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:

1. Extra storage space: storing the same data in many places takes large amount of disk space.
2. Entering same data more than once during data insertion.
3. Deleting data from more than one place during deletion.
4. Modifying data in more than one place.
5. Anomalies may occur in the database if insertion, deletion, modification etc are not 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.

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. 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

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.

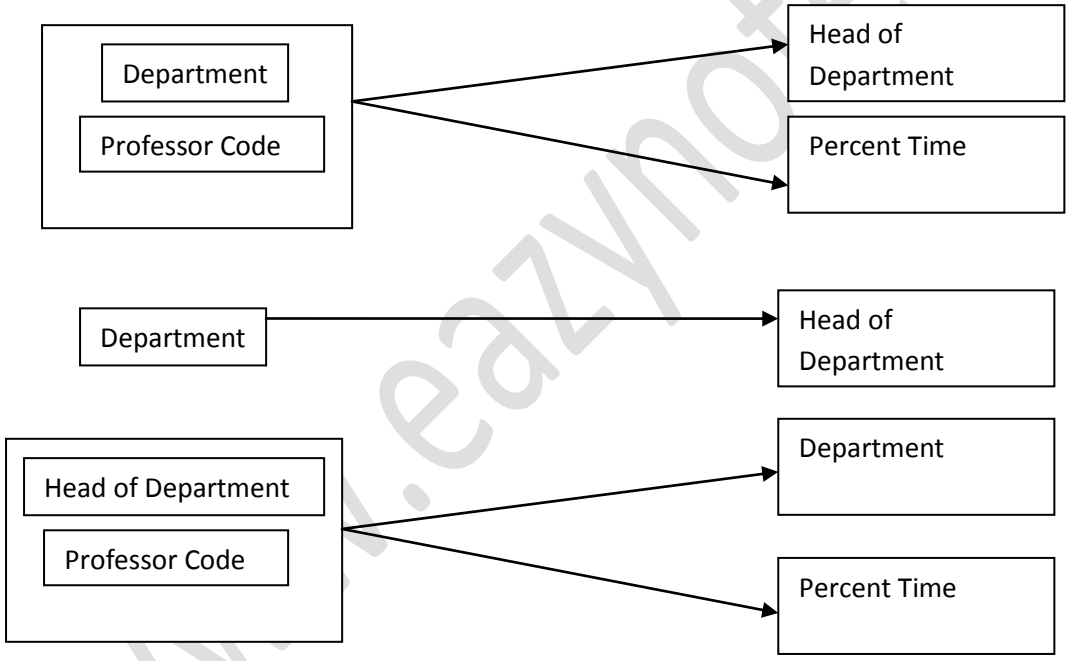
1. The candidate keys are composite.
2. There are more than one candidate keys in the relation.
3. There are some common attributes in the relation.

Professor Code	Department	Head of Dept.	Percent Time
P1	Physics	Ghosh	50
P1	Mathematics	Krishnan	50
P2	Chemistry	Rao	25
P2	Physics	Ghosh	75
P3	Mathematics	Krishnan	100

Consider, as an example, the above relation. It is assumed that:

1. A professor can work in more than one department
2. The percentage of the time he spends in each department is given.
3. Each department has only one Head of Department.

The relation diagram for the above relation is given as the following:



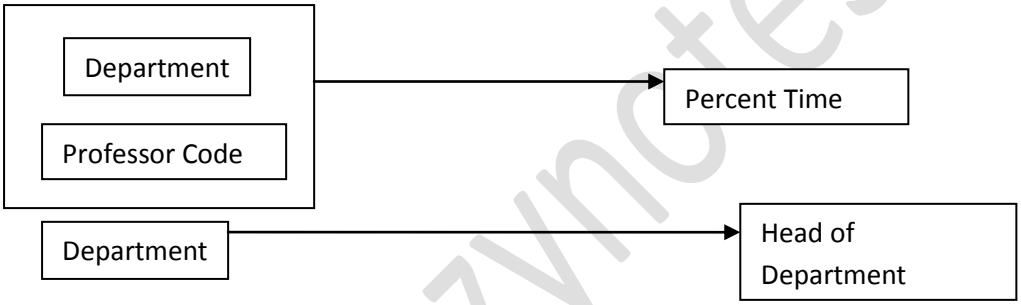
The given relation is in 3NF. Observe, however, that the names of Dept. and Head of Dept. are duplicated. Further, if Professor P2 resigns, rows 3 and 4 are deleted. We lose the information that Rao is the Head of Department of Chemistry.

The normalization of the relation is done by creating a new relation for Dept. and Head of Dept. and deleting Head of Dept. form the given relation. The normalized relations are shown in the following.

Professor Code	Department	Percent Time
P1	Physics	50
P1	Mathematics	50
P2	Chemistry	25
P2	Physics	75
P3	Mathematics	100

Department	Head of Dept.
Physics	Ghosh
Mathematics	Krishnan
Chemistry	Rao

See the dependency diagrams for these new relations.



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:

1. A vendor is capable of supplying many items.
2. A project uses many items.
3. A vendor supplies to many projects.
4. 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:

1. 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.
2. 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-Supply

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

Vendor-Project

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

Fifth Normal Form (5NF)

These relations still have a problem. While defining the 4NF we mentioned that all the attributes depend upon each other. While creating the two tables in the 4NF, although we have preserved the dependencies between Vendor Code and Item code in the first table and Vendor Code and Item code in the second table, we have lost the relationship between Item Code and Project No. If there were a primary key then this loss of dependency would not have occurred. In order to revive this relationship we must add a new table like the following. Please note that during the entire process of normalization, this is the only step where a new table is created by joining two attributes, rather than splitting them into separate tables.

Project No.	Item Code
P1	11
P1	12
P2	11
P3	11
P3	13

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

The above notes are submitted by:

Sabyasachi De

MCA

sabyasachide@yahoo.com