NIBM WORLD WIDE

SOFTWARE ENGINEERING

DATABASE MANAGEMENT SYSTEMS

NORMALIZATION

National Institute of Business Management

Normalization

It is a process of decomposing 'unsatisfactory' relations to smaller relations. Normalization helps eliminate redundancy, organizes data efficiently and reduces potential anomalies during data operations (insertion, updating and deletion operations)

The main Normal forms are:

- First Normal Form (1 NF)
- Second Normal Form (2 NF)
- Third Normal Form (3 NF)
- Boyce Codd Normal Form (BCNF)

First Normal Form (1NF)

The first normal form states that domains of attributes must **include only atomic (simple, indivisible) values** and the values of any attribute in a tuple must be a single value. The 1NF also disallows composite attributes that are themselves multi valued. These are called nested relations because each tuple can have a relation within a relation.

Example 01:

Consider the following Department Table

Dno	Dname	ManagerEno	Dloc
1	HQ	100	Colombo
2	Marketing	200	Colombo Kandy
3	Reserach	300	Galle Gampaha Nuwara Eliya

Table 5.0.1 Department Table

As this relation contains multi valued attributes, it is not in 1 NF. Therefore, break the table into two tables.

Department

Dno	Dname	ManagerEno

Department_Location

Dno	Dloc

Example 02:

Emp-Project {Eno, Ename, {Pno, Hours}}

This relation is an example of a nested relation. Such relations are said to be un-normalized. In order to represent the information in a relational model, normalization must be carried out. This is done by removing the repeating groups.

Emp_Proj

Eno	Ename	Address	Project	
			Pno	Hours
100	Kasun	Col07	4	100
100	Kasun	Col07	7	40
100	Kasun	Col07	2	150
200	Sunil	Kandy	2	100
200	Sunil	Kandy	1	55

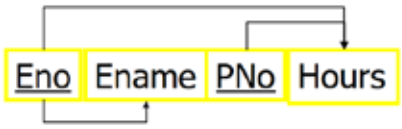
This is not in **First Normal Form** because **there are repeating groups in Ename and Address Column. Therefore, you need break this into two tables.**

Table 5.0.2 Emp_Project Table

To remove the repetitive groups, we need to create a new table for Employee and another table for Emp_Proj with Eno, Pno and Hours attribute.

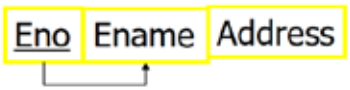
Before Normalization

Employee_Project

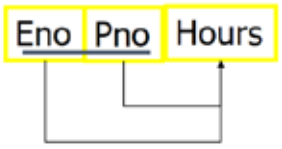


After Normalization

Employee



Emp_Proj



Employee

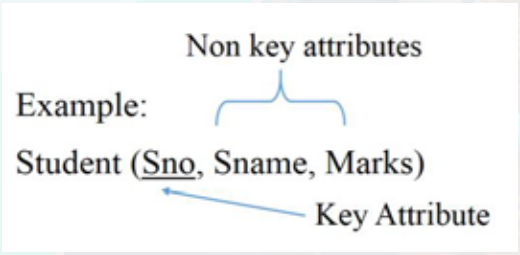
Eno	Ename	Address
100	Kasun	Col07
200	Sunil	Kandy

Emp_Proj

Eno	Pno	Hours
100	4	100
100	7	40
100	2	150
200	2	100
200	1	55

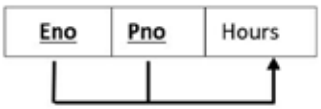
Second Normal Form (2NF)

- Fully Functional Dependency
  - 'B' is fully functionally dependent on 'A', if it is functionally dependent on 'A' and not functionally dependent on any part of 'A'.
  - **2 NF is based on the concept of full functional dependency.**
  - **A relational schema 'R' is in 2 NF if every non-key attribute A in 'R' is fully functionally dependent on the primary key of 'R'.**



This is in Second Normal Form. Because every non key attribute (Name, Address, TP) fully depend on Key Attribute **(Student_No)**

This is in Second Normal Form. Because every non key attribute (Ename, Address, Salary) fully depend on Key Attribute **(Emp_No)**



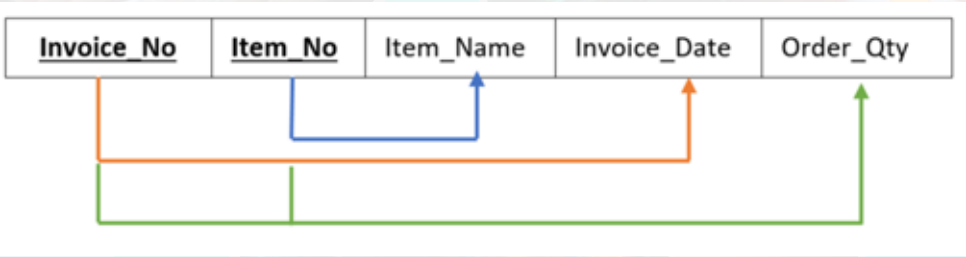
This is in Second Normal Form. Because every non key attribute (Hours) fully depend on Key Attributes **(Eno, Pno)**

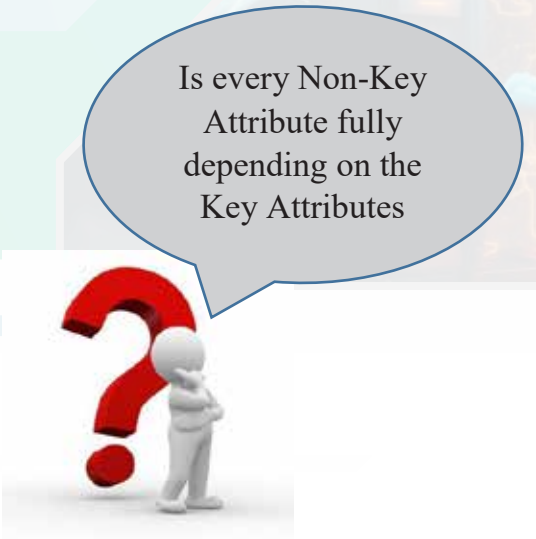
Example 01:

Items (**Invoice_No, Item_No**, Item_Name, Invoice_Date, Order_Qty)

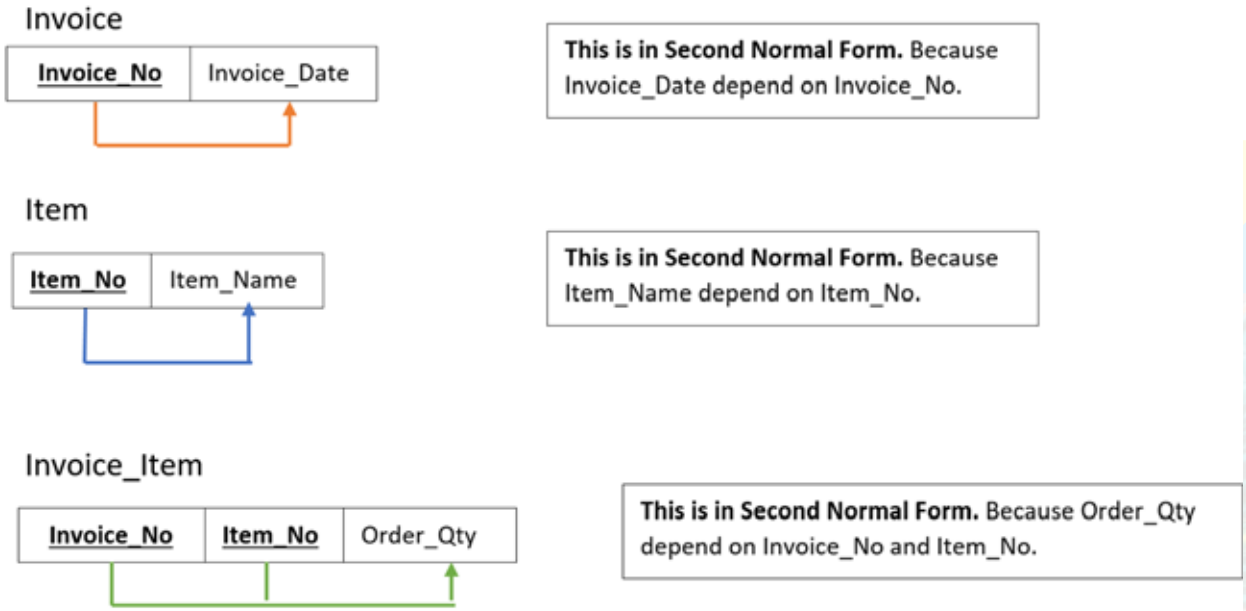
Suppose the PK is {Invoice_No, Item_No} and:

- Invoice_No can be used to find Invoice_Date
- Item_No can be used to find Item_Name
- Invoice_No and Item_No can be used to find Order_Qty.

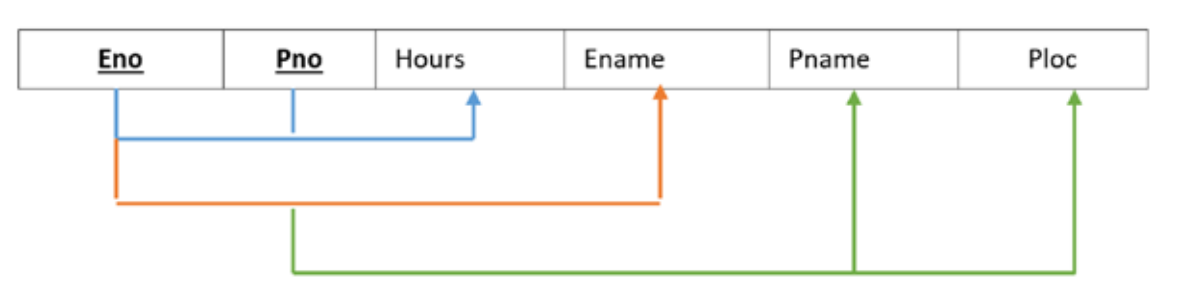




Since non-key attributes Iname and Invdate are not fully functionally dependent on the PK, this relation is not in 2 NF. Therefore, we have to break the relation Items into three relations to satisfy the Second Normal Form:



Example 02:



Is this relation in 2NF? If not what is the reason? Break down this to 2NF.

Since non-key attributes Ename, Pname and Ploc are not fully functionally dependent on the PK, this relation is not in 2 NF.

Therefore, we have to break the relation Items into three relations to satisfy the Second Normal Form:

Employee Table: (**Eno**, Ename)

Project Table: (**Pno**, Pname, Ploc)

Emp_Proj Table: (**Eno, Pno**, Hours)

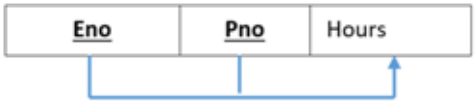
Employee

This is in Second Normal Form because Ename is fully functional depend on Eno

Project

This is in Second Normal Form because Pname, Ploc is fully functional depend on Pno

Emp_Proj



This is in Second Normal Form because Hours is fully functional depend on Eno and Pno (Both Key Attributes)

Third Normal Form (3NF)

Transitive Dependency

If X, Y and Z are attributes and if X ->Y and

Y->Z, then Z is transitively dependent on X.

(X ->Z)

Condition A:

A relation is in 3 NF if and only if it is in 2NF and every non key attribute is non transitively dependent on the primary key.

Condition B:

Suppose in a relation R, a functional dependency X - > A exists, then the following conditions must be satisfied:

X is a super key of R

OR

A is a prime attribute of R (when X is not a super key)

Example 01:

Consider the relation Supplier = {Sno, Pno, Sname, City, Status, Pname, Qty} and the functional dependencies:

{Sno, Pno} -> {Qty}

{Sno} -> {Sname, City}

{Pno} -> {Pname}

{City} ->{Status}

Assume that the primary key of Supplier is {Sno, Pno}. Decompose R in to 3NF relations.

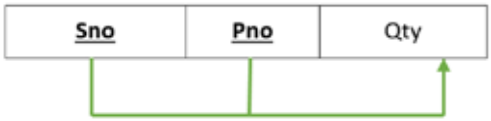
Based on the Fully Functional Dependency concept we need to convert above relation to Second Normal Form.

Second Normal Form

Supplier

Project

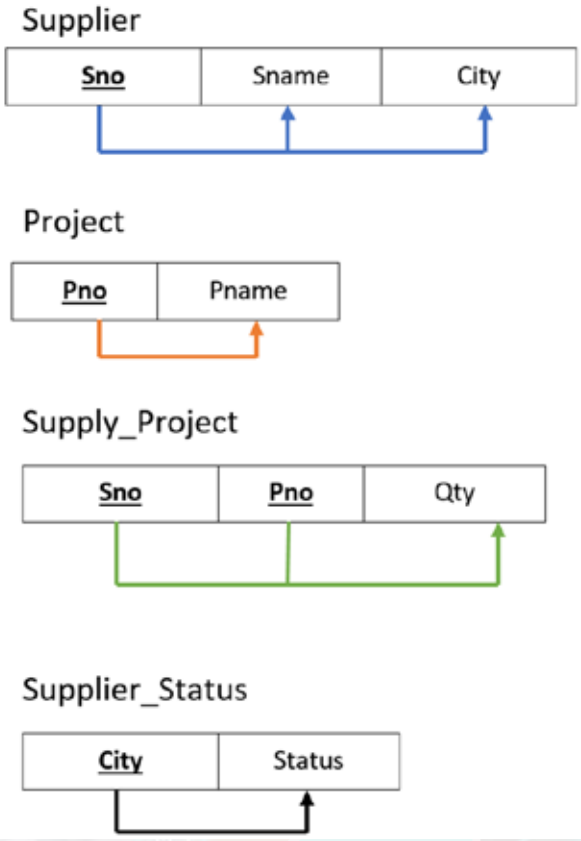
Supply_Project



This is in Second Normal Form because every non key attribute depend on key attributes. But this is not in Third Normal Form because there is a transitive dependency. That means non key attribute depend on another non key attribute

Since Status is depending on City. Above relation is not in Third Normal Form. Therefore, we need to take **City and Status out of the relation and create a new one**.

Third Normal Form



Example 2:

Consider the relation R = {A, B, C, D, E, F, G, H, I} and the functional dependencies,

{A, B} -> {C}

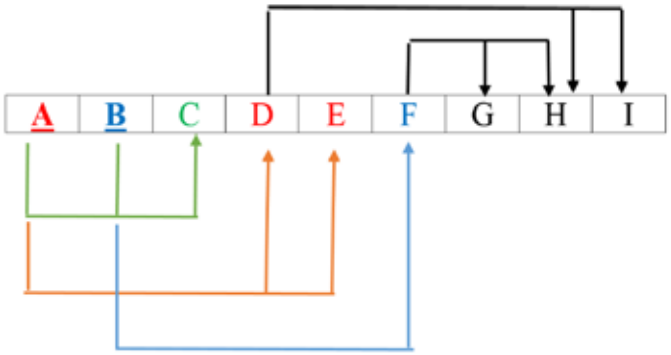
{A} -> {D, E}

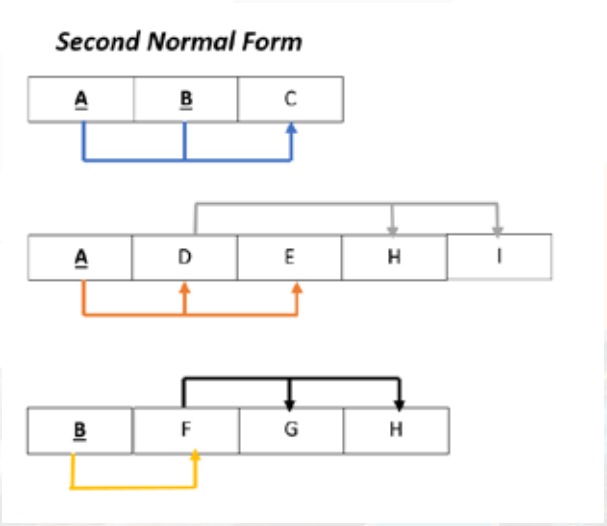
{B} ->{F}

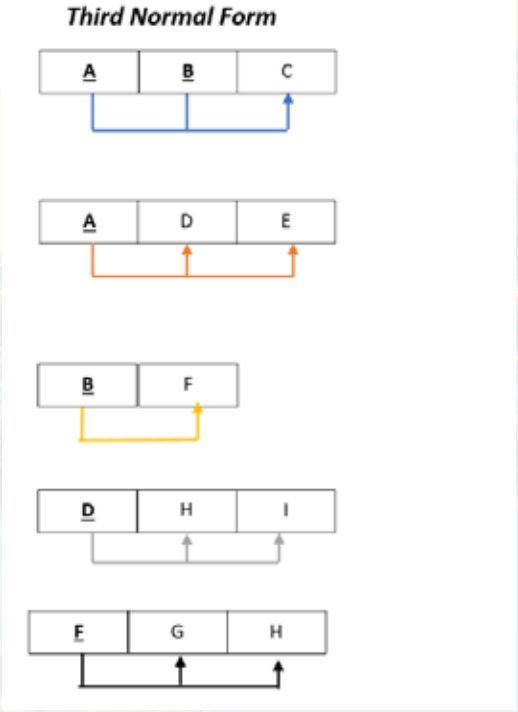
{F}-> {G, H}

{D} -> {H, I}

Assume that the primary keys of R to be A and B. Decompose R into 2NF, then into 3NF relations.







Example 03 (Try this):

Consider the relation R = {A, B, C, D, E, F, G, H, I} and the functional dependencies,

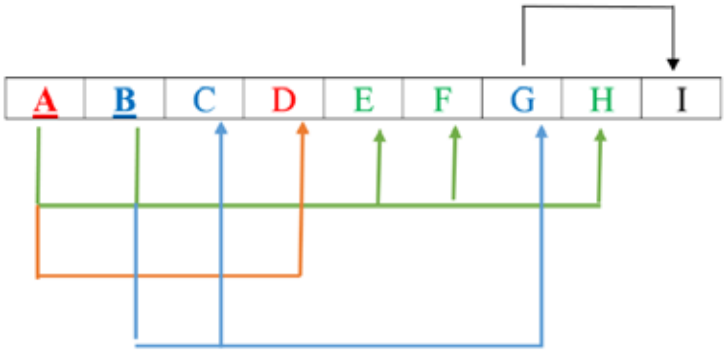
{A, B} -> {E, F, H}

{A} ->{D}

{B} -> {C, G}

{G} -> {I}

Assume that the primary keys of R to be A and B. Decompose R into 2NF, then into 3NF relations.

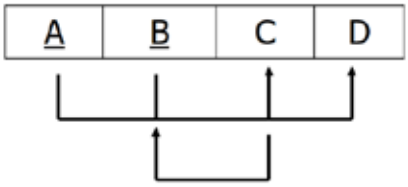


Boyce Codd Normal Form (BCNF)

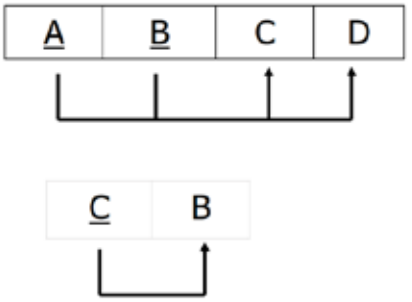
A relational schema R is in BCNF if whenever a functional dependency X ->A holds in R, then X is a super key of R.

Example 1:

The following relation is in 3NF but not in BCNF.

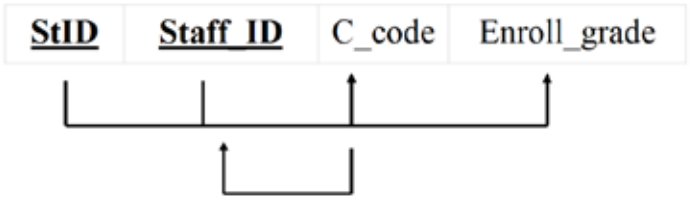


After decomposing the relation to meet BCNF:

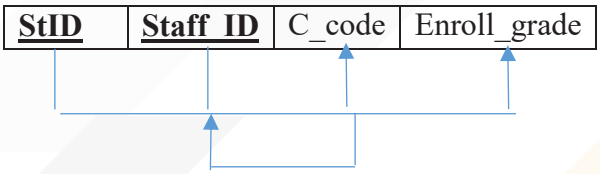


Example 2:

Decompose the following relation to meet BCNF.

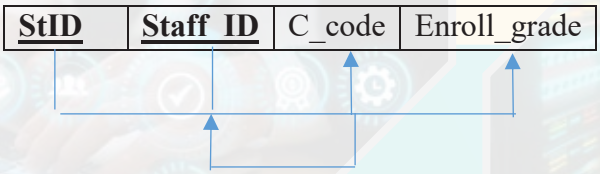


Second Normal Form

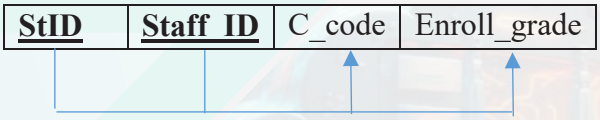


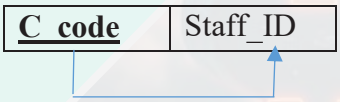
Since there is no transitive dependency this is also in Third Normal Form

Third Normal Form

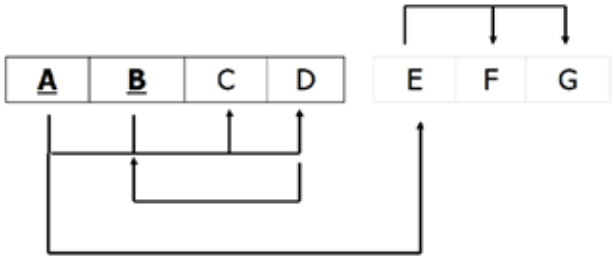


BCNF



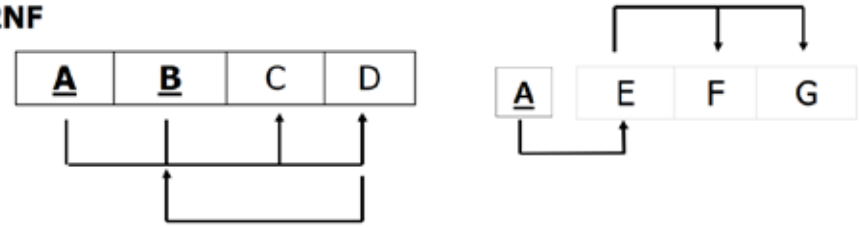


Example 3:

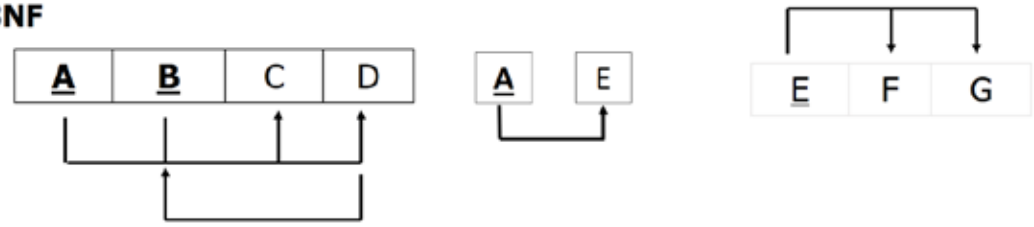


Is this relation normalizing? Decompose the table into suitable normalization form.

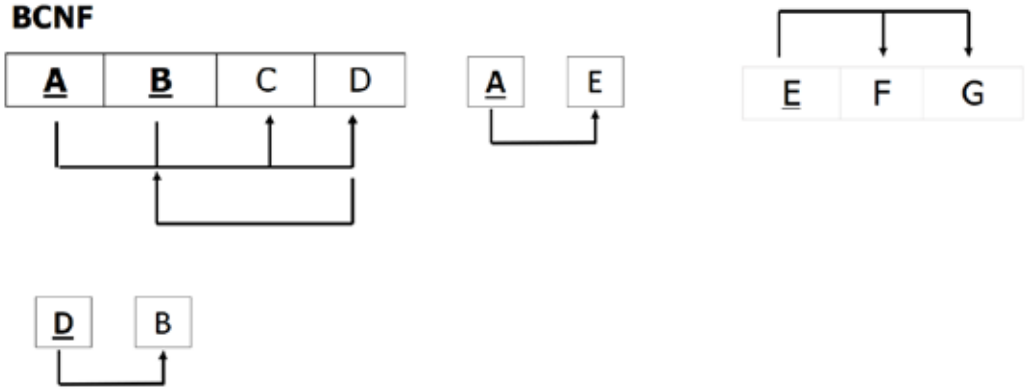
2NF



3NF



BCNF



Summary of Normalization