# Information Management 1
# LECTURE

**Title: Rules of Normalization** **Module No: 5**

## I. INTRODUCTION

In order to establish a well-meaning database, we need to distinguish the scheme of Database Normalization. Normalization which can help to manage the appropriate executions and processes of the complete database. It will also determine in setting up a normalized table to have a prone free, or error free within a certain databases and its records itself.

## II. LEARNING OBJECTIVES

After studying this module, you should be able to:

- Understand all about Normalization.
- Identify and understand the different types of Normalization

## III. TOPICS AND KEY CONCEPTS
## A. RULES OF NORMALIZATION

### NORMALIZATION

It is a method used to eliminate redundancy in data definitions, especially for relationship models.

### First normal form

See to it that relations have no multiple-valued attributes

There should be no repeating groups in a table.

***NOT NORMALIZED***

student ID
name
date of birth
advisor
advisor’s telephone
student
course ID 1
course description 1
course instructor 1

***NORMALIZED – FIRST NORMAL FORM***

student ID
name
date of birth
advisor
advisor’s telephone
student

course ID 1
course description 1
course instructor 1

**Second normal form**

Eliminate functional dependencies on a partial key by putting the fields in a separate table from those that are dependent on the whole key

A functional dependency exists when the value of one thing is fully determined by another.

***NOT NORMALIZED***

| |
|---|
| student ID |
| *student ID* |
| name |
| date of birth |
| advisor |
| advisor’s telephone |
| course ID |
| course description |
| course instructor |

***NORMALIZED – SECOND NORMAL FORM***

| Student table |
|---|
| *student ID* |
| name |
| date of birth |
| advisor |
| advisor’s telephone |

| Student courses table |
|---|
| *student ID* |
| *course ID* |

| Courses table |
|---|
| *course ID* |
| course description |
| course instructor |

## Third normal form

Eliminate functional dependencies on non-key fields by putting them in a separate table. At this stage, all non-key fields are dependent on the key, the whole key and nothing but the key.

No fields may depend on other non-key fields. In other words, each field in a record should contain information about the entity that is defined by the primary key.

### *NORMALIZED – THIRD NORMAL FORM*

| Student table |
|---|
| *student ID*<br>name<br>date of birth<br>advisor ID |

| Student courses table |
|---|
| *student ID*<br>*course ID* |

| Courses table |
|---|
| *course ID*<br>course description<br>course instructor |

| Advisor table |
|---|
| *advisor ID*<br>advisor name<br>advisor telephone |

## Boyce – codd normal form

All dependencies must be explicitly shown through keys.There cannot be a hidden dependency between nonkey and key columns.

Primary key a field that uniquely identifies a record in a table.

Many to many relationship.

### *NORMALIZED – BOYCE-CODD NORMAL FORM*

| Student table |
|---|
| *student ID*<br>name<br>date of birth<br>advisor ID<br>COURSE ID |

| Courses table |
|---|
| course ID<br>course description<br>course instructor<br>Student ID<br>ADVISOR ID |

| Advisor table |
|---|
| *advisor ID*<br>advisor name<br>advisor telephone<br>Student ID<br>Course ID |

## Fourth normal form

Eliminate all multivalued dependencies.

Multivalued dependency is the type of dependency that exists when there are at least three attributes (e.g., A, B, and C) in a relation, with a well-defined set of B and C values for each A value, but those B and C values are independent of each other.

Many to one relationship or one to many relationship.

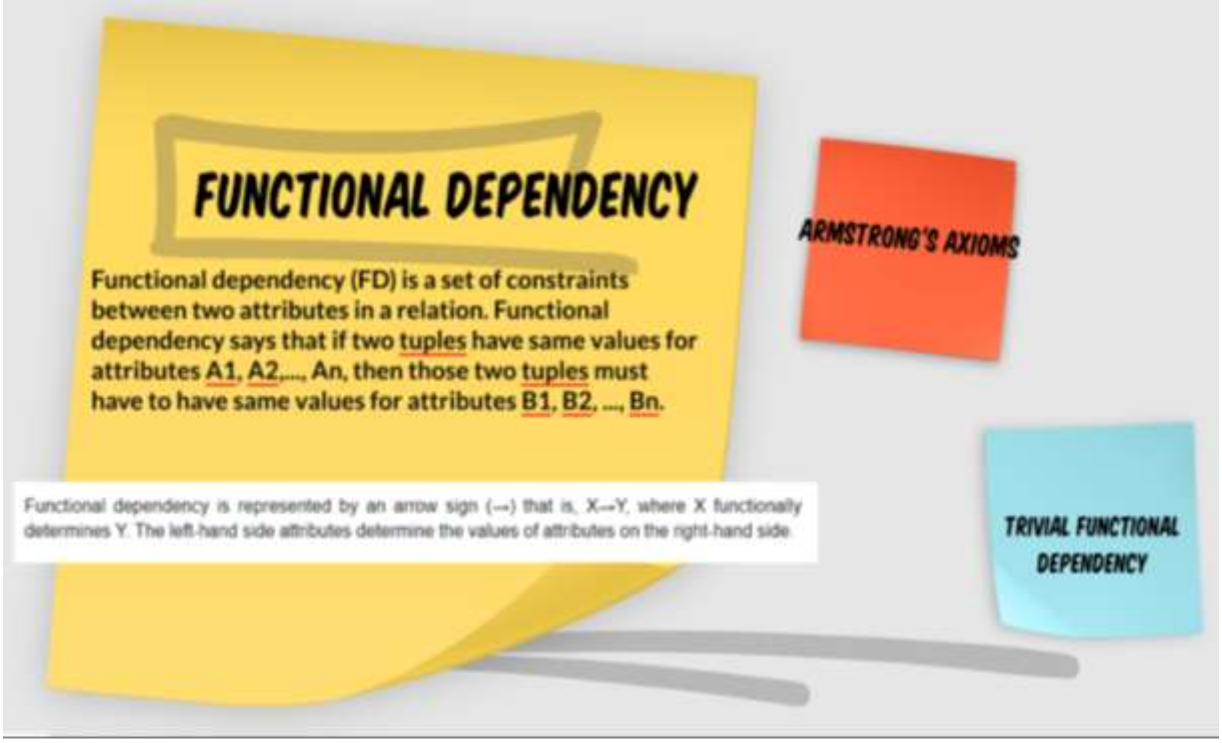
***NORMALIZED – FOURTH NORMAL FORM***

| Student table |
|---|
| student ID<br>name<br>date of birth |

| Courses table |
|---|
| course ID<br>course description<br>course instructor<br>Student ID |

| Advisor table |
|---|
| *advisor ID*<br>advisor name<br>advisor telephone<br>Student ID |

## B. DBMS NORMALIZATION



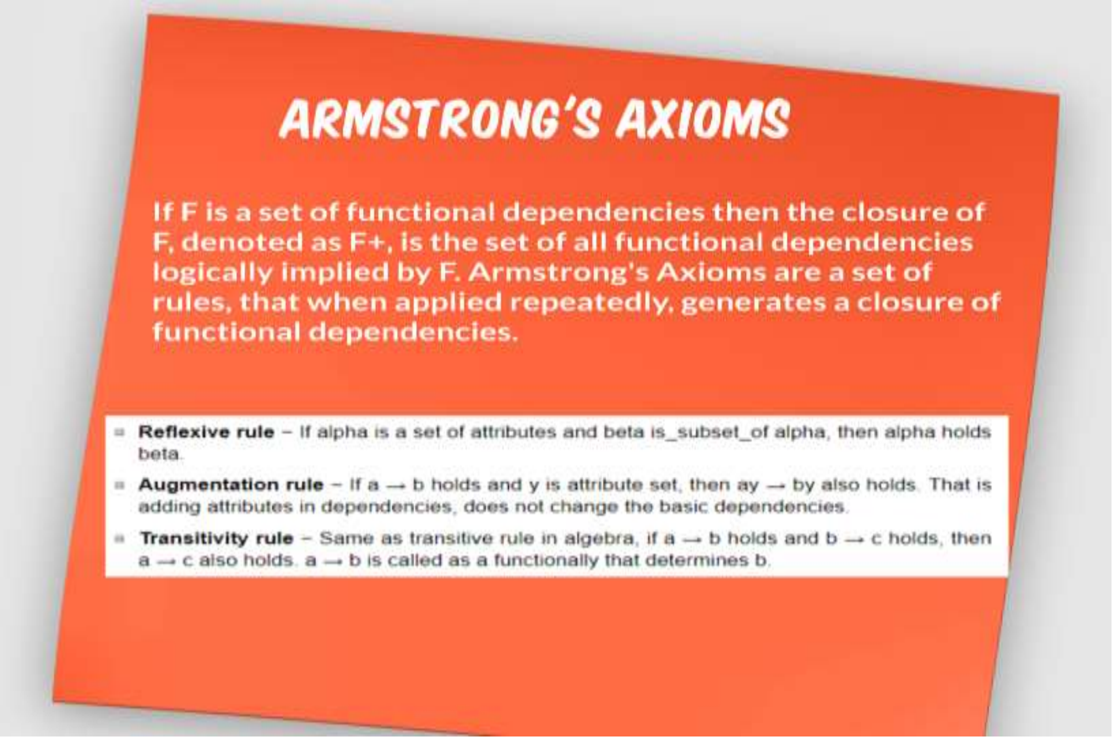
### FUNCTIONAL DEPENDENCY

Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

ARMSTRONG'S AXIOMS

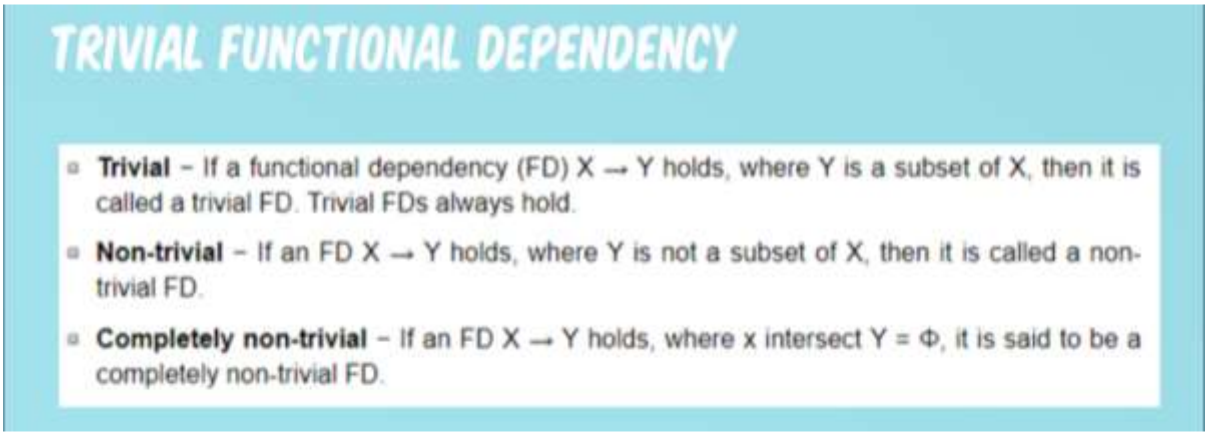
TRIVIAL FUNCTIONAL DEPENDENCY



### ARMSTRONG'S AXIOMS

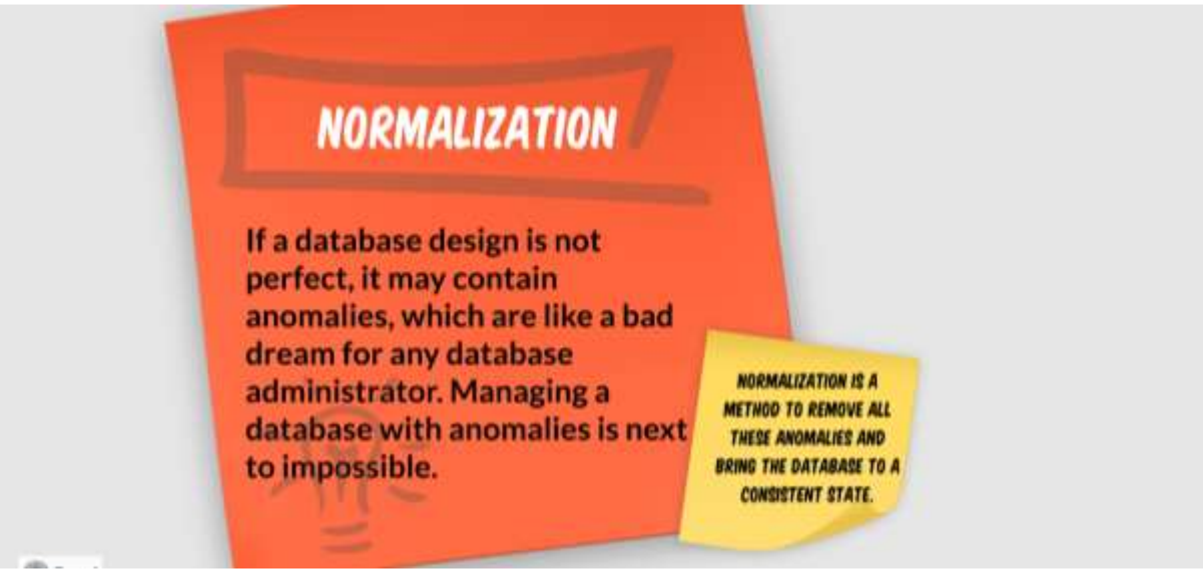
If F is a set of functional dependencies then the closure of F, denoted as F+, is the set of all functional dependencies logically implied by F. Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

- **Reflexive rule** – If alpha is a set of attributes and beta is_subset_of alpha, then alpha holds beta.
- **Augmentation rule** – If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.
- **Transitivity rule** – Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.



TRIVIAL FUNCTIONAL DEPENDENCY

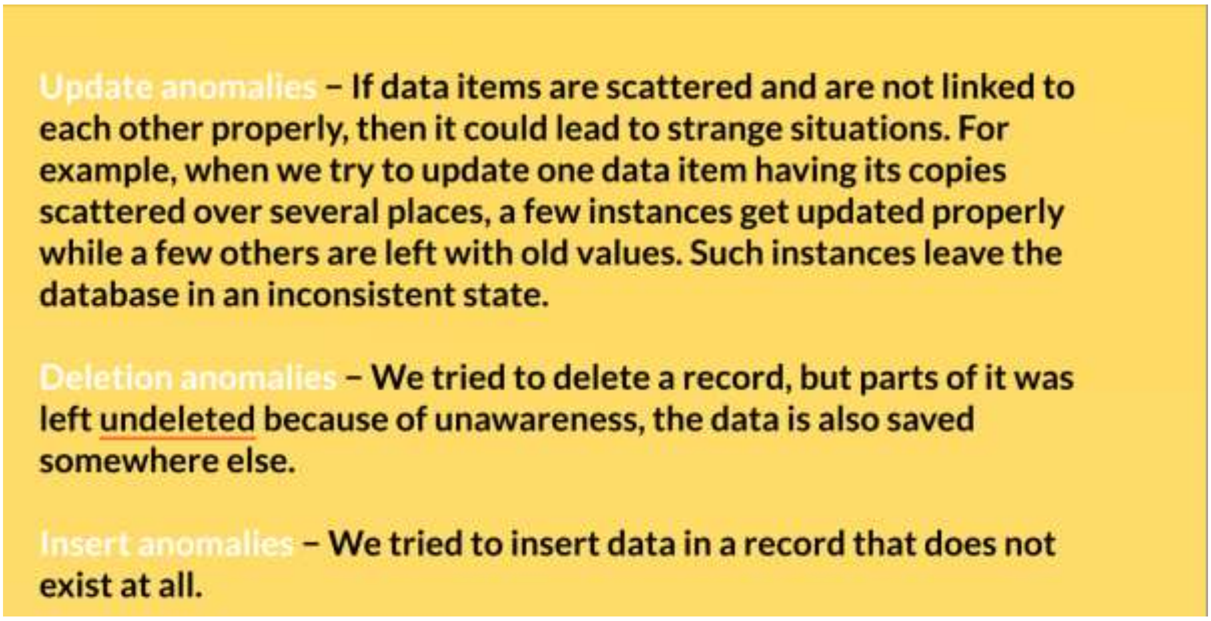
- **Trivial** – If a functional dependency (FD) X → Y holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.
- **Non-trivial** – If an FD X → Y holds, where Y is not a subset of X, then it is called a non-trivial FD.
- **Completely non-trivial** – If an FD X → Y holds, where x intersect Y = Φ, it is said to be a completely non-trivial FD.



NORMALIZATION

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

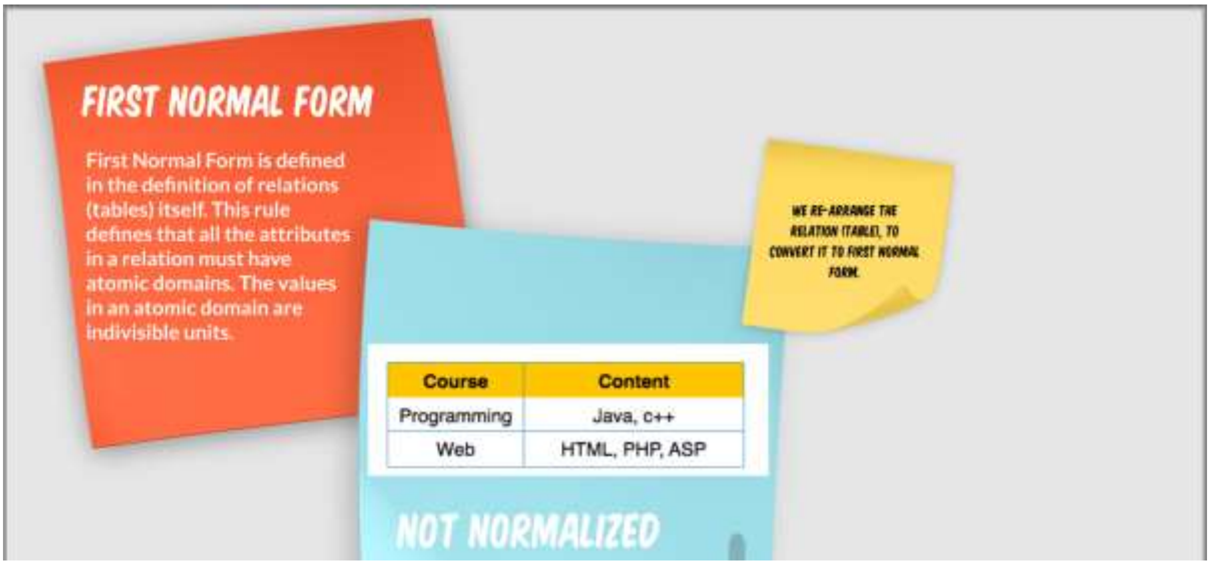
NORMALIZATION IS A METHOD TO REMOVE ALL THESE ANOMALIES AND BRING THE DATABASE TO A CONSISTENT STATE.



Update anomalies – If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.

Deletion anomalies – We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.

Insert anomalies – We tried to insert data in a record that does not exist at all.



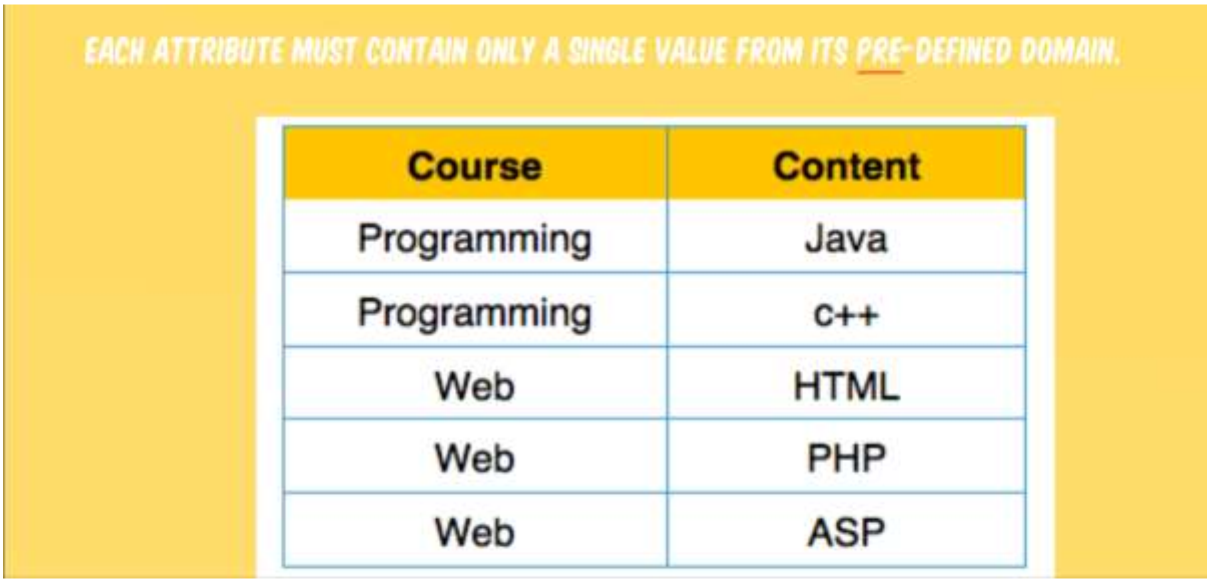
## FIRST NORMAL FORM

First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

WE RE-ARRANGE THE RELATION (TABLE), TO CONVERT IT TO FIRST NORMAL FORM.

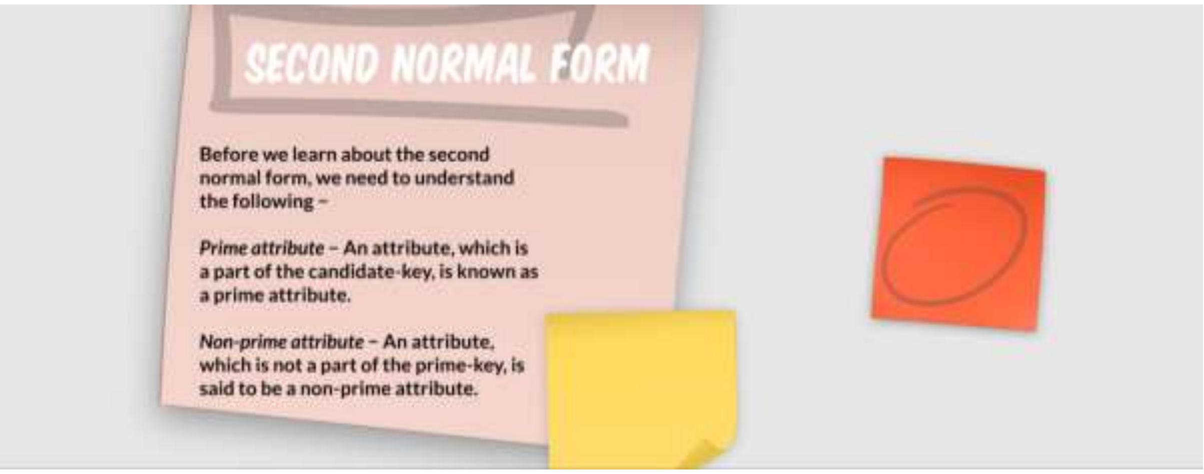
| Course | Content |
|---|---|
| Programming | Java, c++ |
| Web | HTML, PHP, ASP |

NOT NORMALIZED



EACH ATTRIBUTE MUST CONTAIN ONLY A SINGLE VALUE FROM ITS PRE-DEFINED DOMAIN.

| Course | Content |
|---|---|
| Programming | Java |
| Programming | c++ |
| Web | HTML |
| Web | PHP |
| Web | ASP |

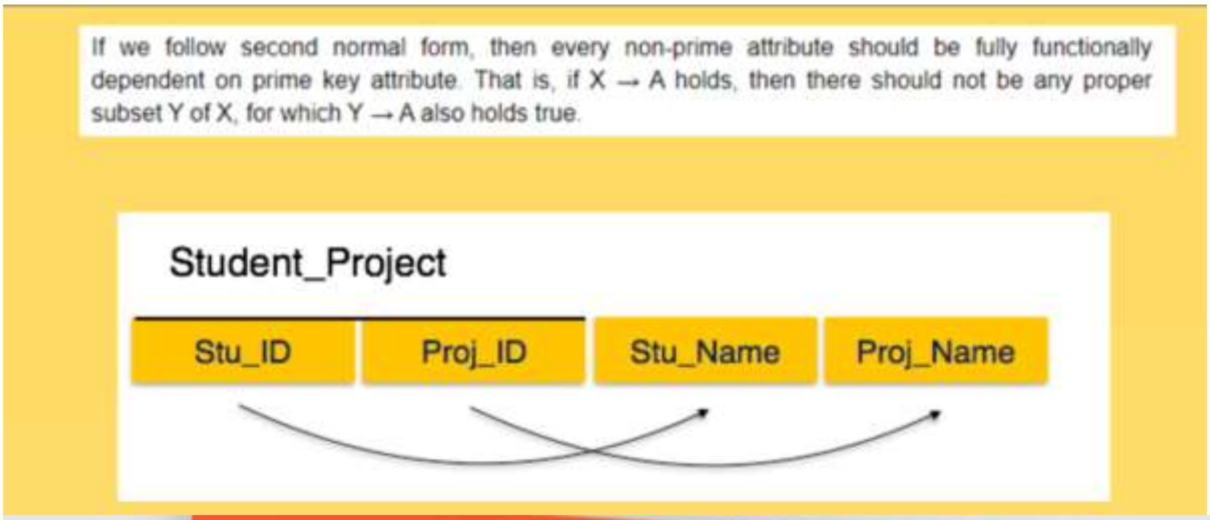


## SECOND NORMAL FORM

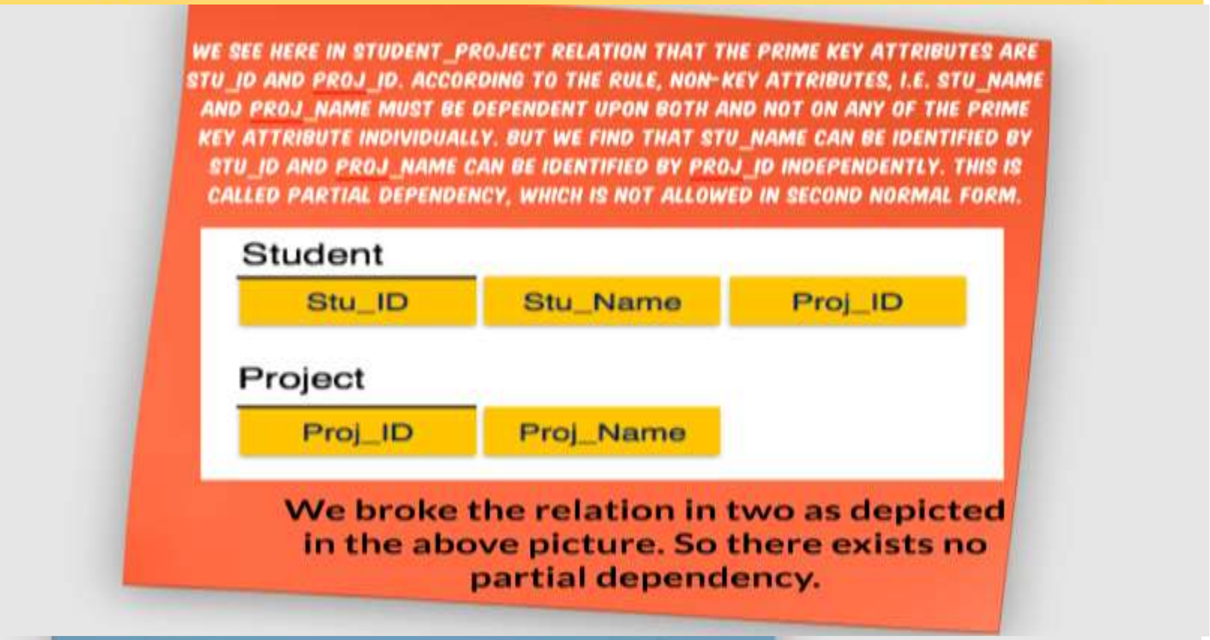
Before we learn about the second normal form, we need to understand the following –

*Prime attribute* – An attribute, which is a part of the candidate-key, is known as a prime attribute.

*Non-prime attribute* – An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

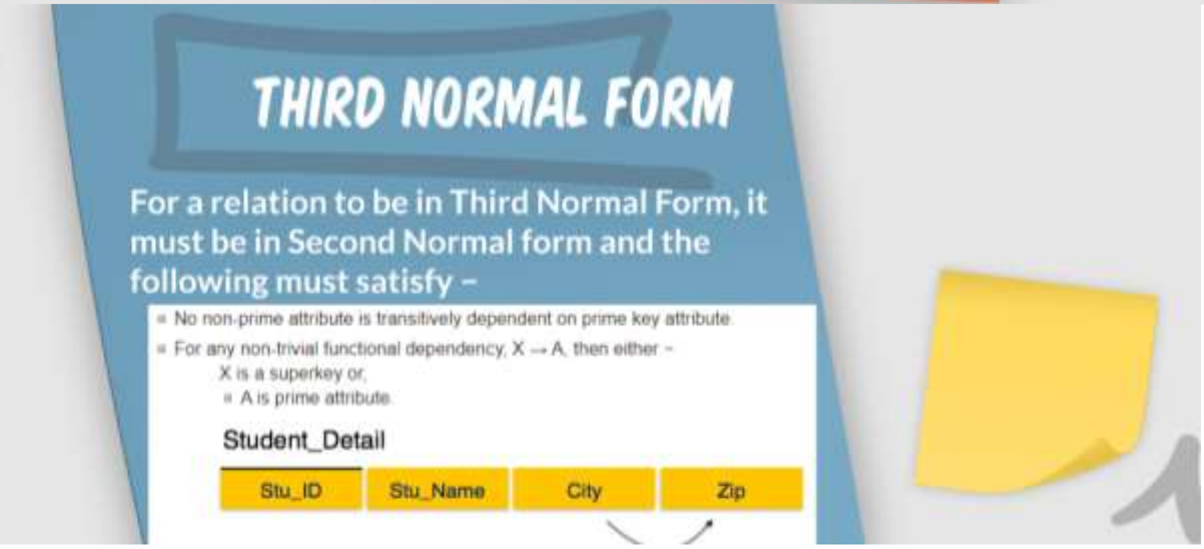


If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



WE SEE HERE IN STUDENT_PROJECT RELATION THAT THE PRIME KEY ATTRIBUTES ARE STU_ID AND PROJ_ID. ACCORDING TO THE RULE, NON-KEY ATTRIBUTES, I.E. STU_NAME AND PROJ_NAME MUST BE DEPENDENT UPON BOTH AND NOT ON ANY OF THE PRIME KEY ATTRIBUTE INDIVIDUALLY. BUT WE FIND THAT STU_NAME CAN BE IDENTIFIED BY STU_ID AND PROJ_NAME CAN BE IDENTIFIED BY PROJ_ID INDEPENDENTLY. THIS IS CALLED PARTIAL DEPENDENCY, WHICH IS NOT ALLOWED IN SECOND NORMAL FORM.

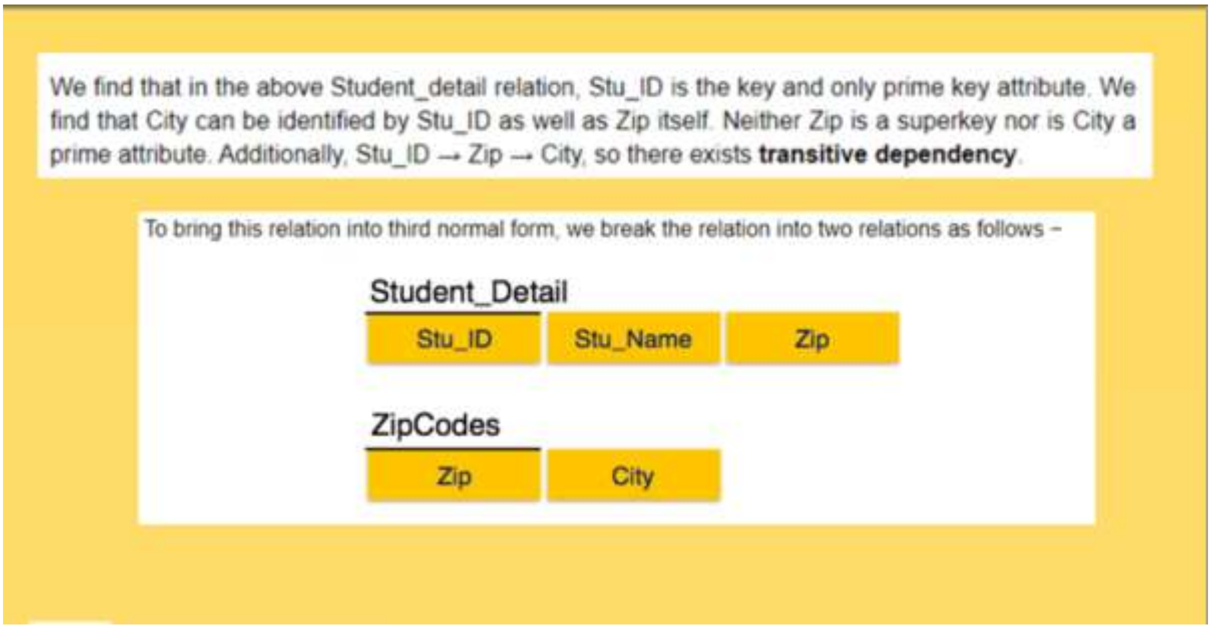
We broke the relation in two as depicted in the above picture. So there exists no partial dependency.



## THIRD NORMAL FORM

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy –

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either –
  X is a superkey or,
  - A is prime attribute.



We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

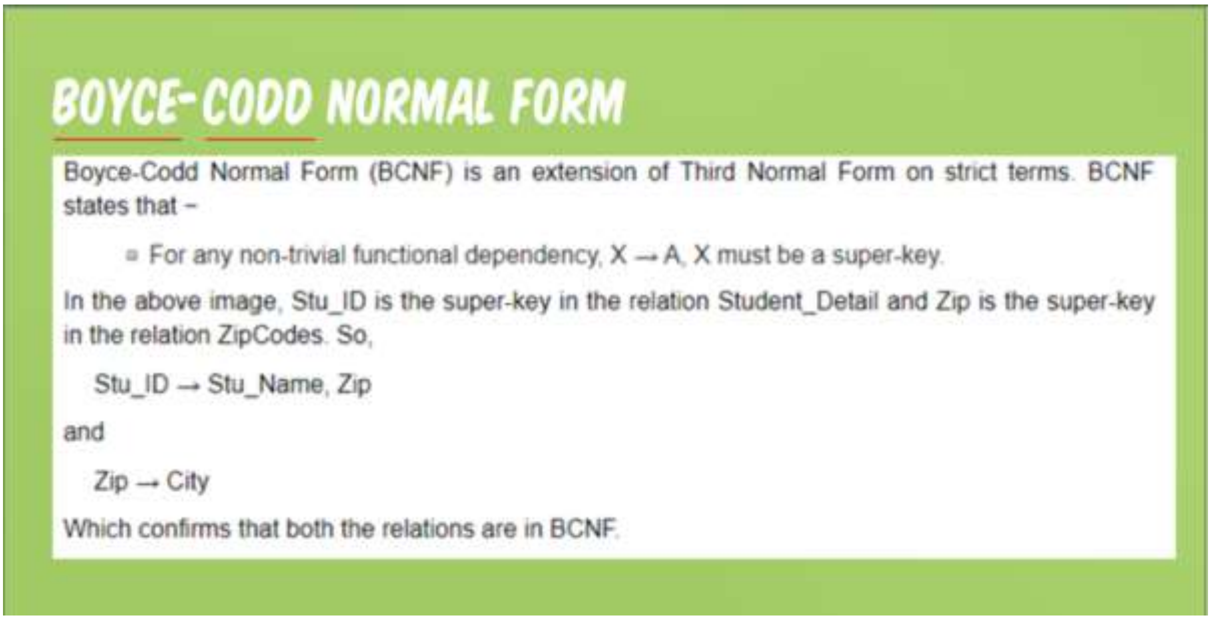
To bring this relation into third normal form, we break the relation into two relations as follows −

Student_Detail

| Stu_ID | Stu_Name | Zip |
|---|---|---|

ZipCodes

| Zip | City |
|---|---|



## BOYCE-CODD NORMAL FORM

Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,

Stu_ID → Stu_Name, Zip

and

Zip → City

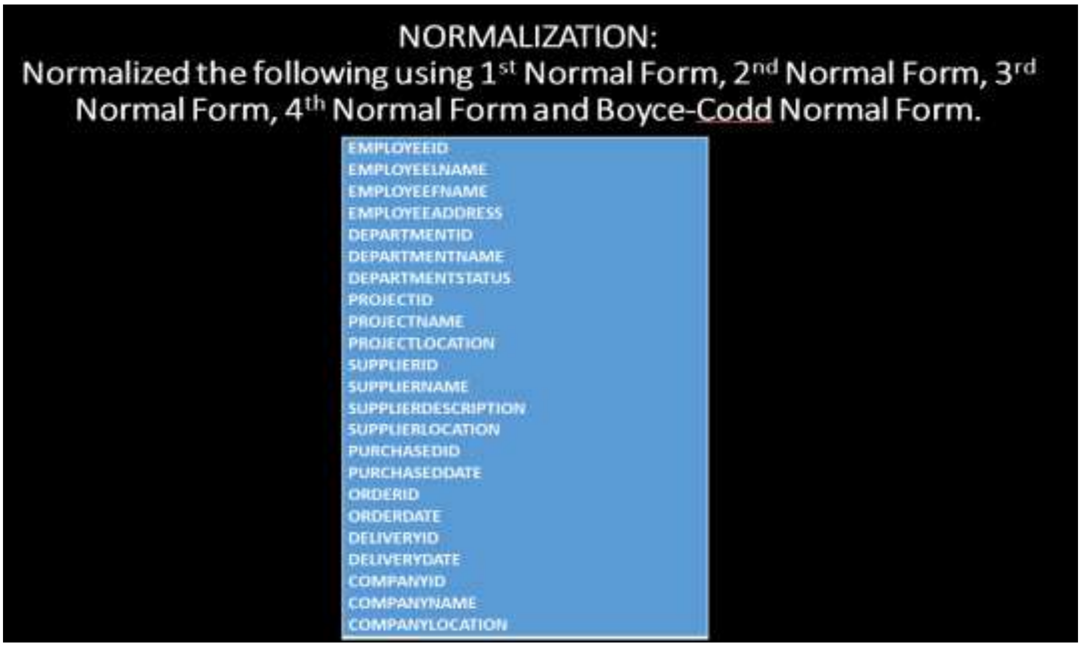
Which confirms that both the relations are in BCNF.

## IV. TEACHING AND LEARNING MATERIALS RESOURCES

- PC Computer || Laptop || Smartphone
- Internet Connection
- Browsers
- Any available Programming Software
- GC-LAMP
- Google Classroom
- Google Meet
- Facebook Group
- Facebook Messenger
- For online activity sites:
    - ✓ https://www.blogger.com/about/?r=1-null_user
    - ✓ https://www.wix.com/html5bing/hiker-blog?utm_source=bing&utm_medium=cpc&utm_campaign=ms_en_e_1_NEW^bl_blogging_rest&experiment_id=blogging^be^79714673617818^blogging&msclkid=983ab99d6f3e1cb92c8de6b674948445

## V. LEARNING TASKS

### A. ENGAGE



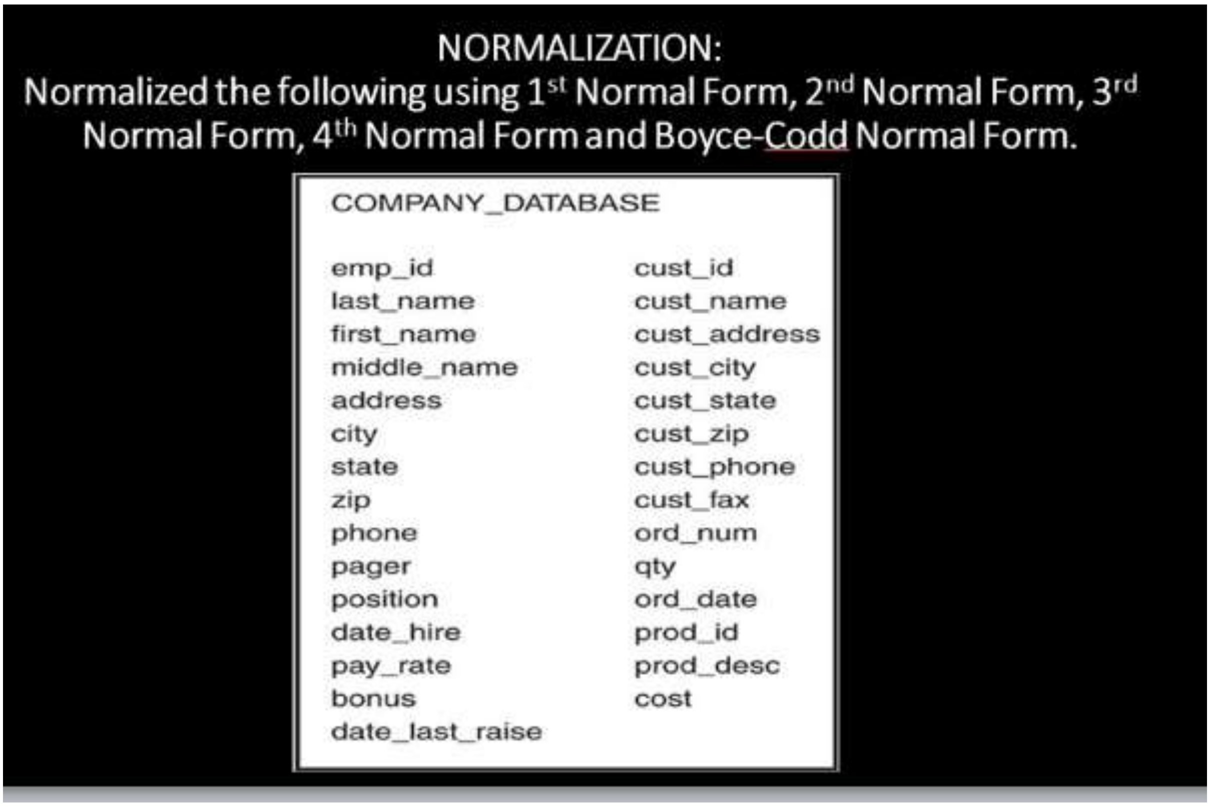
NORMALIZATION:
Normalized the following using 1st Normal Form, 2nd Normal Form, 3rd Normal Form, 4th Normal Form and Boyce-Codd Normal Form.

EMPLOYEEID
EMPLOYEELNAME
EMPLOYEEFNAME
EMPLOYEEADDRESS
DEPARTMENTID
DEPARTMENTNAME
DEPARTMENTSTATUS
PROJECTID
PROJECTNAME
PROJECTLOCATION
SUPPLIERID
SUPPLIERNAME
SUPPLIERDESCRIPTION
SUPPLIERLOCATION
PURCHASEDID
PURCHASEDDATE
ORDERID
ORDERDATE
DELIVERYID
DELIVERYDATE
COMPANYID
COMPANYNAME
COMPANYLOCATION

Rubric:

| | | | | | |
|---|---|---|---|---|---|
| Completed the activities and understood the topic based on the given answer | Outstanding 50 points | Very Good 40 points | Good 30 points | Fair 20 points | No Work Output |

## B. EXPLORE & EXPLAIN



NORMALIZATION:

Normalized the following using 1st Normal Form, 2nd Normal Form, 3rd Normal Form, 4th Normal Form and Boyce-Codd Normal Form.

COMPANY_DATABASE

emp_id
last_name
first_name
middle_name
address
city
state
zip
phone
pager
position
date_hire
pay_rate
bonus
date_last_raise

cust_id
cust_name
cust_address
cust_city
cust_state
cust_zip
cust_phone
cust_fax
ord_num
qty
ord_date
prod_id
prod_desc
cost

Rubric:

| | | | | | |
|---|---|---|---|---|---|
| Completed the activities and understood the topic based on the given answer | Outstanding 50 points | Very Good 40 points | Good 30 points | Fair 20 points | No Work Output |

### C. ELABORATE & EVALUATION

Answer the following questions: **Identification**

____________**1.** Many to one relationship or one to many relationship.

____________**2.** Many to many relationship.

____________**3.** It is a method used to eliminate redundancy in data definitions, especially for relationship models.

____________**4.** No fields may depend on other non-key fields.

____________**5.** There should be no repeating groups in a table.

Rubrics:

| Each correct answer will be given 5 points. Total score = 25 points | Question 1 | Question 2 | Question 3 | Question 4 | Question 5 | Total Score |
|---|---|---|---|---|---|---|
| | | | | | | |

## VI. REFERENCES

- Database System for Management J.F. Courtney, et al. – Global Text Project, 2010
- DBMS Tutorial, retrieved from https://www.tutorialspoint.com/dbms/, retrieved on August 5, 2019
- SQL Tutorial, retrieved from https://www.w3schools.com/sql/, retrieved on August 5, 2019
- http://w3schools.com/sql
- http://www.tutorialspoint.com/java