Third Normal Form (3NF)

Remove transitive dependencies.

- **Transitive dependence - two separate entities exist within one table.**
- **Any transitive dependencies are moved into a smaller (subset) table.**

3NF further improves data integrity.

- **Prevents update, insert, and delete anomalies.**

Example

item	color
T-shirt	red
T-shirt	blue
polo	red
polo	yellow
sweatshirt	blue
sweatshirt	black

item	price	tax
T-shirt	12.00	0.60
polo	12.00	0.60
sweatshirt	25.00	1.25

Tables are not in third normal form because:

• Tax depends on price, not item

Example

item	color
T-shirt	red
T-shirt	blue
polo	red
polo	yellow
sweatshirt	blue
sweatshirt	black

item	price
T-shirt	12.00
polo	12.00
sweatshirt	25.00

price	tax
12.00	0.60
25.00	1.25

Tables are now in third normal form.

Example

Name	Assignment 1	Assignment 2
Jeff **Smith**	Article Summary	Poetry Analysis
Nancy **Jones**	Article Summary	Reaction Paper
Jane **Scott**	Article Summary	Poetry Analysis

Table is not in first normal form because:

- **Assignment field repeating**
- **First and last name in one field**
- **No (guaranteed unique) primary key field**

First Name	Last Name	Assignment
Jeff	Smith	Article Summary
Jeff	Smith	Poetry Analysis
Nancy	Jones	Article Summary
Nancy	Jones	Reaction Paper
Jane	Scott	Article Summary
Jane	Scott	Poetry Analysis

Table is now in first normal form.

Student ID	First Name	Last Name
1	Jeff	Smith
2	Nancy	Jones
3	Jane	Scott

Assignment ID	Student ID	Description
1	1	Article Summary
2	1	Poetry Analysis
1	2	Article Summary
3	2	Reaction Paper
1	3	Article Summary
2	3	Poetry Analysis

Tables are in second normal form

Assignment ID	Description
1	Article Summary
2	Poetry Analysis
3	Reaction Paper

Student ID	First Name	Last Name
1	Jeff	Smith
2	Nancy	Jones
3	Jane	Scott

Assignment ID	Student ID
1	1
1	2
1	3
2	1
2	3
3	2

Tables are in third normal form.

Example

Employee (unnormalized)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C, Perl, Java
2	Barbara Jones	224	IT	Linux, Mac
3	Jake Rivera	201	R&D	DB2, Oracle, Java

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Kevin Jacobs	201	R&D	C
1	Kevin Jacobs	201	R&D	Perl
1	Kevin Jacobs	201	R&D	Java
2	Barbara Jones	224	IT	Linux
2	Barbara Jones	224	IT	Mac
3	Jake Rivera	201	R&D	DB2
3	Jake Rivera	201	R&D	Oracle
3	Jake Rivera	201	R&D	Java

Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D

Skills (2NF)

emp_no	skills
1	C
1	Perl
1	Java
2	Linux
2	Mac
3	DB2
3	Oracle
3	Java

Dept_no and **dept_name** are **functionally dependent** on **emp_no** however, department can be considered a separate entity.

Employee (2NF)

emp_no	name	dept_no	dept_name
1	Kevin Jacobs	201	R&D
2	Barbara Jones	224	IT
3	Jake Rivera	201	R&D

Employee (3NF)

emp_no	name	dept_no
1	Kevin Jacobs	201
2	Barbara Jones	224
3	Jake Rivera	201

Department (3NF)

dept_no	dept_name
201	R&D
224	IT

BCNF Definition

Boyce-Codd Normal Form is a stricter version of 3NF that applies to relations where there may **be overlapping candidate keys**.
A relation is said to be in ***Boyce-Codd normal form*** if it is in 3NF and every non-trivial **FD** given for this relation has a **candidate key as its determinant**. That is, for every **X → Y, X** is a candidate key.

Example

Consider a ***Guest Lecture relation*** for a college as shown in *Table* below. Assume each teacher teaches only one subject.
Candidate Keys: {Subject, Lecture_Day}, {Lecture_Day, Teacher}

Guest Lecture relation

Subject	Lecture_Day	Teacher
Graphics	Monday	Dr. Arindham Singh
Databases	Monday	Dr. Emily Jose
Java	Wednesday	Dr. Prasad
Graphics	Tuesday	Dr. Arindham Singh
Java	Thursday	Dr. George White

In the above relation, there are no non-atomic values hence, it is in **1NF**.
All the attributes are part of candidate keys hence it is in **2NF and 3NF**.
An FD, **Teacher → Subject** exists for the above relation. However, **Teacher alone is not a candidate key**; therefore, the above relation is **not a BCNF**. To convert it into a BCNF relation, **we decompose it into relations Subject area experts** and **Lecture timetable** as shown below.

Subject	Teacher
Graphics	Dr. Arindham Singh
Databases	Dr. Emily Jose
Java	Dr. Prasad
Java	Dr. George White

(a) Subject area experts' relation

Subject	Lecture_Day	Teacher
Graphics	Monday	Dr. Arindham Singh
Databases	Monday	Dr. Emily Jose
Java	Wednesday	Dr. Prasad
Graphics	Tuesday	Dr. Arindham Singh
Java	Thursday	Dr. George White

(b) Lecture timetable

Teacher is now a candidate keys in *table above (a) Subject area experts' relation*.

A relation is in Boyce-Codd Normal Form (BCNF) if every determinant is a candidate key.

Also Known As: BCNF

Examples:

Conisder a database table that stores employee information and has the attributes employee_id, first_name, last_name, title. In this table, the field employee_id determines first_name and last_name. Similarly, the tuple (first_name, last_name) determines employee_id.

The Boyce-Codd normal form (abbreviated BCNF) is a "key heavy" derivation of the third normal form. The simplest way I can think to define it is if the key uniquely identifies a row, but the key includes more columns than are actually required to uniquely identify a row, then a table is in BCNF. For example:

```
CREATE TABLE t_employees1 (
  employee_id    INT        IDENTITY
, last_name      VARCHAR(25) NOT NULL
, first_name     VARCHAR(25) NOT NULL
  CONSTRAINT XPKt_employees1
    PRIMARY KEY (employee_id, last_name, first_name)
--  other columns as needed
  )
```

This example of the **t_employees1** table is in BCNF. To coax it into 3NF (third normal form), I would use:

```
CREATE TABLE t_employees2 (
  Employee_id     INT        IDENTITY
  CONSTRAINT XPKt_employees2
    PRIMARY KEY (employee_id)
, last_name       VARCHAR(25) NOT NULL
, first_name      VARCHAR(25) NOT NULL
--  other columns as needed
  )
```

Candidate key

A candidate key is a combination of attributes that can be uniquely used to identify a database record without any extraneous data. Each table may have one or more candidate keys. One of these candidate keys is selected as the table primary key.

There are three fundamental demands on the candidate keys that we should never deviate from, if it is to become the subject for a primary key:

1. The candidate key must be unique within its domain (the entity it represents, and beyond, if you also intend to access external entities, with is clearly illustrated in this article).
2. The candidate key cannot hold **NULLs** (NULL is not zero. Zero is a number. NULL is 'unknown value').
3. The candidate key should never change. It must hold the same value for a given occurrence of an entity for the lifetime of that entity.

In 1, we say that the main purpose of the candidate key is to help us to identify one single row in a table, regardless of whether there exist billions of rows. This sets high demands on the flexibility in terms of delivering uniqueness.

In 2, we say that a candidate key always, without exception, must hold a value. Without it, we break post 1.

In 3, we say that no matter what happens, inside our business or outside of it, we are not allowed to change the value of the candidate key. Adding an attribute (expanding) the candidate key, would pose a change. Some say that you should be able to change the primary key, and do a 'cascade update' to all underlying tables, ref above.