

**NORMALIZATION:** Normalization is a technique of organizing the data into multiple tables. Normalization process automatically eliminates data redundancy (repetition) and also avoiding Insertion, Update and Deletion problems.

**Problems without Normalization:** If a table is not properly normalized and have data redundancy then it will not only occupy extra memory space but will also make it difficult to handle insert, delete and update operations in student table.

**STUDENT DETAILS**

| Roll no | Name   | Branch | Hod   | Office Number |
|---------|--------|--------|-------|---------------|
| 101     | SAI    | CSE    | Mr. X | 040-53337     |
| 102     | ALLEN  | CSE    | Mr. X | 040-53337     |
| 103     | JAMES  | CSE    | Mr. X | 040-53337     |
| 104     | MILLER | CSE    | Mr. X | 040-53337     |

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields BRANCH, HOD and OFFICE\_NUMBER is repeated for the students who are in the same branch in the college, this is Data Redundancy.

**Insertion problem:**

If we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students. These scenarios are nothing but Insertion problem. Reason for data redundancy is two different related data stored in the same table.

**Student data + Branch data**

### **Updation problem:**

If we want to change HOD name then system admin has to update all students records with new HOD name. and if by mistake we miss any record, it will lead to data inconsistency. This is Updation problem.

Ex: Mr. X leaves and Mr. Y join as a new HOD for CSE. Then the table will be like below,

**STUDENT DETAILS**

| Roll no | Name   | Branch | Hod   | Office Number |
|---------|--------|--------|-------|---------------|
| 101     | SAI    | CSE    | Mr. Y | 040-53337     |
| 102     | ALLEN  | CSE    | Mr. Y | 040-53337     |
| 103     | JAMES  | CSE    | Mr. Y | 040-53337     |
| 104     | MILLER | CSE    | Mr. Y | 040-53337     |
| 105     | WARNER | CSE    | Mr. Y | 040-53337     |

### **Deletion problem:**

In our Student Details table, two different information's are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is called as Deletion problem.

**HOW NORMALIZATION WILL SOLVE ALL THESE PROBLEMS:**

**STUDENT DETAILS**

| Roll no | Name   | Branch | Hod   | Office Number |
|---------|--------|--------|-------|---------------|
| 101     | SAI    | CSE    | Mr. Y | 040-53337     |
| 102     | ALLEN  | CSE    | Mr. Y | 040-53337     |
| 103     | JAMES  | CSE    | Mr. Y | 040-53337     |
| 104     | MILLER | CSE    | Mr. Y | 040-53337     |
| 105     | WARNER | CSE    | Mr. Y | 040-53337     |

**NOTE: NOW WE NEED TO DECOMPOSING A STUDENT TABLE INTO TWO TABLES LIKE BELOW,**

**STUDENT DETAILS**

| Roll no | Name   | Branch (FK) |
|---------|--------|-------------|
| 101     | SAI    | CSE         |
| 102     | ALLEN  | CSE         |
| 103     | JAMES  | CSE         |
| 104     | MILLER | CSE         |
| 105     | WARNER | CSE         |

### BRANCH DETAILS

| Branch (PK) | Hod   | Office Number |
|-------------|-------|---------------|
| CSE         | Mr. Y | 040-53337     |

**NOTE: BY THE ABOVE EXAMPLE WE AVOID INSERTION, DELETION AND UPDATION PROBLEMS.**

**Types of Normal Forms:** Normalization can be achieved in multiple ways:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF
5. Fourth Normal Form
6. Fifth Normal form

### **First Normal Form (1NF):**

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. Each column should contain atomic value (atomic = single value).

Ex: column1                      column2

A                      X, Y

B                      W, X

C                      Y

D                      Z

**2. A COLUMN SHOULD CONTAIN VALUES THAT ARE SAME DATATYPE.**

**EX:**            **NAME**            **DOB**

**SAI            01-JAN-92**

**JONES            24-APR-84**

**18-DEC-85            MILLER**

**3. All the columns in a table should have unique names.**

**EX:**            **NAME**        **NAME**        **DOB**

**SAI    SAI    16-OCT-93**

**4. The order in which data is stored, does not matter.**

**EX:**            **ROLLNO            FIRST\_NAME    LAST\_NAME**

**1            SAI            KUMAR**

**2            JONES            ROY**

**4            MILLER            JOY**

**3            JAMES            WARTON**

**EX:**                            **STUDENT TABLE**

| <b>Roll no</b> | <b>Name</b>  | <b>Subject</b>      |
|----------------|--------------|---------------------|
| <b>101</b>     | <b>SAI</b>   | <b>JAVA, ORACLE</b> |
| <b>102</b>     | <b>JONES</b> | <b>PYTHON</b>       |
| <b>103</b>     | <b>ALLEN</b> | <b>C, C++</b>       |

The above table already satisfies 3 rules out of the 4 rules, as all our column names are unique, we have stored data in the order we wanted to and we have not inter-mixed different type of data in columns.

But out of the 3 different students in our table, 2 have opted for more than 1 subject. And we have stored the subject names in a single column. But as per the 1st Normal form each column must contain atomic value.

To avoid this problem, we have to break the values into atomic values. Here is our updated table and it now satisfies the First Normal Form.

#### < COMPOSITE PRIMARY KEY >

| Roll no | Name  | Subject |
|---------|-------|---------|
| 101     | SAI   | ORACLE  |
| 101     | SAI   | JAVA    |
| 102     | JONES | PYTHON  |
| 103     | ALLEN | C       |
| 103     | ALLEN | C++     |

**NOTE:** By doing so, although a few values are getting repeated but values for the SUBJECT column are now atomic for each record/row.

## Second Normal Form (2NF):

For a table to be in the Second Normal Form, it must satisfy two conditions:

1. The table should be in the First Normal Form.
2. There should be no Partial Dependency.

**WHAT IS DEPENDENCY:** IN A TABLE IF NON-KEY COLUMNS (NON-PRIMARY KEY) ARE DEPENDS ON KEY COLUMN (PRIMARY KEY) THEN IT IS CALLED AS FULLY DEPENDENCY / FUNCTIONAL DEPENDENCY.

(PK)

EX: A B C D

Here, "A" IS A KEY COLUMN → "B", "C", "D" ARE NON-KEY COLUMNS.

EX:

(PK)

STUDENT TABLE

| STUDENT_ID | Name   | Branch | ADDRESS |
|------------|--------|--------|---------|
| 101        | SAI    | CSE    | HYD     |
| 102        | SAI    | IT     | MUM     |
| 103        | JAMES  | CSE    | CHENNAI |
| 104        | MILLER | CSE    | HYD     |

**NOTE: A PRIMARY KEY COLUMN (STID) CAN BE USED TO FETCH DATA ANY COLUMN IN THE TABLE.**

**WHAT IS PARTIAL DEPENDENCY: IN A TABLE IF NON-KEY COLUMN DEPENDS ON PART OF THE KEY COLUMN, THEN IT IS CALLED AS PARTIAL DEPENDENCY**

**<PRIMARY KEY (A, B) / COMPOSITE PRIMARY KEY>**

**EX:            A     B     C     D**

**Here, "A AND B "IS A KEY COLUMNS → " C"," D" ARE NON-KEY COLUMNS. THEN "D" DEPENDS ON "B" BUT NOT "A" COLUMN.**

**EX: Let's create another table for Subject, which will have SUBJECT\_ID and SUBJECT\_NAME fields and SUBJECT\_ID will be the primary key.**

**<PRIMARY KEY>**

**SUBJECT TABLE**

| <b>SUBJECT_ID</b> | <b>SUBJECT_NAME</b> |
|-------------------|---------------------|
| <b>1</b>          | <b>ORACLE</b>       |
| <b>2</b>          | <b>JAVA</b>         |
| <b>3</b>          | <b>PYTHON</b>       |

**Now we have a student table with student information and another table Subject for storing subject information.**

**Let's create another table Score, to store the marks obtained by students in the respective subjects.**

**We will also be saving name of the teacher who teaches that subject along with marks.**



**(COMPOSITE PRIMARY KEY)      SCORE TABLE**

| STUDENT_ID | SUBJECT_ID | MARKS | TEACHER        |
|------------|------------|-------|----------------|
| 101        | 1          | 70    | ORACLE Teacher |
| 101        | 2          | 75    | JAVA Teacher   |
| 102        | 1          | 80    | OACLE Teacher  |
| 103        | 3          | 68    | PYTHON Teacher |

In the score table we are saving the STUDENT\_ID to know which student's marks are these and SUBJECT\_ID to know for which subject the marks are for.

Together STUDENT\_ID + SUBJECT\_ID forms composite primary key for this table, which can be the Primary key.

**NOTE:**

1. IN ABOVE SCORE TABLE," TEACHER COLUMN" IS ONLY DEPENDS ON SUBJECT\_ID BUT NOT ON STUDENT\_ID IS CALLED AS "PARTIAL DEPENDENCY".
2. IF THERE IS NO COMPOSITE PRIMARY KEY ON A TABLE THEN THERE IS NO PARTIAL DEPENDENCY.

**HOW TO REMOVE PARTIAL DEPENDENCY:** there are many different solutions to remove partial dependency.so our objective is to remove "teacher" column from score table and add to subject table. hence, the subject table will become

### **SUBJECT TABLE**

| <b>SUBJECT_ID</b> | <b>SUBJECT_NAME</b> | <b>TEACHER</b>        |
|-------------------|---------------------|-----------------------|
| <b>1</b>          | <b>ORACLE</b>       | <b>ORACLE Teacher</b> |
| <b>2</b>          | <b>JAVA</b>         | <b>JAVA Teacher</b>   |
| <b>3</b>          | <b>PYTHON</b>       | <b>PYTHON Teacher</b> |

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

### **<COMPOSITE PRIMARY KEY>**

| <b>STUDENT_ID</b> | <b>SUBJECT_ID</b> | <b>Marks</b> |
|-------------------|-------------------|--------------|
| <b>101</b>        | <b>1</b>          | <b>70</b>    |
| <b>101</b>        | <b>2</b>          | <b>75</b>    |
| <b>102</b>        | <b>1</b>          | <b>80</b>    |
| <b>103</b>        | <b>3</b>          | <b>68</b>    |

### **Third Normal Form (3NF):**

**For a table to be in the third normal form there is two conditions.**

- 1. It should be in the Second Normal form.**
- 2. And it should not have Transitive Dependency.**

**TRANSITIVE DEPENDENCY: IN TABLE IF NON-KEY COLUMN DEPENDS ON NON-KEY COLUMN, THEN IT IS CALLED AS TRANSITIVE DEPENDENCY.**

**(Composite Primary key)**

**EX: A B C D**

**Here, "A AND B "ARE KEY COLUMNS → " C"," D" ARE NON-KEY COLUMNS. THEN "D" DEPENDS ON "C" BUT NOT "A & B" COLUMNS.**

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

**<COMPOSITE PRIMARY KEY> SCORE TABLE**

| STUDENT_ID | SUBJECT_ID | MARKS | EXAM_NAME | TOTAL_MARKS |
|------------|------------|-------|-----------|-------------|
|            |            |       |           |             |
|            |            |       |           |             |
|            |            |       |           |             |

**with exam name and total marks added to our score table, it saves more data now. primary key for our score table is a composite key, which means it's made up of two attributes or columns → student +subject**

**our new column exam name depends on both student and subject. for example, a mechanical engineering student will have workshop exam but a computer science student won't. and for some subjects you have practical exams and for some you don't. so, we can say that exam name is dependent on both student id and subject id.**

**well, the column total marks depend on exam name as with exam type the total score changes. for example, practical is less marks while theory exams are having more marks.**

**but exam name is just another column in the score table. it is not a primary key and total marks depends on it.**

**this is 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:** again, the solution is very simple. take out the column's exam name and total marks from score table and put them in an exam table and use the exam\_id wherever required.

**Score Table: In 3rd Normal Form**

| STUDENT_ID | SUBJECT_ID | MARKS | EXAM_ID(FK) |
|------------|------------|-------|-------------|
|            |            |       |             |
|            |            |       |             |

**Exam table**

| EXAM_ID(PK) | EXAM_NAME   | TOTAL_MARKS |
|-------------|-------------|-------------|
| 1           | Workshop    | 200         |
| 2           | Mains       | 70          |
| 3           | Practical's | 30          |

**SUPER KEY & CANDIDATE KEY:**

**SUPER KEY:** A COLUMN (OR) COMBNATION OF COLUMNS WHICH ARE UNIQUELY IDENTIFYING A ROW IN A TABLE IS CALLED AS SUPER KEY.

**CANDIDATE KEY:** A MINIMAL SUPER KEY WHICH IS UNIQUELY IDENTIFYING A ROW IN A TABLE IS CALLED AS CANDIDATE KEY.

**(OR)**

**A SUPER KEY WHICH IS SUBSET OF ANOTHER SUPER KEY, BUT THE COMBINATION OF SUPER KEYS IS NOT A CANDIDATE KEY.**

**IN DB DESIGN ONLY DB DESIGNER USES SUPER KEY AND CANDIDATE KEY. THAT MEAN FIRST DESIGNERS SELECT SUPER KEYS AND THEN ONLY THEY ARE SELETING CANDIDATE KEYS FROM THOSE SUPER KEYS.**

**EX: STUDENT TABLE**

| STUDENT_ID | NAME  | BRANCH | MAILID   | REG_NUMBER |
|------------|-------|--------|--|------------|
| 101        | SAI   | CSE    | <a href="mailto:sai@gamil.com">sai@gamil.com</a>   | CS-10021   |
| 102        | JONES | CSE    | <a href="mailto:joy@gmail.com">joy@gmail.com</a>   | CS-10022   |
| 103        | ALLEN | IT     | <a href="mailto:all@ymail.com">all@ymail.com</a>   | IT-20021   |
| 104        | SAI   | EEE    | <a href="mailto:mi@hotmail.com">mi@hotmail.com</a> | EE-30021   |

**EX. OF SUPER KEYS:**

stdid | stdid + mailid |  
 mailid | mailid + reg\_number | stdid + mailid + reg\_number  
 reg\_number | reg\_number + stdid |

**EX. ON CANDIDATE KEYS:**

stdid

mailid

reg\_number

**Boyce- Codd Normal Form (BCNF):**

**For a table to satisfy the Boyce- Codd Normal Form, it should satisfy the following two conditions:**

- 1. It should be in the Third Normal Form.**
- 2. And, for any dependency  $A \rightarrow B$ , A should be a super key.**

**EX:**

**(COMPOSITE PRIMARY KEY) College Enrollment Table**

| <b>STUDENT_ID</b> | <b>SUBJECT(B)</b> | <b>PROFESSOR(A)</b> |
|-------------------|-------------------|---------------------|
| <b>101</b>        | <b>Java</b>       | <b>P. Java</b>      |
| <b>101</b>        | <b>C++</b>        | <b>P. Cpp</b>       |
| <b>102</b>        | <b>Java</b>       | <b>P. Java2</b>     |
| <b>103</b>        | <b>Oracle</b>     | <b>P. Oracle</b>    |
| <b>104</b>        | <b>Java</b>       | <b>P. Java</b>      |

in the table above, student id, subject form primary key, which means subject column, is a prime attribute. but there is one more dependency, professor → 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.

below we have the structure for both the tables.

**Student Table**

| <b>STUDENT_ID</b> | <b>PROFESSOR_ID</b> |
|-------------------|---------------------|
| <b>101</b>        | <b>1</b>            |
| <b>101</b>        | <b>2</b>            |

**Professor Table  
(COMPOSITE PRIMARY KEY)**

| <b>PROFESSOR_ID</b> | <b>professor</b> | <b>Subject</b> |
|---------------------|------------------|----------------|
| <b>1</b>            | <b>P. Java</b>   | <b>Java</b>    |
| <b>2</b>            | <b>P. Cpp</b>    | <b>C++</b>     |

**And now, this relation satisfies Boyce-Codd Normal Form.**

**Fourth Normal Form (4NF):**

**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. A table does not contain more than one independent multi-valued attribute / Multi Valued Dependency.**

**Multi valued Dependency:** In a table one column same value match with multiple values of another column is called as multi valued dependency.

**NOTE:** Generally, when a table having more than one independent multi valued attributes then the table having more duplicate data for reducing this duplicate data then DB DESIGNERS use 4NF process otherwise no need (it is optional).

**Ex: COLLEGE ENROLLMENT TABLE (5NF)**

| <b>STUDENT_ID</b> | <b>COURSE</b> | <b>HOBBY</b>   |
|-------------------|---------------|----------------|
| <b>1</b>          | <b>ORACLE</b> | <b>Cricket</b> |
| <b>1</b>          | <b>JAVA</b>   | <b>Reading</b> |
| <b>1</b>          | <b>C#</b>     | <b>Hockey</b>  |

in the table above, there is no relationship between the columns course and hobby. they are independent of each other.so there is multi-value dependency, which leads to un-necessary repetition of data.

identify independent multi valued attributes and those attributes move into separate tables these tables are called as 4nf tables. these tables do not contain more than one independent multi valued attribute (column).

**Hobbies Table (4NF)**

| <b>STUDENT_ID</b> | <b>Hobby</b>   |
|-------------------|----------------|
| <b>1</b>          | <b>Cricket</b> |
| <b>1</b>          | <b>Reading</b> |
| <b>1</b>          | <b>Hockey</b>  |

**Course Opted Table (4NF)**

| <b>STUDENT_ID</b> | <b>Course</b> |
|-------------------|---------------|
| <b>1</b>          | <b>ORACLE</b> |
| <b>1</b>          | <b>JAVA</b>   |
| <b>1</b>          | <b>C#</b>     |



### **Fifth Normal Form (5NF):**

**If a table having multi valued attributes and also that table cannot decomposed into multiple tables is called as fifth normal form.**

**Generally, in 4NF resource table some attributes are not logically related where as in 5NF resource table all attributes are related to one to another.**

**Fifth normal form is also called as project joined normal form because if possible decomposing table into number of tables and also whenever we are joining those tables then the result records must be available in resource table.**

