Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form or BCNF is an extension to the third normal form, and is also known as 3.5 Normal Form.

Follow the video above for complete explanation of BCNF. Or, if you want, you can even skip the video and jump to the section below for the complete tutorial.

In our last tutorial, we learned about the third normal form and we also learned how to remove **transitive dependency** from a table, we suggest you to follow the last tutorial before this one.

Rules for 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 → 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 → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

Time for an Example

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

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

As you can see, we have also added some sample data to the table.

In the table above:

- 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.

This table satisfies the **1st Normal form** because all the values are atomic, column names are unique and all the values stored in a particular column are of same domain.

This table also satisfies the **2nd Normal Form** as their is no **Partial Dependency**.

And, there is no **Transitive Dependency**, hence the table also satisfies the **3rd Normal Form**.

But this table is not in **Boyce-Codd Normal Form**.

Why this table is not in BCNF?

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

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++
and so on...		

And now, this relation satisfy Boyce-Codd Normal Form. In the next tutorial we will learn about the **Fourth Normal Form**.

A more Generic Explanation

In the picture below, we have tried to explain BCNF in terms of relations.

Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC.**

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.

$$\begin{tikzcd}
 & R(A, B, C, D) \arrow[dl, dash] \arrow[dr, dash] & \\
R1\ (A, D, C) & & R2(D, B)
\end{tikzcd}$$

Breaking, table into two tables, one with A, D and C while the other with D and B.