

Department of CSE

COURSE NAME: DBMS

COURSE CODE:23AD2102R

Topic: Normalization –Normal forms, 1NF,2NF

Session - 7

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#}.

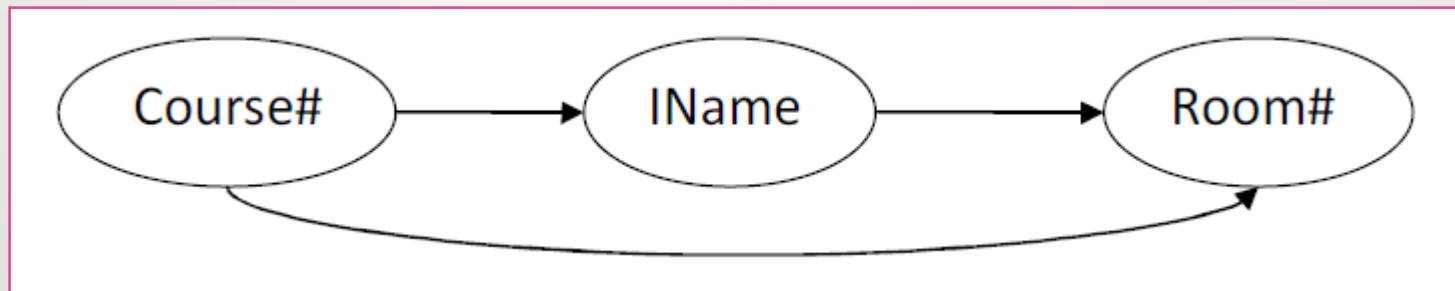
Partial Functional Dependency – In a given relation R, X and Y are attributes. **Attribute Y is partially dependent on attribute X only if it is dependent on sub-set of attribute X where X is composite in nature.**

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.



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: $AB \rightarrow A$

$AB \rightarrow B$

$AB \rightarrow AB$

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

$A \rightarrow BC$

$BC \rightarrow DE$

$D \rightarrow F$

$CF \rightarrow G$

Given a Relation $R=(A,B,C,D,E)$ and Functional Dependencies are: $F=\{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E\}$ Determine all Candidate keys of R and the highest normal form of R with justify your Answer?

A **candidate key** is a minimal superkey that can uniquely identify all attributes in the relation. To find candidate keys, we use the **attribute closure** method to check which combinations of attributes can determine all attributes in R.

Functional Dependencies:


1. $BC \rightarrow D$
2. $AC \rightarrow BE$
3. $B \rightarrow E$

Attribute closure test:


- Step 1.1: Identify all attributes in $R=\{A,B,C,D,E\}$
- Step 1.2: Start with minimal sets of attributes.

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).

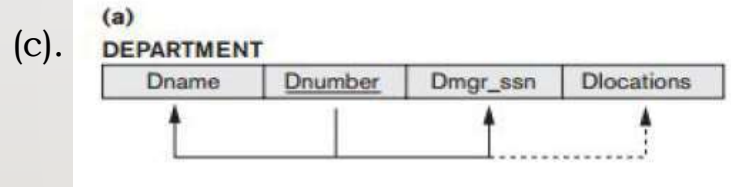


Figure (a)
A relation schema

(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**

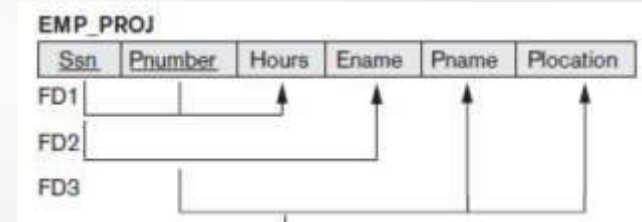
Second Normal Form (2NF)

- 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

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



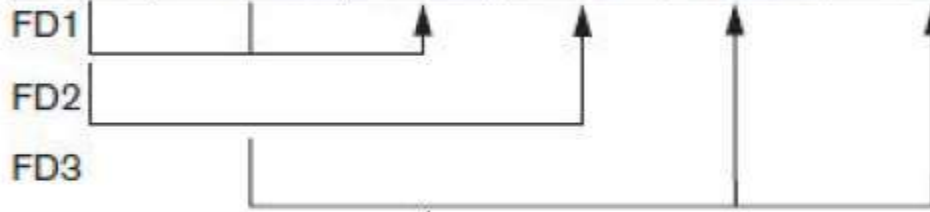
- It is clear from these functional dependencies that the table has partial dependencies and **it is not in 2NF**

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

- 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.

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