Normalization in DBMS: 1NF, 2NF, 3NF, BCNF & 4NF with Examples

Normalization is a technique of organizing the data in the database. It is a systematic approach which is used to **remove or reduce data redundancy** in the tables and remove the **insert, update, and delete anomalies**. It mainly divides the larger table into smaller tables and links them using a relationship to increase the clarity of data. Normalization was introduced by IBM researcher **E.F. Codd** in **1970s**.

Anomalies in DBMS

Following are the three types of anomalies that occur when the database is not normalized:

1. Insertion Anomaly
2. Updation Anomaly
3. Deletion Anomaly

To understand these anomalies, let’s take an example:
Below table University consists of seven attributes: **Sid, Sname, Cid, Cname, Fid, Fname,** and **Salary.** And the Sid acts as a key attribute or a primary key in the relation.
Table: University

Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
2	Shyam	C2	Java	F2	Boby	28000
3	Ankit	C1	DBMS	F1	Sachin	30000
4	saurabh	C1	DBMS	F1	Sachin	30000

1. Insertion Anomaly
Suppose a new faculty joins the University, and the Database Administrator inserts the faculty data into the above table. But he is not able to insert because Sid is a primary key, and can’t be NULL. So this type of anomaly is known as an insertion anomaly.
Table: University

Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
2	Shyam	C2	Java	F2	Boby	28000
3	Ankit	C1	DBMS	F1	Sachin	30000
4	saurabh	C1	DBMS	F1	Sachin	30000
				F3	Arun	29000

Insertion Anomaly

2. Delete Anomaly

When the Database Administrator wants to delete the student details of **Sid=2** from the above table, then it will delete the faculty and course information too which cannot be recovered further.

Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
~~2~~	~~Shyam~~	Deletion anomaly				
3	Ankit	C1	DBMS	F1	Sachin	30000
4	Saurabh	C1	DBMS	F1	Sachin	30000

3. Update Anomaly

When the Database Administrator wants to change the salary of faculty F1 from 30000 to 40000 in above table University, then the database will update salary in more than one row due to data redundancy. So, this is an update anomaly in a table.

Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
2	Shyam	C2	Java	F2	Boby	28000
3	Ankit	C1	DBMS	F1	Sachin	30000
4	saurabh	C1	DBMS	F1	Sachin	30000

To remove all these anomalies, we need to normalize the data in the database.

Normal forms

Database Normalization is divided into the following Normal forms:

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (3.5NF/BCNF)

First Normal Form (1NF)

According to the E.F. Codd, a relation will be in 1NF, if each cell of a relation contains only an atomic value. This normal form states that an attribute of a relation cannot hold multiple values. It should hold only single-valued attributes. Values stored in an attribute should be of the same domain.

Example:

The following **student relation** is not in 1NF because the Subject attribute contains multiple values.

Student_id	Name	Subject
101	Akash	Computer Network, JAVA
102	Vikrant	Database Management System
103	Amrita	Software Engineering, Compiler Design

The below relation student is in 1NF:

Student_id	Name	Subjects
101	Akshay	Computer Network
101	Akshay	JAVA
102	Aman	Database Management System
103	Anjali	Software Engineering
103	Anjali	Compiler Design

Relation is in 1NF

Second Normal Form (2NF)

According to the E.F. Codd, a relation is in **2NF,** if it satisfies the following conditions:

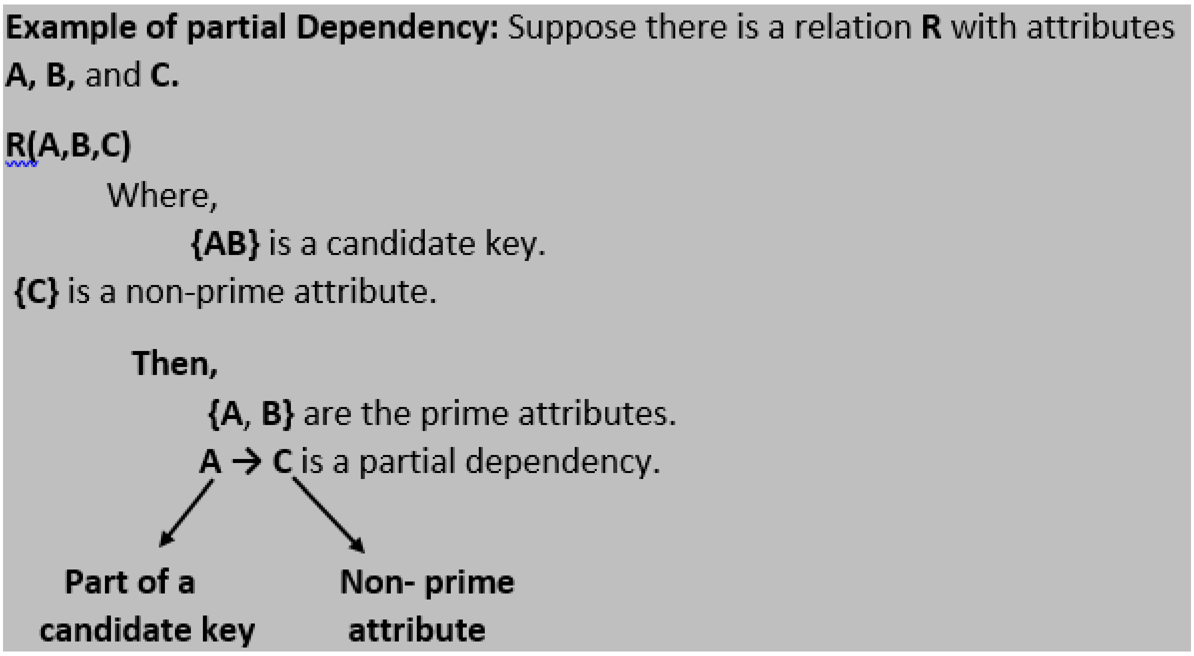
- A relation must be in 1NF.
- And the candidate key in a relation should determine all non-prime attributes or no partial dependency should exist in the relation.

Prime attributes: The attributes which are used to form a candidate key are called prime attributes.

Non-Prime attributes: The attributes which do not form a candidate key are called non-prime attributes.

Partial Dependency: If a non-prime attribute can be determined by the part of the candidate key in a relation, it is known as a partial dependency. Or we can say that, if L.H.S is the proper subset of a candidate key and R.H.S is the non-prime attribute, then it shows a partial dependency.

Example of partial Dependency: Suppose there is a relation **R** with attributes **A, B,** and **C.**



Example of partial Dependency: Suppose there is a relation **R** with attributes **A, B,** and **C.**

R(A,B,C)

Where,

{AB} is a candidate key.

{C} is a non-prime attribute.

Then,

{A, B} are the prime attributes.

A → C is a partial dependency.

Example of Second normal form:

Example: Suppose a training institute wants to store the data of student and the **programming_languages** they learn. Since a student can learn more than one programming_language, the relation can have multiple rows for a same student. Following relation shows the data of the students:

student_id	programming_langauges	marks	student_age
101	Computer Network	88	20
101	JAVA	88	20
102	Database Management System	97	20
103	JAVA	96	21
103	Compiler Design	96	21

Prime attribute – stud_id, programming language
Non prime – marks,age

Student_id ---→age
Programming language --→
Stud_id+programming language---→ marks
Candidate Keys: {student_id, programming_language}
Non-prime attribute: student_age
The above relation is in 1 NF because each attribute contains atomic values.
However, it is not in 2NF because a non-prime attribute **student_age is** dependent on **student_id,** which is a proper subset of a candidate key.
This violates the rule for second normal form as a rule says “no non-prime attribute should be dependent on the part of a candidate key of the relation”.
To make the relation in 2NF, we can break it in two tables like:
Student_details table:

student_id	student_age
101	20
101	20
102	20
103	21
103	21

student_programminglangauge table:

student_id	programming_langauge
101	Computer Network
101	JAVA
102	Database Management System
103	Software Engineering
103	Compiler Design

Now, both the tables follow 2NF.

student_id	student_age
101	20
102	20
103	21

Example 2:

Product id	custid	custname	address	product name	price	order date
1	1	xxxx	aaaa	lays	30	1 Nov
2	2	yyyyy	zzzzz	biscuits	40	2 Nov
1	2	yyyyy	zzzzz	lays	30	1 Nov
1	3	zzzz	rrrrr	lays	30	3 nov

Product---(Productid,product name,price)
Customer(custid,cname,address)
Prod cust (productid,custid,orderdate)

Tells which product which customer has bought

One customer can buy many products
And one product can be bought by many customer
Pid+custid
Non prime columns ------
Pime column
Pid ----→productname,,price
Custid -→custname is functionally depenedent on custid
custaddress
Pid+custid---→ orderdate, pkid

So in this table we have partial functional dependency so this is not in 2 NF

Third Normal Form (3NF)

According to the E.F. Codd, a relation is in **third normal form (3NF)** if it satisfies the following conditions:

- A relation must be in second normal form (2NF).
- And there should be no transitive functional dependency exists for non-prime attributes in a relation.

Third Normal Form is used to achieve data integrity and reduce the duplication of data.

A relation is in 3NF if and only if any one of the following condition will satisfy for each non-trivial functional dependency **X→ Y:**

1. X is a super key or candidate key
2. And, Y is a prime attribute, i.e., Y is a part of candidate key.

Transitive Dependency: If **X → Y and Y→ Z** are two functional dependencies, **X → Z** is called as a transitive functional dependency.

Example of 3NF:

Suppose a school wants to store the address of each student, they create a table named **student_details** that looks like:

Rollno	**State**	**City**
1	Punjab	Chandigarh
2	Haryana	Ambala
3	Punjab	Chandigarh
4	Haryana	Ambala
5	Uttar Pradesh	Ghaziabad

Candidate Key: {Rollno}
Prime attribute: Rollno
Non-prime attribute: {State, City}
The above relation is not in third normal form, because as a rule says, there should be no transitive functional dependency in the relation.
Here, **City** (a non-prime attribute) depends on **State** (a non-prime attribute), and **State** depends on **Rollno**. The non-prime attributes (State, City) are transitively dependent on the candidate key(Rollno). Thus, it violates the rule of third normal form.
To covert the relation in 3NF, you have to decompose the relation as:
Table: Student_state

Rollno	City
1	Chandigarh
2	Ambala
3	Chandigarh
4	Ambala
5	Ghaziabad

Table:Student_city

State	City
Punjab	Chandigarh
Haryana	Ambala
Uttar Pradesh	Ghaziabad

Now, both the tables follow the third normal form (3NF).

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form (BCNF) is the advance version of the third normal form (3NF) that's why it is also known as a **3.5NF**
According to the E.F. Codd, a relation is in **Boyce-Codd normal form (3NF)** if it satisfies the following conditions:

- A relation is in 3NF.

- And, for every functional dependency, **X → Y,** L.H.S of the functional dependency (X) be the super key of the table.

Example of BCNF:

Suppose there is a college where one faculty teach in more than one department. They create a table like:

F_id	F_address	Course_id	Course_name
101	Delhi	C1	MCA
101	Delhi	C2	MBA
102	Noida	C1	MCA
102	Noida	C2	MBA

In the above relation functional dependencies are as follows:

F_id → F_address

Course_id → Course_name

Candidate key: {Fid, Course_id}

Above relation is not in BCNF as neither **F_id** nor **Course_id** alone are keys**.**

To make the relation in BCNF, we can break the table into three parts like this:

Facult_address

F_id	F_address
101	Delhi
102	Noida

Course

Course_id	Course_name
C1	MCA
C2	MBA

Faculty_Course

F_id	Course_id
101	C1

101	C2
102	C1
102	C2

Fourth Normal Form

According to the E.F. Codd, a relation is in **fourth normal form (4NF)** if it satisfies the following conditions:

- A relation is in BCNF.
- And, there is no multivalued dependency exists in the relation.

Multivalued dependency: For a dependency **X → Y,** if for a single value of X, multiple values of Y exists, then the relation may have a multi-valued dependency. It is represented by the double arrow sign **(→→).**
A relation with multivalued dependencies violates the fourth normal form (4NF), because it creates unnecessary redundancy of data.
Example:
The relation student consists of three attributes: student_id, Name, and Course.

student_id	Name	Course
101	Ankit	Python
102	Kartikey	Java
103	Krishna	R programming
101	Ankit	JAVA
105	Akash	PHP

In the above relation, **Name and Course** are two independent attributes and both are dependent on student_id.
In this case, these two attributes can be called as multivalued dependent on student_id. Following are the representation of these dependencies:
Student_id →→ Name
Student_id →→ Course
So, to make the above relation into the fourth normal form (4NF), decompose it into two tables:
Student_name

student_id	Name
101	Ankit
102	Kartikey
103	Krishna
105	Akash

Student_course

student_id	Course
101	Python
102	Java
103	R programming
101	JAVA
105	PHP

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing

21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey