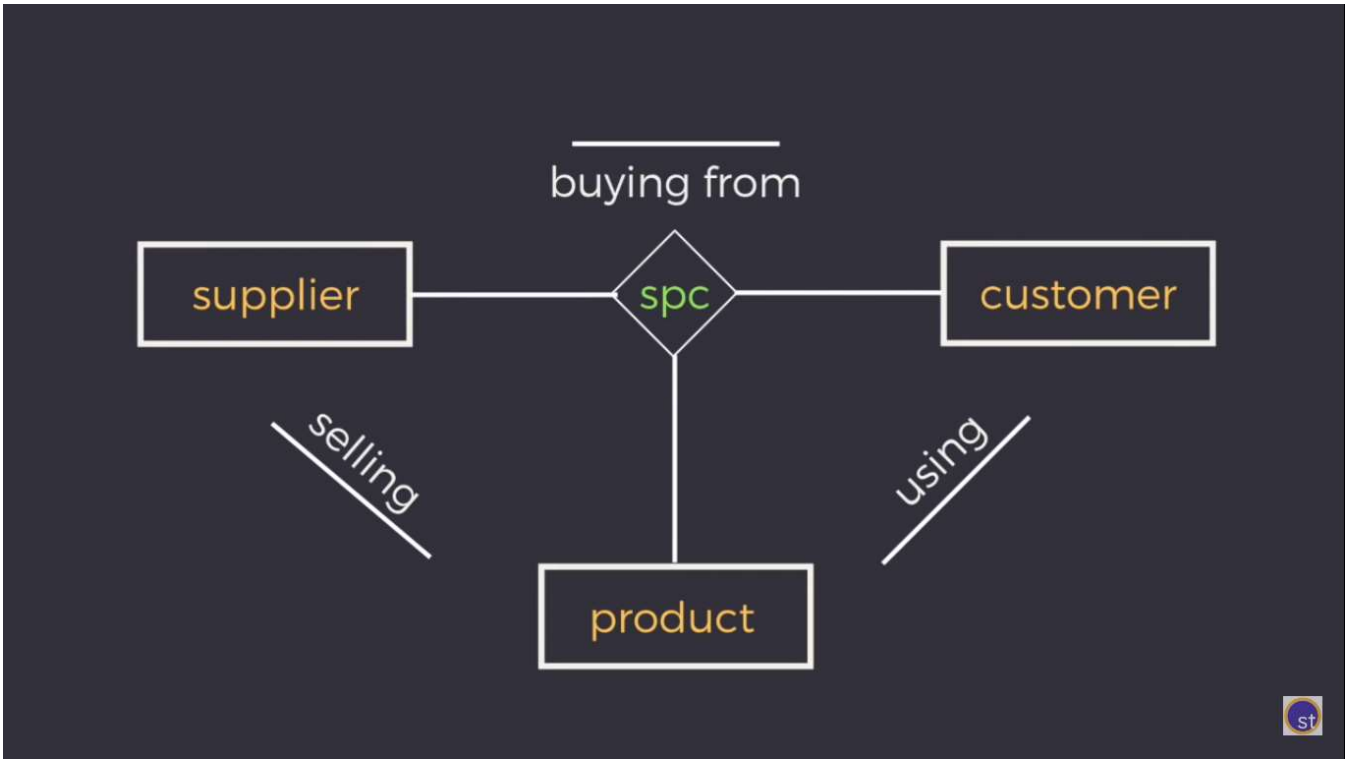
out of the products that
a given supplier supplies

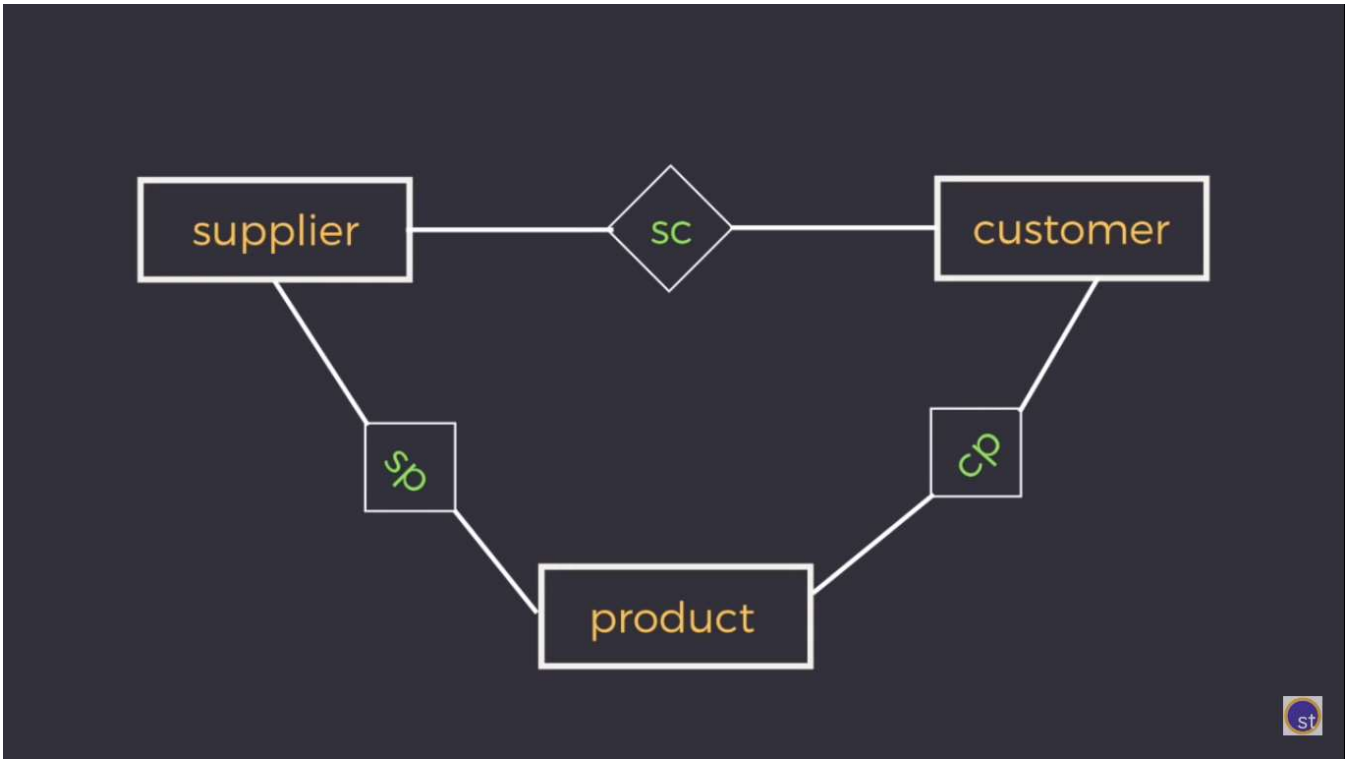


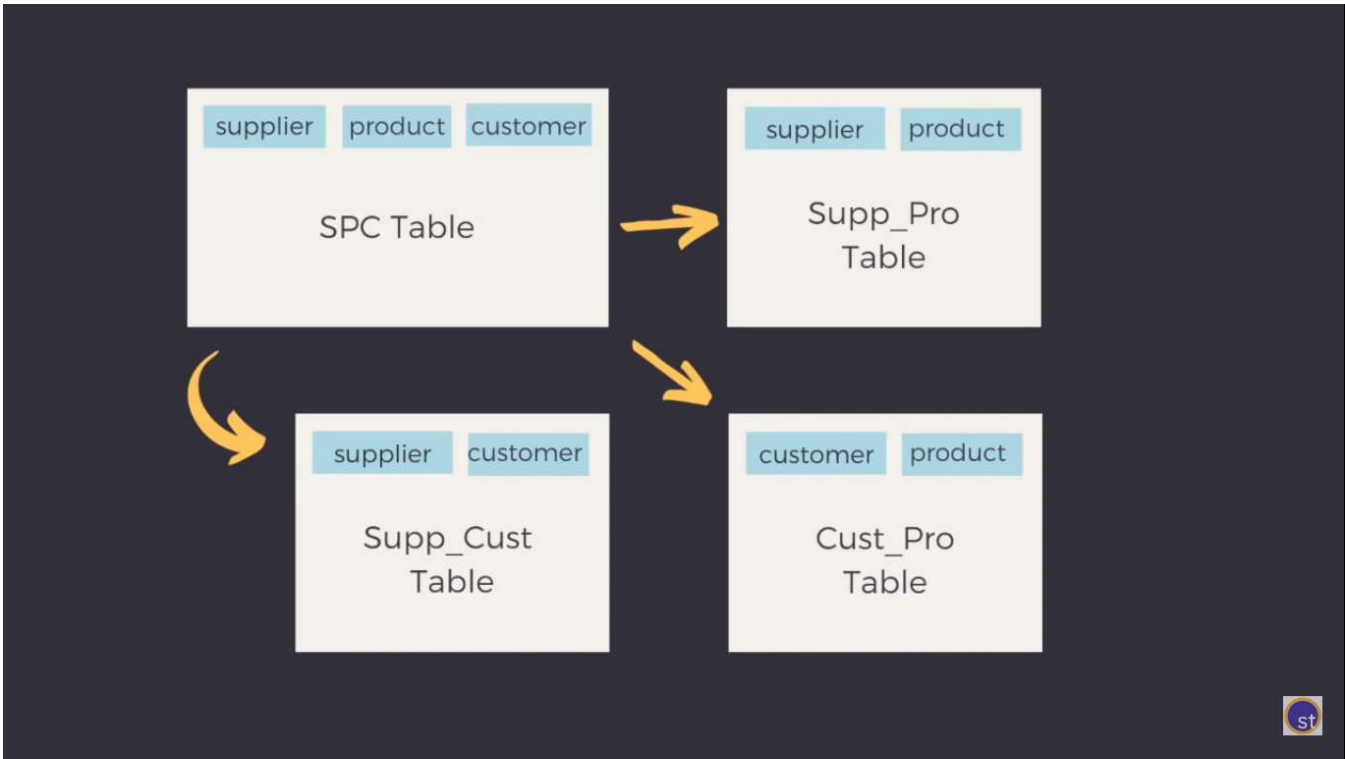
If multiple suppliers supply the same product

One to one Relation









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



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

SELLS

SUPP_CUST TABLE

supplier	customer
ACME	FORD

SUPPLIES TO

CUST_PRO TABLE

customer	product
FORD	72X SW

USES



{
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

But it's **true**

as per the **original** table

SPC Table

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

*Ford might
be buying
GEAR L from
ACME*





Additional information is
created or information is lost.





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





THANK YOU

8-May-24



72