

COURSE NAME: DBMS

COURSE CODE:23AD2102A

Topic: NORMALIZATION PROCESS

Session - 6

AIM OF THE SESSION

To familiarize students with the basic concept of normalization and its forms

INSTRUCTIONAL OBJECTIVES



This Session is designed to: discuss and study the concepts of normalization, its need, types with examples.

LEARNING OUTCOMES



At the end of this session, you should be able to: understand and apply the concepts of normalization.

Functional Dependency

Functional dependency – In a given relation R , X and Y are attributes. Attribute Y is said to be functionally dependent on attribute X **if each value of X determines exactly one value of Y** . This is represented as $X \rightarrow Y$.

The functional dependency $\alpha \rightarrow \beta$ holds on R if and only if for any legal relations $r(R)$, whenever any two tuples t_1 and t_2 of r agree on the attributes α , they also agree on the attributes β . That is,

Example: Consider $r(A,B)$ with the following instance of r .

A	B
1	4
1	5
3	7

$$t_1[A] = t_2[A] \Rightarrow t_1[B] = t_2[B]$$

On this instance, $A \rightarrow B$ does **NOT** hold, but $B \rightarrow A$ does hold.

**For example,
consider the following relation**

Reports(Student#, Course#, CourseName, Iname, Room#, Marks, Grade)

✓ In this relation, **{Student#, Course#}** together can determine exactly one value of Marks. This can be symbolically represented as

$$\{\mathbf{Student\#, Course\#}\} \rightarrow \mathbf{Marks}$$

- ✓ This type of dependency is called as **Functional Dependency**.
- ✓ In the above example **Marks** is functionally dependent on **{Student#, Course#}**.

Other functional dependencies in the above example are:

- Course# → CourseName
- Course# → Iname (Assuming one course is taught by one and only one instructor)
- Iname → Room# (Assuming each instructor has his/her own room)
- Marks → Grade
- Course# → Room#

Full Functional Dependency - In a given relation R, X and Y are attributes. Attribute Y is fully functionally dependent on attribute X **only if it is not functionally dependent on sub-set of X where X is composite in nature.**

In the above example,

Marks is **fully functionally dependent** on **{Student#, Course#}** and not on sub-set of {Student#, Course#}. This means Marks cannot be determined either by Student# or by Course#. It can be determined using Student# and Course# together. Hence, Marks is fully functionally dependent on {Student#, Course#}.

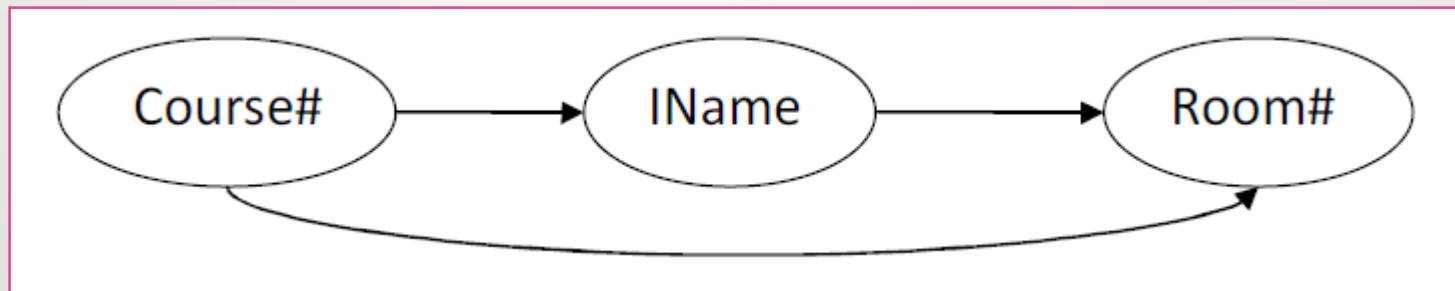
Partial Functional Dependency – In a given relation R, X and Y are attributes. **Attribute Y is partially dependent on attribute X only if it is dependent on sub-set of attribute X where X is composite in nature.**

In the above example,

CourseName, IName, Room# are **partially dependent** on {Student#, Course#} because Course# alone determines the CourseName, IName, Room#.

Transitive Dependency – In a given relation R, if attribute X determines attribute Y and attribute Y determines attribute Z, then attribute X determines attribute Z. Such a dependency is called **transitive dependency**.

Following example shows a transitive dependency.



DETERMINANT

A determinant in a database table is any attribute that you can use to determine the values assigned to other attribute(s) in the same row.

Examples: Consider a table with the attributes `employee_id`, `first_name`, `last_name` and `date_of_birth`. In this case, the field `employee_id` determines the remaining three fields. The name fields do not determine the `employee_id` because the firm may have more than one employee with the same first and/or last name. Similarly, the DOB field does not determine the `employee_id` or the name fields because more than one employee may share the same birthday.

Trivial Dependency-A Functional Dependency $X \rightarrow Y$ is said to be trivial functional dependency if Y is a subset of X ($Y \subseteq X$). In other words if R.H.S of some FD is the subset of L.H.S of FD is called Trivial Functional Dependency.

Example: $AB \rightarrow A$

$AB \rightarrow B$

$AB \rightarrow AB$

Non-trivial – If an FD $X \rightarrow Y$ holds, where Y is not a subset of X , then it is called a non-trivial FD.

Completely non-trivial – If an FD $X \rightarrow Y$ holds, where $X \cap Y = \emptyset$, it is said to be a completely non-trivial FD.

Properties of Functional Dependencies/Axioms/Inference Rules

- **Armstrong's axioms** are a set of rules, that when applied repeatedly generates a closure of functional dependencies

1. Reflexive Rules: if X is a set of attributes and Y is a subset of X then X holds Y.

$$Y \subseteq X \Rightarrow X \rightarrow Y$$

2. Augmentation Rule: if X hold Y and Z is a set of attributes then XZ holds YZ.

$$X \rightarrow Y \text{ then } XZ \rightarrow YZ$$

3. Transitive Rule: if X holds Y and Y holds Z , then X holds Z.

$$X \rightarrow Y \text{ and } Y \rightarrow Z \text{ then } X \rightarrow Z$$

4. Additive or Union Rule: if X holds Y and X holds Z ,then X holds YZ.

$$X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

5. Pseudo Transitive Rule: if X holds Y and YZ holds W, then XZ holds W.

$$X \rightarrow Y \text{ and } YZ \rightarrow W \text{ then } XZ \rightarrow W$$

6. Productive Rule or Decomposition Rule: if X holds YZ and X holds Y, then X holds Z.

$$X \rightarrow YZ \text{ then } X \rightarrow Y \text{ and } X \rightarrow Z$$

4.Attribute Closure:

The set of all those attributes which can be functionally determined from an attribute set is called as closure of that attribute set.

Closure of an attribute set $\{X\}$ is denoted as $\{x\}^+$.

Steps to find closure of an attribute set:

1. Add the attributes contained in the attribute set for which closure is being calculated to the result set.
2. Recursively add the attributes to the result set which can be functionally determined from the attributes already contained in the result set.

Example: Consider a relation $R(A,B,C,D,E,F,G)$ with the functional dependencies

$A \rightarrow BC$

$BC \rightarrow DE$

$D \rightarrow F$

$CF \rightarrow G$

Closure of Attribute A:

$$A^+ = \{A\}$$

$$= \{ABC\} \quad \because A \rightarrow BC$$

$$= \{ABCDE\} \quad \because BC \rightarrow DE$$

$$= \{ABCDEF\} \quad \because D \rightarrow F$$

$$= \{ABCDEFG\} \quad \because CF \rightarrow G$$

Closure of Attribute set BC:

$$BC^+ = \{BC\}$$


$$= \{BCDE\} \quad \because BC \rightarrow DE$$

$$= \{BCDEF\} \quad \because D \rightarrow F$$


$$= \{BCDEFG\} \quad \because CF \rightarrow G$$

What is Redundancy?

Roll_List					
ID	Name	Dept	HoD	Phno	
101	Ram	CSE	Pr.A	123456789	
102	Sita	ECE	Pr.B	555555555	
103	Ravi	CSE	Pr.A	123456789	
104	Kalyan	IT	Pr.C	555622231	
105	Karthik	CSE	Pr.A	123456789	



Student		
ID	Name	Dept
101	Ram	CSE
102	Sita	ECE
103	Ravi	CSE
104	Kalyan	IT
105	Karthik	CSE



Department		
Dept	HoD	Phno
CSE	Pr.A	123456789
ECE	Pr.B	555555555
IT	Pr.C	555622231

- Consider a relation **Roll_list** Initially with attributes **ID, Name, Dept.**
- later** add the attributes **HoD** and **Phno** and observe the highlighted tuples how Redundancy increases.
- By **decomposing** the relation(roll_list) into two sub relations redundancy can be minimized.

Problems Caused by Redundancy

Storing the same information redundantly, that is, in more than one place within a database, can lead to several problems:

- **Redundant storage:** Some information is stored repeatedly.
- **Update anomalies:** If one copy of such repeated data is updated, an inconsistency is created unless all copies are similarly updated.
- **Insertion anomalies:** It may not be possible to store some information unless some other information is stored as well.
- **Deletion anomalies:** It may not be possible to delete some information without losing some other information as well.

<i>ssn</i>	<i>name</i>	<i>lot</i>	<i>rating</i>	<i>hourly_wages</i>	<i>hours_worked</i>
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

❑ For example the rating value 8 corresponds to the hourly wage 10, and this association is repeated three times. In addition to wasting space by storing the same information many times, redundancy leads to potential inconsistency. For example, the *hourly wages in the first tuple* could be updated without making a similar change in the second tuple, which is an example of an **update anomaly**.

❑ Also we cannot insert a tuple for an employee **unless we know the hourly wage for the employee's rating value**, which is an example of an **insertion anomaly**.

- If we delete all tuples with a given rating value (e.g., we delete the tuples for Smethurst and Guldu) we lose the association between that *rating* value and its *hourly_wage* value (a *deletion anomaly*).

Normalization

Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion anomalies.

- These would include two properties
 - **Lossless join property**- which guarantees that the generation of spurious tuples will not occur.
 - **Dependency preservation property** - This ensures that each functional dependency is represented in some individual relation resulting after decomposition.
- **Prime attribute** - An attribute of relation schema R is called a prime attribute of R if it is a member of some candidate key of R.
- **Non prime attribute** - An attribute is called nonprime if it is not a prime attribute—that is, if it is not a member of any candidate key.

If a relation schema has more than one key, each is called a candidate key. One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary keys.

Normal forms: The normal form is a relation refers to the highest normal form condition that it meets and hence indicates the degree to which it has been normalized.

Normal forms are used to eliminate or reduce redundancy in database tables.

Types of Normal Forms

- 1st Normal Form(1NF)
- 2nd Normal Form(2NF)
- 3rd Normal Form(3NF)
- Boyce-Codd Normal Form (BCNF)
- 4th Normal Form(4NF)
- 5th Normal Form(5NF)

First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following four rules:

1. It should only have single(atomic) valued *attributes/columns*.
2. Values stored in a column should be of the same domain.
3. All the columns in a table should have unique names.
4. The order in which data is stored, does not matter.

Summary: A relation R is said to be in the first normal form **if and only if all the attributes of the relation R are atomic in nature.**

For example, consider the Department table given in figure (b). It is not in 1NF because one of its attributes Dlocations is non-atomic as it contains more than one value in row 1. To make it 1NF compliant, create a separate row for each value of Dlocations of row 1 as shown in figure (c).

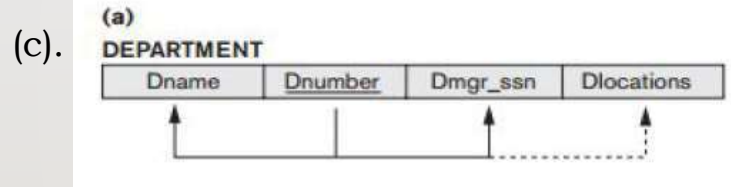


Figure (a)
A relation schema

(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Fig (c) 1NF version
of the same relation

(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Fig b) Sample State of relation Department
that is **not in 1NF**

EXAMPLE 2: STUDENT_COURSE_RESULT TABLE

Student_Details			Course_Details				Results		
101	Davis	11/4/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	82	A
102	Daniel	11/6/1987	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	62	C
101	Davis	11/4/1986	H6	American History		4	11/22/2004	79	B
103	Sandra	10/2/1988	C3	Bio Chemistry	Basic Chemistry	11	11/16/2004	65	B
104	Evelyn	2/22/1986	B3	Botany		8	11/26/2004	77	B
102	Daniel	11/6/1987	P3	Nuclear Physics	Basic Physics	13	11/12/2004	68	B
105	Susan	8/31/1985	P3	Nuclear Physics	Basic Physics	13	11/12/2004	89	A
103	Sandra	10/2/1988	B4	Zoology		5	11/27/2004	54	D
105	Susan	8/31/1985	H6	American History		4	11/22/2004	87	A
104	Evelyn	2/22/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	65	B

Table in 1NF

Student_Course_Result Table

Student#	Student Name	Dateof Birth	Cours #	CourseName	Pre Requisite	Duration In Days	DateOf Exam	Marks	Grade
101	Davis	04-Nov-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	82	A
102	Daniel	06-Nov-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	62	C
101	Davis	04-Nov-1986	H6	American History		4	22-Nov-2004	79	B
103	Sandra	02-Oct-1988	C3	Bio Chemistry	Basic Chemistry	11	16-Nov-2004	65	B
104	Evelyn	22-Feb-1986	B3	Botany		8	26-Nov-2004	77	B
102	Daniel	06-Nov-1986	P3	Nuclear Physics	Basic Physics	13	12-Nov-2004	68	B
105	Susan	31-Aug-1985	P3	Nuclear Physics	Basic Physics	13	12-Nov-2004	89	A
103	Sandra	02-Oct-1988	B4	Zoology		5	27-Nov-2004	54	D
105	Susan	31-Aug-1985	H6	American History		4	22-Nov-2004	87	A
104	Evelyn	22-Feb-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	65	B

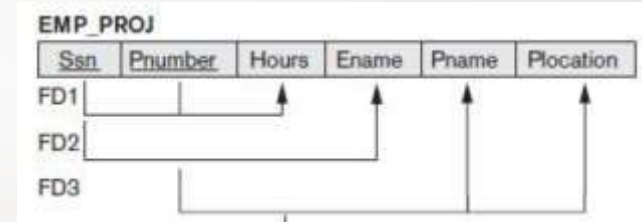
Second Normal Form (2NF)

- A Relation is said to be in **Second Normal Form (2NF)** if and only if:
 - It is in First normal form (1NF)
 - No partial dependency exists between non-key attributes and key attributes

- Partial Dependency exists, when for a composite primary key, **any attribute in the table depends only on a part of the primary key and not on the complete primary key.**

- As an example, consider the EMP_PROJ schema shown below;
- For this table, the key is {Ssn, Pnumber}.
- The functional dependencies are as follows

- $\{Ssn, Pnumber\} \rightarrow Hours$
- $Ssn \rightarrow Ename$
- $Pnumber \rightarrow Pname$
- $Pnumber \rightarrow Plocation$



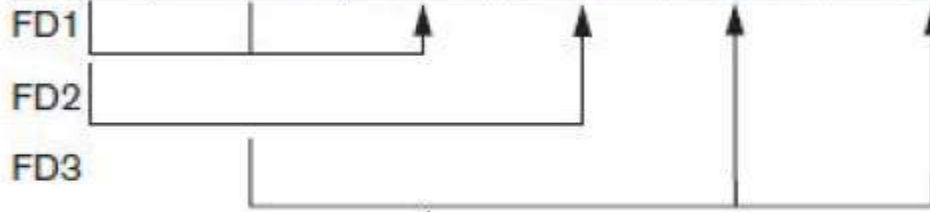
- It is clear from these functional dependencies that the table has partial dependencies and **it is not in 2NF**

$(Ssn \rightarrow Ename, Pnumber \rightarrow Pname, Pnumber \rightarrow Plocation)$

- To make it 2NF compliant, remove all partial dependencies.
- For this, we need to split EMP_PROJ table into 3 tables as EP1, EP2 and EP3 as shown above
- To **remove Partial dependency**, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

EMP_PROJ

<u>Ssn</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation
------------	----------------	-------	-------	-------	-----------



2NF Normalization

EP1

<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------



EP2

<u>Ssn</u>	Ename
------------	-------



EP3

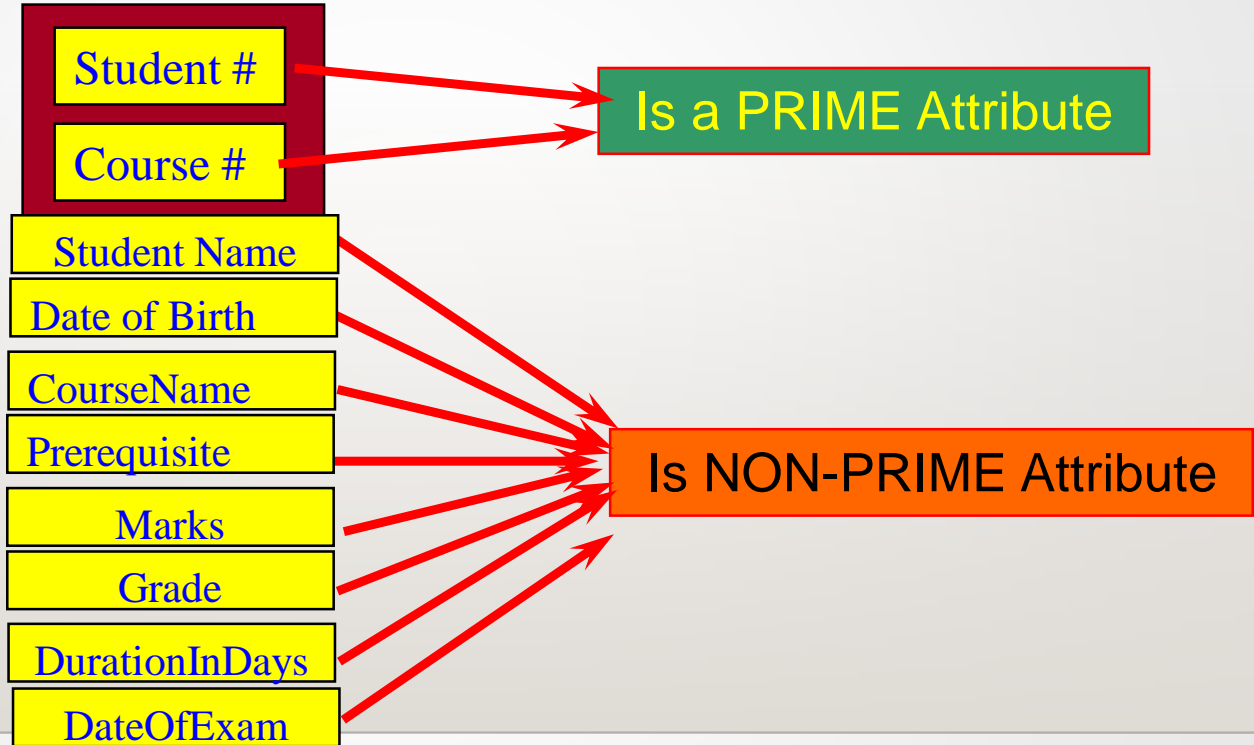
<u>Pnumber</u>	Pname	Plocation
----------------	-------	-----------



Now these 3 tables do not contain partial dependencies and hence they are in 2NF.

EXAMPLE 2: SECOND NORMAL FORM

Report(S#,C#,StudentName,DateOfBirth,CourseName,PreRequisite,DurationInDays,DateOfExam,Marks,Grade)



EXAMPLE 2: SECOND NORMAL FORM

- STUDENT# is key attribute for Student,
- COURSE# is key attribute for Course
- STUDENT# COURSE# together form the composite key attributes for Results relationship.
- Other attributes like StudentName (Student Name), DateofBirth, CourseName, PreRequisite, DurationInDays, DateofExam, Marks and Grade are non-key attributes.

To make this table 2NF compliant, we have to remove all the partial dependencies.

SECOND NORMAL FORM

S#,C#



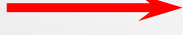
Marks

S#,C#



Grade

S#



StudentName

S#



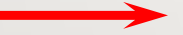
DOB

C#



CourseName

C#



Prerequisite

C#



Duration

C#



DateOfExam

Fully Functionally dependent on
composite Candidate key

Partial Dependency

Partial Dependency

Partial Dependency

Let us re-visit our 1NF table structure.

STUDENT# is key attribute for Student, COURSE# is key attribute for Course

STUDENT# COURSE# together form the composite key attributes for Results relationship.

Other attributes like StudentName (Student Name), DateofBirth, CourseName, PreRequisite, DurationInDays, DateofExam, Marks and Grade are non-key attributes.

To make this table 2NF compliant, we have to remove all the partial dependencies.

StudentName, DateofBirth, Address depends only on STUDENT#

CourseName, PreRequisite, DurationInDays depends only on COURSE#

DateofExam depends only on COURSE# Marks and Grade depends on STUDENT# COURSE#

To remove this partial dependency we can create four separate tables, Student, Course and Result Exam_Date tables as shown below.

In the first table (STUDENT), the key attribute is STUDENT# and all other non-key attributes are fully functionally dependant on the key attributes.

In the second table (COURSE), COURSE# is the key attribute and all the non-key attributes are fully functionally dependant on the key attributes.

In third table (Result) STUDENT# COURSE# together are key attributes and all other non key attributes Marks and Grade fully functionally dependant on the key attributes.

In the fourth table (Exam_Date), DateOfExam depends only on Course#.

These four tables also are compliant with First Normal Form definition. Hence these four tables are in Second Normal Form (2NF).

SECOND NORMAL FORM - TABLES IN 2 NF

STUDENT TABLE

Student#	StudentName	DateofBirth
101	Davis	04-Nov-1986
102	Daniel	06-Nov-1987
103	Sandra	02-Oct-1988
104	Evelyn	22-Feb-1986
105	Susan	31-Aug-1985
106	Mike	04-Feb-1987
107	Juliet	09-Nov-1986
108	Tom	07-Oct-1986
109	Catherine	06-Jun-1984

COURSE TABLE

Course#	Course Name	Pre Requisite	Duration InDays
M1	Basic Mathematics		11
M4	Applied Mathematics	M1	7
H6	American History		4
C1	Basic Chemistry		5
C3	Bio Chemistry	C1	11
B3	Botany		8
P1	Basic Physics		8
P3	Nuclear Physics	P1	13
B4	Zoology		5



SECOND NORMAL FORM – TABLES IN 2 NF

Marks Garde Table

Student#	Cours e#	Marks	Grad e
101	M4	82	A
102	M4	62	C
101	H6	79	B
103	C3	65	B
104	B3	77	B
102	P3	68	B
105	P3	89	A
103	B4	54	D
105	H6	87	A
104	M4	65	B

Exam_Date Table

Course#	DateOfExam
M4	11-Nov-04
H6	22-Nov-04
C3	16-Nov-04
B3	26-Nov-04
P3	12-Nov-04
B4	27-Nov-04

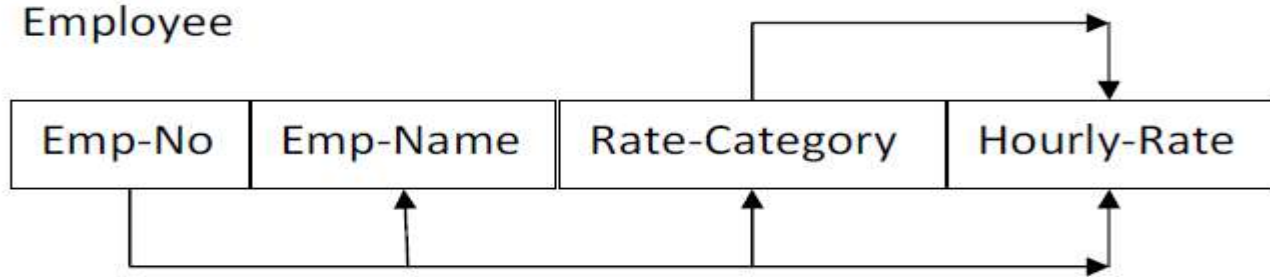
Third Normal Form (3NF)

- ❑ A Relation is said to be in **Third Normal Form (3NF)** if and only if:
 - ❑ It is in Second Normal Form (2NF)
 - ❑ **No transitive dependency** exists between non-key attributes and key attributes.
- ❑ When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key, then we say it is ***Transitive Dependency***

How to remove Transitive Dependency?

- Again the solution is very simple.
- Take out the columns that caused transitive dependency from the existing table and move them to some other table where it fits in well.

For example, consider the following Employee table.



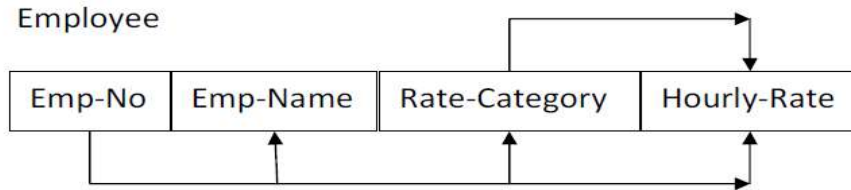
This table contains a **transitive dependency** as given below;

Emp-No → Rate-Category → Hourly-Rate

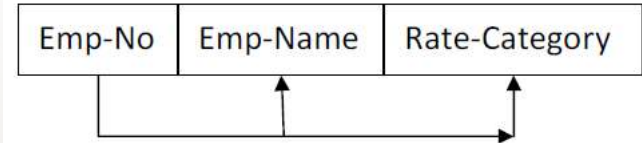
Hence, Employee **table is not in 3NF**

- ❑ To make it **3NF compliant**, we need to remove this transitive dependency.
- ❑ To do that, we need to split Employee table into two tables (Employee table and Rate table) as given below

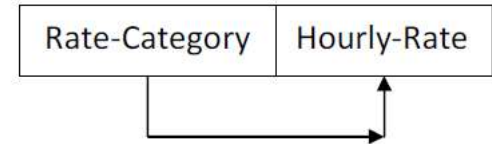
For example, consider the following Employee table.



Employee Table



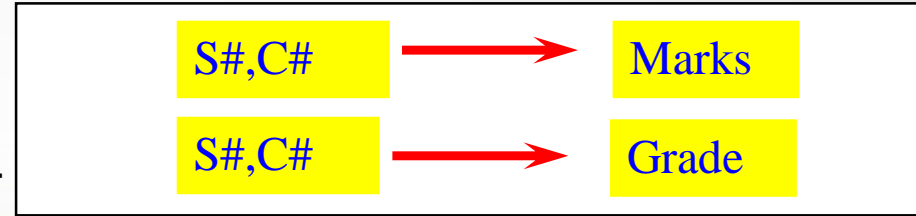
Rate Table



- ❑ Now, both Employee and Rate tables are in 3NF as **they do not have transitive dependencies.**

EXAMPLE 2: THIRD NORMAL FORM:

- STUDENT# and COURSE# are the key attributes.
- All other attributes, except grade are non-partially, non-transitively dependent on key attributes.
- Student#, Course# - > Marks
- Marks -> Grade



Note : - All transitive dependencies are eliminated



EXAMPLE 2: THIRD NORMAL FORM:

Student #	Course#	Marks
101	M4	82
102	M4	62
101	H6	79
103	C3	65
104	B3	77
102	P3	68
105	P3	89
103	B4	54
105	H6	87
104	M4	65

MARKSGRADE TABLE

UpperBound	LowerBound	Grade
100	95	A+
94	85	A
84	70	B
69	65	B-
64	55	C
54	45	D
44	0	E

Boyce-Codd Normal Form (BCNF)

- ❖ A relation is said to be in BCNF *if and only if all the determinants are candidate keys.*
- ❖ BCNF relation is a strong 3NF relation. i.e. all BCNF relations are in 3NF *but the reverse is not true.*

For example, consider the following Result table

Result Table			
Student#	EmailID	Course#	Marks

This table has two candidate keys – {**Student#, Course#**} and {**EmailID, Course#**}.

- ❑ This table is in **3NF** because it has **no partial** and **transitive dependencies** between key attributes and non-key attributes.
- ❑ But this table is **Not in BCNF** because **all determinants are not candidate keys**.
- ❑ This can be observed from the following FDs;

- $\{ \text{Student\#}, \text{Course\#} \} \rightarrow \text{Marks}$
- $\{ \text{EmailID}, \text{Course\#} \} \rightarrow \text{Marks}$
- $\text{Student\#} \rightarrow \text{EmailID}$
- $\text{EmailID} \rightarrow \text{Student\#}$

❑ Though the determinants of **first** two FDs are candidate keys but the determinants of the last two FDs are not candidate keys. Thus it is violating the BCNF condition

let a FD be

$$X \rightarrow Y$$

Here '**X**' is called **Determinant**
and '**Y**' is called **Dependent**

- ❑ To make this table into BCNF compliant, we need to decompose the Result table into two tables as shown below;

Result Table

Student#	EmailID	Course#	Marks
----------	---------	---------	-------



Student Table

Student#	EmailID
----------	---------

Result Table

Student#	Course#	Marks
----------	---------	-------

Before Normalization

STUDENT		
<u>sid</u>	<u>Subject</u>	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

In the table above:

One student can enrol for multiple subjects. For example, student with **student_id** 101, has opted for subjects - Java & C++

For each subject, a professor is assigned to the student.

And, there can be multiple professors teaching one subject like we have for Java.

What do you think should be the **Primary Key**?

Well, in the table above **student_id**, **subject** together form the primary key, because using **student_id** and **subject**, we can find all the columns of the table.

One more important point to note here is, one professor teaches only one subject, but one subject may have two different professors.

Hence, there is a dependency between **subject** and **professor** here, where **subject** depends on the professor name.

This table satisfies the **1st Normal form** because all the values are atomic, column names are unique and all the values stored in a particular column are of same domain.

This table also satisfies the **2nd Normal Form** as there is no **Partial Dependency**.

And, there is no **Transitive Dependency**, hence the table also satisfies the **3rd Normal Form**.

But this table is not in **Boyce-Codd Normal Form**.

Why this table is not in BCNF?

In the table above, **student_id**, **subject** form primary key, which means **subject** column is a **prime attribute**.

But, there is one more dependency, **professor** → **subject**.

And while **subject** is a prime attribute, **professor** is a **non-prime attribute**, which is not allowed by BCNF.

How to satisfy BCNF?

To make this relation(table) satisfy BCNF, we will decompose this table into two tables, **student** table and **professor** table.

After Normalization

STUDENT	
<u>sid</u>	<u>pid</u>
101	1
101	2
102	3
103	4
104	1

PROFESSOR		
<u>pid</u>	professor	subject
1	P.Java	Java
2	P.Cpp	C++
3	P.Java2	Java
4	P.Chash	C#

Fourth Normal Form(4NF)

Definition:

A **Relation** R is in 4NF if and only if the following conditions are satisfied

- (i) R is in already BCNF or 3NF
- (ii) R contains no Multivalued dependencies (MVDs)

A **multivalued dependency** $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that $t1[X] = t2[X]$, then two tuples t3 and t4 should also exist in r with the following properties, where we use Z to denote $(R - (X \cup Y))$:

A table is said to have multi-valued dependency, if the following conditions are true.

1. For a dependency $A \twoheadrightarrow B$, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
3. And, for a relation $R(A,B,C)$, if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.
4. If all these conditions are true for any relation (table), it is said to have multi-valued dependency.

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$

(a) **EMP** The EMP relation with two MVDs: $\text{Ename} \twoheadrightarrow \text{Pname}$ and $\text{Ename} \twoheadrightarrow \text{Dname}$

<u>Ename</u>	<u>Pname</u>	<u>Dname</u>
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$

(b) **EMP_PROJECTS**

<u>Ename</u>	<u>Pname</u>
Smith	X
Smith	Y

EMP_DEPENDENTS

<u>Ename</u>	<u>Dname</u>
Smith	John
Smith	Anna

Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

Fourth Normal Form

<u>courseno</u>	<u>lecturer</u>	<u>text</u>
cs250	lindsey	Intro to SML
cs250	lindsey	SML for beginners
cs250	lindsey	More SML
cs250	capon	Intro to ML
cs250	capon	ML for beginners
cs250	capon	More SML
cs260	kahn	Raster Graphics
cs260	kahn	Ray Tracing for Fun
cs260	bush	Raster Graphics
cs260	bush	Ray Tracing for Fun
cs270	zobel	Chips with everything
cs270	zobel	Intro to Electronics
cs270	woods	Chips with everything
cs270	woods	Intro to Electronics
cs280	capon	Object Design



<u>courseno</u>	<u>lecturer</u>
cs250	lindsey
cs250	capon
cs260	kahn
cs260	bush
cs270	zobel
cs270	woods
cs280	capon

trivial dependencies only

<u>courseno</u>	<u>text</u>
cs250	Intro to SML
cs250	SML for beginners
cs250	More SML
cs260	Raster Graphics
cs260	Ray Tracing for Fun
cs270	Chips with everything
cs270	Intro to Electronics
cs280	Object Design

$\text{courseno} \twoheadrightarrow \text{lecturer}$

$\text{courseno} \twoheadrightarrow \text{text}$

Fifth Normal Form(5NF) or Projection Join Normal Form(PJ/NF)

Definition:

A **Relation** R is in 5NF if and only if the following conditions are satisfied

- (i) R is in already 4NF
- (ii) R contains no JOIN Dependencies or it cant be further non loss decomposed

➤ **Fifth Normal Form** in Database Normalization is generally not implemented in real life database design.

Join Dependency

Let 'R' be a relation schema and $R_1, R_2, R_3, \dots, R_n$ be the decomposition of R, R is said to satisfy the join dependency $(R_1, R_2, R_3, \dots, R_n)$ if and only iff:

$$\Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \bowtie \dots \bowtie \Pi_{R_n}(R) = R$$

(Order of join doesn't matter)

Is this relation is in 5NF?

PNAME	SKILL	JOB
AMAN	DBA	J1
MOHAN	TESTER	J2
ROHAN	PROGRAMMER	J3
SOHAN	ANALYST	J1

R1(Pname, Skill)	
Pname	Skill
Aman	DBA
Mohan	Tester
Rohan	Programmer
Sohan	Analyst

R2(Pname, Job)	
Pname	Job
Aman	J1
Mohan	J2
Rohan	J3
Sohan	J1

R3(Skill, Job)	
Skill	job
DBA	J1
Tester	J2
Programmer	J3
Analyst	J1

$R1 \bowtie R2 \bowtie R3 = R$

PNAME	SKILL	JOB
AMAN	DBA	J1
MOHAN	TESTER	J2
ROHAN	PROGRAMMER	J3
SOHAN	ANALYST	J1

Conclusion:
This relation is not in 5NF

Join Dependencies

- There are relations where a nonadditive-join decomposition can only be realized with more than two relation schemas

Supply

Supplier	Part	Proj
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Smith	Bolt	ProjY

- If a supplier s supplies part p , a project j uses part p , and the supplier s supplies at least one part to project j , then supplier s also supplies part p to project j
- Relation is “all key”, involves no nontrivial FDs or MVDs \Rightarrow is in 4NF

*** Non additive join decomposition:** When the sub relations combine again then the new relation must be the same as the original relation was before decomposition. Consider a relation R if we decomposed it into sub-parts relation R1 and relation R2.

Join Dependencies, cont.

Sup_Part

Supplier	Part
Smith	Bolt
Smith	Nut
Adamsky	Bolt

Part_Proj

Part	Proj
Bolt	ProjX
Nut	ProjY
Bolt	ProjY

Sup_Proj

Supplier	Proj
Smith	ProjX
Smith	ProjY
Adamsky	ProjY

Join over Part

Supplier	Part	Proj
Smith	Bolt	ProjX
Smith	Bolt	ProjY
Smith	Nut	ProjY
Adamsky	Bolt	ProjX
Adamsky	Bolt	ProjY

spurious

Join over Supplier and Proj

Original relation Supply

Fifth Normal Form (5NF)

- A relation schema R is in 5NF w.r.t. a set F of FDs, MVDs, and JDs if for every nontrivial $JD(R_1, R_2, \dots, R_n)$, each R_i is a superkey

Supplier	Part	Proj
----------	------	------

↓ 5NF normalization

Supplier	Part
----------	------

Supplier	Proj
----------	------

Part	Proj
------	------

- 5NF is also called PJNF (project-join normal form)
- Since a MVD is a special case of a JD, every relation in 5NF is also in 4NF
- Every relation can be non losslessly decomposed into 5NF relations

DECOMPOSITION ALGORITHMS FOR NORMALIZATION

Algorithm 15.2. Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E .

Note: Explanatory comments are given at the end of some of the steps. They follow the format: (**comment**).

1. Set $F := E$.
2. Replace each functional dependency $X \rightarrow \{A_1, A_2, \dots, A_n\}$ in F by the n functional dependencies $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n$. (**This places the FDs in a canonical form for subsequent testing**)
3. For each functional dependency $X \rightarrow A$ in F
 - for each attribute B that is an element of X
 - if $\{ \{F - \{X \rightarrow A\} \} \cup \{ (X - \{B\}) \rightarrow A \} \}$ is equivalent to F
 - then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F .
 - (**This constitutes removal of an extraneous attribute B contained in the left-hand side X of a functional dependency $X \rightarrow A$ when possible**)
4. For each remaining functional dependency $X \rightarrow A$ in F
 - if $\{F - \{X \rightarrow A\}\}$ is equivalent to F ,
 - then remove $X \rightarrow A$ from F . (**This constitutes removal of a redundant functional dependency $X \rightarrow A$ from F when possible**)

Example 1: Let the given set of FDs be $E: \{B \rightarrow A, D \rightarrow A, AB \rightarrow D\}$. We have to find the minimal cover of E .

- All above dependencies are in canonical form (that is, they have only one attribute on the right-hand side), so we have completed step 1 of Algorithm 15.2 and can proceed to step 2. In step 2 we need to determine if $AB \rightarrow D$ has any redundant (extraneous) attribute on the left-hand side; that is, can it be replaced by $B \rightarrow D$ or $A \rightarrow D$?
- Since $B \rightarrow A$, by augmenting with B on both sides (IR2), we have $BB \rightarrow AB$, or $B \rightarrow AB$ (i). However, $AB \rightarrow D$ as given (ii).
- Hence by the transitive rule (IR3), we get from (i) and (ii), $B \rightarrow D$. Thus $AB \rightarrow D$ may be replaced by $B \rightarrow D$.
- We now have a set equivalent to original E , say $E': \{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$. No further reduction is possible in step 2 since all FDs have a single attribute on the left-hand side.
- In step 3 we look for a redundant FD in E' . By using the transitive rule on $B \rightarrow D$ and $D \rightarrow A$, we derive $B \rightarrow A$. Hence $B \rightarrow A$ is redundant in E' and can be eliminated.
- Therefore, the minimal cover of E is $F: \{B \rightarrow D, D \rightarrow A\}$.

The reader can verify that the original set F can be inferred from E ; in other words, the two sets F and E are equivalent.

Algorithm 15.4 Relational Synthesis into 3NF with Dependency Preservation and Nonadditive Join Property

Input: A universal relation R and a set of functional dependencies F on the attributes of R .

1. Find a minimal cover G for F (use Algorithm 15.2).
2. For each left-hand-side X of a functional dependency that appears in G , create a relation schema in D with attributes $\{X \cup \{A_1\} \cup \{A_2\} \dots \cup \{A_k\}\}$, where $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$ are the only dependencies in G with X as left-hand side (X is the key of this relation).
3. If none of the relation schemas in D contains a key of R , then create one more relation schema in D that contains attributes that form a key of R . (Algorithm 15.2(a) may be used to find a key.)
4. Eliminate redundant relations from the resulting set of relations in the relational database schema. A relation R is considered redundant if R is a projection of another relation S in the schema; alternately, R is subsumed by S .⁷

Step 3 of Algorithm 15.4 involves identifying a key K of R . Algorithm 15.2(a) can be used to identify a key K of R based on the set of given functional dependencies F . Notice that the set of functional dependencies used to determine a key in Algorithm 15.2(a) could be either F or G , since they are equivalent.

Example 1 of Algorithm 15.4. Consider the following universal relation:

$U (\text{Emp_ssn}, \text{Pno}, \text{Esal}, \text{Ephone}, \text{Dno}, \text{Pname}, \text{Plocation})$

Emp_ssn, Esal, and Ephone refer to the Social Security number, salary, and phone number of the employee. Pno, Pname, and Plocation refer to the number, name, and location of the project. Dno is the department number.

The following dependencies are present:

FD1: $\text{Emp_ssn} \rightarrow \{\text{Esal}, \text{Ephone}, \text{Dno}\}$

FD2: $\text{Pno} \rightarrow \{\text{Pname}, \text{Plocation}\}$

FD3: $\text{Emp_ssn}, \text{Pno} \rightarrow \{\text{Esal}, \text{Ephone}, \text{Dno}, \text{Pname}, \text{Plocation}\}$

By virtue of FD3, the attribute set $\{\text{Emp_ssn}, \text{Pno}\}$ represents a key of the universal relation. Hence F , the set of given FDs, includes $\{\text{Emp_ssn} \rightarrow \text{Esal}, \text{Ephone}, \text{Dno}; \text{Pno} \rightarrow \text{Pname}, \text{Plocation}; \text{Emp_ssn}, \text{Pno} \rightarrow \text{Esal}, \text{Ephone}, \text{Dno}, \text{Pname}, \text{Plocation}\}$.

By applying the minimal cover Algorithm 15.2, in step 3 we see that Pno is an extraneous attribute in $\text{Emp_ssn}, \text{Pno} \rightarrow \text{Esal}, \text{Ephone}, \text{Dno}$. Moreover, Emp_ssn is extraneous in $\text{Emp_ssn}, \text{Pno} \rightarrow \text{Pname}, \text{Plocation}$. Hence the minimal cover consists of FD1 and FD2 only (FD3 being completely redundant) as follows (if we group attributes with the same left-hand side into one FD):

Minimal cover $G: \{\text{Emp_ssn} \rightarrow \text{Esal}, \text{Ephone}, \text{Dno}; \text{Pno} \rightarrow \text{Pname}, \text{Plocation}\}$

The second step of Algorithm 15.4 produces relations R_1 and R_2 as:

$R_1 (\text{Emp_ssn}, \text{Esal}, \text{Ephone}, \text{Dno})$

$R_2 (\text{Pno}, \text{Pname}, \text{Plocation})$

In step 3, we generate a relation corresponding to the key $\{\text{Emp_ssn}, \text{Pno}\}$ of U . Hence, the resulting design contains:

$R_1 (\text{Emp_ssn}, \text{Esal}, \text{Ephone}, \text{Dno})$

$R_2 (\text{Pno}, \text{Pname}, \text{Plocation})$

$R_3 (\text{Emp_ssn}, \text{Pno})$

This design achieves both the desirable properties of dependency preservation and nonadditive join.

15.3.2 Nonadditive Join Decomposition into BCNF Schemas

The next algorithm decomposes a universal relation schema $R = \{A_1, A_2, \dots, A_n\}$ into a decomposition $D = \{R_1, R_2, \dots, R_m\}$ such that each R_i is in BCNF and the decomposition D has the lossless join property with respect to F . Algorithm 15.5 utilizes property NJB and claim 2 (preservation of nonadditivity in successive decompositions) to create a nonadditive join decomposition $D = \{R_1, R_2, \dots, R_m\}$ of a universal relation R based on a set of functional dependencies F , such that each R_i in D is in BCNF.

Algorithm 15.5. Relational Decomposition into BCNF with Nonadditive Join Property

Input: A universal relation R and a set of functional dependencies F on the attributes of R .

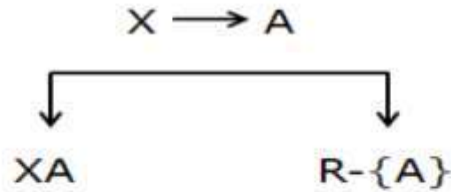
1. Set $D := \{R\}$;
2. While there is a relation schema Q in D that is not in BCNF do
{
 choose a relation schema Q in D that is not in BCNF;
 find a functional dependency $X \rightarrow Y$ in Q that violates BCNF;
 replace Q in D by two relation schemas $(Q - Y)$ and $(X \cup Y)$;
}

BCNF algorithm:

- It is used to decompose any given relation to BCNF directly.
- This algorithm gives guarantee for: Final BCNF decomposition.
Lossless decomposition (Final BCNF decomposition will always be Lossless)
- **Note:** This algorithm fails to give guarantee for dependency preservation.

Steps:

1. Identify the dependencies which violates the BCNF definition and consider that as $X \rightarrow A$
2. Decompose the relation R into XA & $R - \{A\}$ (R minus A).
3. Validate if both the decomposition are in BCNF or not. If not re-apply the algorithm on the decomposition that is not in BCNF.



All the decomposition resulted by this algorithm would be in BCNF and they would be lossless however some of the decomposition will preserved the dependencies and rest not.

BCNF is not a preferred normal form as it does not guarantee for dependency preservation.

Example: $R(A,B,C,D,E) \{ AB \rightarrow CD, D \rightarrow E, A \rightarrow C, B \rightarrow D \}$
Find the BCNF decomposition of the above relation.

Solution: Given Relation is $R(A,B,C,D,E)$ with dependencies as $\{AB \rightarrow CD, D \rightarrow E, A \rightarrow C, B \rightarrow D\}$
Candidate Key for this relation is AB.

Hence Prime Attributes: A, B. Non Prime Attributes: C, D, E.

using the dependency definition:

$AB \rightarrow CD$ (Full Dependency – CD is dependent on candidate key)

$D \rightarrow E$ (Transitive Dependency : non prime derives non prime)

$A \rightarrow C$ (Partial Dependency : Prime derives non prime)

$B \rightarrow D$ (Partial Dependency : Prime derives non prime)

Hence the dependency which violates BCNF are $D \rightarrow E, A \rightarrow C, B \rightarrow D$.

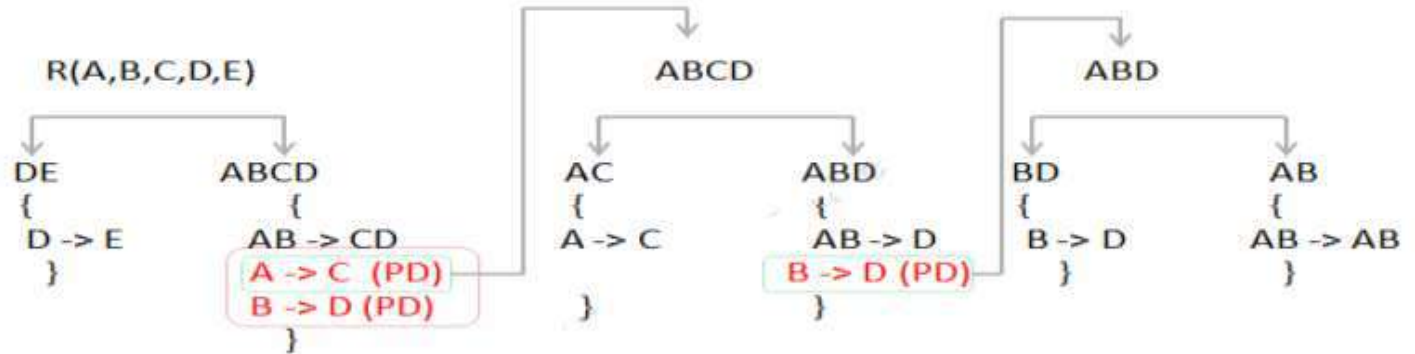
so will take one by one the dependencies which violates the BCNF definition.

so will take $D \rightarrow E$ first as $X \rightarrow 'A'$ {not the A listed in relation as attributes}

So $X = D$ & $'A' = E$.

X'A' will be DE and R-{A'} will be ABCD

R(A, B, C, D, E)
 Functional Dependencies {AB → CD, D → E, A → C, B → D}



Final BCNF Decomposition will be:
 D(DE{D → E} | AC{A → C} | BD{B → D} | AB{AB → AB})

*PD : Partial Dependency

in above decomposition $AB \rightarrow CD$ is missing which is in the original relation. however we can derive $AB \rightarrow CD$ as below:

$AB \rightarrow AB \Rightarrow AB \rightarrow A$ and $AB \rightarrow B$

now $AB \rightarrow A$ and $A \rightarrow C \Rightarrow AB \rightarrow C$ ——— 1

also $AB \rightarrow B$ and $B \rightarrow D \Rightarrow AB \rightarrow D$ ——— 2

combining 1 & 2 $\Rightarrow AB \rightarrow CD$

hence we can derive missing $AB \rightarrow CD$ from the decomposed relation. hence in the decomposition dependency also preserved.

All BCNF decomposition guarantees Lossless decomposition, hence above decomposition is also lossless.

SUMMARY

- Normalization is the process of organizing data in a database to reduce data redundancy and improve data integrity.
- It involves breaking down larger tables into smaller, more focused tables that are related by common attributes.
- This is accomplished through a series of normal forms, each of which has specific requirements that must be met in order to be considered fully normalized. Normalization eliminates redundant data, helps prevent data anomalies and inconsistencies, and makes databases more efficient, accurate, and easy to manage.
- The most commonly used normal forms are first normal form (1NF), second normal form (2NF), and third normal form (3NF), with higher levels of normalization available for more complex databases.

SELF-ASSESSMENT QUESTIONS

1. Which of the following is a benefit of normalization in database design?

- (a) Reduced data redundancy**
- (b) Decreased data consistency
- (c) Improved data integrity
- (d) Reduced storage space

2. Which normal form requires that all non-key columns in a table be dependent only on the primary key and not on other non-key columns?

- (a) First Normal Form (1NF)
- (b) Second Normal Form (2NF)**
- (c) Third Normal Form (3NF)
- (d) Fourth Normal Form (4NF)

- 1. Define normalization and why it is required.**
- 2. List out various normalization forms.**
- 3. Analyze various relations to identify its normal form.**
- 4. Summarize partial functional and transitive dependency.**

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan
2. Fundamentals of Database Systems by Ramez Elmasri and Shamkant B. Navathe

THANK YOU



Team – DBMS