NORMALIZATION

While designing a database out of an entity–relationship model, the main problem existing in that "raw" database is redundancy. Redundancy is storing the same data item in more one place. A redundancy creates several problems like the following:

- ***Extra storage space:*** storing the same data in many places takes large amount of disk space.
- ***Insertion Anomaly:*** Entering same data more than once during data insertion.
- ***Deletion Anomaly:*** Deleting data from more than one place during deletion.
- ***Modification Anomaly:*** Modifying data in more than one place.
- **Anomalies** may occur in the database if insertion, deletion, modification etc are no done properly. It creates inconsistency and unreliability in the database.

To solve this problem, the "raw" database needs to be normalized. This is a step by step process of removing different kinds of redundancy and anomaly at each step. At each step a specific rule is followed to remove specific kind of impurity in order to give the database a slim and clean look.

Un-Normalized Form (UNF)

If a table contains non-atomic values at each row, it is said to be in UNF. An atomic value is something that can not be further decomposed. A non-atomic value, as the name suggests, can be further decomposed and simplified. Consider the following table:

Emp-Id	Emp-Name	Month	Sales	Bank-Id	Bank-Name
E01	AA	Jan	1000	B01	SBI
		Feb	1200		
		Mar	850		
E02	BB	Jan	2200	B02	UTI
		Feb	2500		
E03	CC	Jan	1700	B01	SBI
		Feb	1800		
		Mar	1850		
		Apr	1725		

In the sample table above, there are multiple occurrences of rows under each key Emp-Id. Although considered to be the primary key, Emp-Id cannot give us the unique identification facility for any single row. Further, each primary key points to a variable length record (3 for E01, 2 for E02 and 4 for E03).

First Normal Form (1NF)

A relation is said to be in 1NF if it contains no non-atomic values and each row can provide a unique combination of values. The above table in UNF can be processed to create the following table in 1NF.

Emp-Id	Emp-Name	Month	Sales	Bank-Id	Bank-Name
E01	AA	Jan	1000	B01	SBI
E01	AA	Feb	1200	B01	SBI
E01	AA	Mar	850	B01	SBI
E02	BB	Jan	2200	B02	UTI
E02	BB	Feb	2500	B02	UTI
E03	CC	Jan	1700	B01	SBI
E03	CC	Feb	1800	B01	SBI
E03	CC	Mar	1850	B01	SBI
E03	CC	Apr	1725	B01	SBI

As you can see now, each row contains unique combination of values. Unlike in UNF, this relation contains only atomic values, i.e. the rows can not be further decomposed, so the relation is now in 1NF.

So, A relation is in **first normal form** if it meets the definition of a relation:

- Each attribute (column) value must be a single value only.
- All values for a given attribute (column) must be of the same type.
- Each attribute (column) name must be unique.
- The order of attributes (columns) is insignificant
- No two tuples (rows) in a relation can be identical.
- The order of the tuples (rows) is insignificant.

Second Normal Form (2NF)

A relation is said to be in 2NF f if it is already in 1NF and each and every attribute fully depends on the primary key of the relation. Speaking inversely, if a table has some attributes which is not dependant on the primary key of that table, then it is not in 2NF.

Let us explain. Emp-Id is the primary key of the above relation. Emp-Name, Month, Sales and Bank-Name all depend upon Emp-Id. But the attribute Bank-Name depends on Bank-Id, which is not the primary key of the table. So the table is in 1NF, but not in 2NF. If this position can be removed into another related relation, it would come to 2NF.

Emp-Id	Emp-Name	Month	Sales	Bank-Id
E01	AA	JAN	1000	B01
E01	AA	FEB	1200	B01
E01	AA	MAR	850	B01
E02	BB	JAN	2200	B02
E02	BB	FEB	2500	B02
E03	CC	JAN	1700	B01
E03	CC	FEB	1800	B01
E03	CC	MAR	1850	B01
E03	CC	APR	1726	B01

Bank-Id	Bank-Name
B01	SBI
B02	UTI

After removing the portion into another relation we store lesser amount of data in two relations without any loss information. There is also a significant reduction in redundancy.

Third Normal Form (3NF)

A relation is said to be in 3NF, if it is already in 2NF and there exists no transitive dependency in that relation. Speaking inversely, if a table contains transitive dependency, then it is not in 3NF, and the table must be split to bring it into 3NF.

What is a transitive dependency? Within a relation if we see

A → B [B depends on A] And B → C [C depends on B]

Then we may derive

A → C [C depends on A]

Such derived dependencies hold well in most of the situations. For example if we have

Roll → Marks And Marks → Grade

Then we may safely derive

Roll → Grade.

This third dependency was not originally specified but we have derived it. The derived dependency is called a transitive dependency when such dependency becomes improbable.

Third Normal Form (3NF)

For example we have been given

Roll → City

And

City → STDCode

If we try to derive Roll → STDCode it becomes a transitive dependency, because obviously the STDCode of a city cannot depend on the roll number issued by a school or college. In such a case the relation should be broken into two, each containing one of these two dependencies:

Roll → City

And

City → STD code

For Example:

A bank uses the following relation:

Vendor(ID, Name, Account_No, Bank_Code_No, Bank)

The attribute ID is the identification key. All attributes are single valued (1NF). The table is also in 2NF.

The following dependencies exist:

1. Name, Account_No, Bank_Code_No are functionally dependent on ID

(ID --> Name, Account_No, Bank_Code_No)

2. Bank is functionally dependent on Bank_Code_No

(Bank_Code_No --> Bank)

Vendor

ID	Name	Account_No	Bank_Code_No	Bank

Startsituation

Result after normalisation

Vendor

ID	Name	Account_No	Bank_Code_No

Bank

Bank_Code_No	Bank

The table in this example is in 1NF and in 2NF. But there is a transitive dependency between Bank_Code_No and Bank, because Bank_Code_No is not the primary key of this relation. To get to the third normal form (3NF), we have to put the bank name in a separate table together with the clearing number to identify it.

Boyce-Code Normal Form (BCNF)

A relationship is said to be in BCNF if it is already in 3NF and the left hand side of every dependency is a candidate key. A relation which is in 3NF is almost always in BCNF. These could be same situation when a 3NF relation may not be in BCNF the following conditions are found true.

- **A relation is in BCNF if every determinant is a candidate key.**
- **Recall that not all determinants are keys.**
- **Those determinants that are keys we initially call *candidate keys*.**
- **Eventually, we select a single candidate key to be *the key* for the relation.**

Consider the following example:

- **Funds consist of one or more Investment Types.**
- **Funds are managed by one or more Managers**
- **Investment Types can have one more Managers**
- **Managers only manage one type of investment.**

Relation:

FUNDS (FundID, InvestmentType, Manager)

FundID	InvestmentType	Manager
99	Common Stock	Smith
99	Municipal Bonds	Jones
33	Common Stock	Green
22	Growth Stocks	Brown
11	Common Stock	Smith

FD1: FundID, InvestmentType → Manager

FD2: FundID, Manager → InvestmentType

FD3: Manager → InvestmentType

- **In this case, the combination FundID and InvestmentType form a *candidate key* because we can use FundID, InvestmentType to uniquely identify a tuple in the relation.**
- **Similarly, the combination FundID and Manager also form a *candidate key* because we can use FundID, Manager to uniquely identify a tuple.**

- **Manager by itself is not a candidate key because we cannot use Manager alone to uniquely identify a tuple in the relation.**
- **Is this relation FUNDS(FundID, InvestmentType, Manager) in 1NF, 2NF or 3NF ?**
  Given we pick FundID, InvestmentType as the *Primary Key:*
  1NF for sure.
  2NF because all of the non-key attributes (Manager) is dependant on all of the key.
  3NF because there are no transitive dependencies.
- **However consider what happens if we delete the tuple with FundID 22. We loose the fact that Brown manages the InvestmentType "Growth Stocks."**

Therefore, while FUNDS relation is in 1NF, 2NF and 3NF, it is in BCNF because not all determinants (Manager in FD3) are candidate keys.
The following are steps to normalize a relation into BCNF:

- **I. List all of the determinants.**
- **II. See if each determinant can act as a key (candidate keys).**
- **III. For any determinant that is *not* a candidate key, create a new relation from the functional dependency. Retain the determinant in the original relation.**

For our example: FUNDS (FundID, InvestmentType, Manager)

1. **The determinants are: FundID, InvestmentType, FundID, Manager**
2. **Which determinants can act as keys ?**
   FundID, InvestmentType *YES*
   FundID, Manager *YES*
   Manager *NO*
1. **Create a new relation from the functional dependency:**
   MANAGERS(<u>Manager</u>, InvestmentType)
   FUND_MANAGERS(<u>FundID</u>, <u>Manager</u>)

Decomposition for BCNF

Manager → InvestmentType

violates BCNF [since Manager is not a candidate key].
If X->Y violates BCNF then divide R into R1(X, Y) and R2(R-Y).

In this last step, we have retained the determinant "Manager" in the original relation MANAGERS.

Each of the new relations should be checked to ensure they meet the definitions of 1NF, 2NF, 3NF and BCNF

For example: Consider a relation R with attributes (student, subject, teacher).

F: { (student, Teacher) -> subject
(student, subject) -> Teacher
Teacher -> subject}

Student	Teacher	Subject
Jhansi	P.Naresh	Database
jhansi	K.Das	C
subbu	P.Naresh	Database
subbu	R.Prasad	C

Candidate keys are
(student, teacher) and (student, subject).
The above relation is in 3NF [since there is no transitive dependency].

- A relation R is in BCNF if for every non-trivial FD X->Y, X must be a key.
- The above relation is not in BCNF, because in the FD (teacher->subject), teacher is not a key.
- This relation suffers with anomalies –
  For example, if we try to delete the student Subbu, we will lose the information that R. Prasad teaches C. These difficulties are caused by the fact the teacher is determinant but not a candidate key.

For example: Consider a relation R with attributes (student, subject, teacher).

Decomposition for BCNF

Teacher-> subject violates BCNF [since teacher is not a candidate key].

If X->Y violates BCNF then divide R into R1(X, Y) and R2(R-Y).

So R is divided into two relations R1(Teacher, subject) and R2(student, Teacher).

R1

Teacher	Subject
P.Naresh	database
K.DAS	C
R.Prasad	C

R2

Student	Teacher
Jhansi	P.Naresh
Jhansi	K.Das
Subbu	P.Naresh
Subbu	R.Prasad

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.**

R(A, B, C, D)

R1 (A, D, C) R2(D, B)

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

Fourth Normal Form (4NF)

When attributes in a relation have multi-valued dependency, further Normalization to 4NF and 5NF are required. Let us first find out what multi-valued dependency is.

"A multi-valued dependency is a typical kind of dependency in which each and every attribute within a relation depends upon the other, yet none of them is a unique primary key."

We will illustrate this with an example. Consider a vendor supplying many items to many projects in an organization. The following are the assumptions:

- **A vendor is capable of supplying many items.**
- **A project uses many items.**
- **A vendor supplies to many projects.**
- **An item may be supplied by many vendors.**

A multi valued dependency exists here because all the attributes depend upon the other and yet none of them is a primary key having unique value.

Vendor Code	Item Code	Project No.
V1	I1	P1
V1	I2	P1
V1	I1	P3
V1	I2	P3
V2	I2	P1
V2	I3	P1
V3	I1	P2
V3	I1	P3

The given relation has a number of problems. For example:

- If vendor V1 has to supply to project P2, but the item is not yet decided, then a row with a blank for item code has to be introduced.
- The information about item I1 is stored twice for vendor V3.

Observe that the relation given is in 3NF and also in BCNF. It still has the problem mentioned above. The problem is reduced by expressing this relation as two relations in the Fourth Normal Form (4NF).

A relation is in 4NF if it has no more than one independent multi valued dependency or one independent multi valued dependency with a functional dependency.

The table can be expressed as the two 4NF relations given as following. The fact that vendors are capable of supplying certain items and that they are assigned to supply for some projects in independently specified in the 4NF relation.

Vendor Code	Item Code
V1	I1
V1	I2
V2	I2
V2	I3
V3	I1

Vendor Code	Project No.
V1	P1
V1	P3
V2	P1
V3	P2

Fifth Normal Form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

Example

In the table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Let us finally summarize the normalization steps we have discussed so far.

Input Relation	Transformation	Output Relation
All Relations	Eliminate variable length record. Remove multi-attribute lines in table.	1NF
1NF Relation	Remove dependency of non-key attributes on part of a multi-attribute key.	2NF
2NF	Remove dependency of non-key attributes on other non-key attributes.	3NF
3NF	Remove dependency of an attribute of a multi attribute key on an attribute of another (overlapping) multi-attribute key.	BCNF
BCNF	Remove more than one independent multi-valued dependency from relation by splitting relation.	4NF
4NF	Add one relation relating attributes with multi-valued dependency.	5NF