Database Systems: Functional Dependency

In this Lecture you will Learn about:

1. **Prime & Non-Prime Attributes in RDBMS**
2. **Functional Dependency in RDBMS**
3. **Functional Dependency Types**
4. **Finding Super Keys and Candidate Keys**

Prime or Key Attributes

Prime attributes are attributes that are part of any candidate key or the primary key of a relation.

These attributes are used to uniquely identify tuples (rows) in the relation.

In other words, prime attributes directly contribute to the identification of each tuple in the relation.

Non Prime or Non Key Attributes

Non-prime attributes are attributes that are **not part of any candidate key or the primary key** of a relation.

These attributes are **not used for uniquely identifying** tuples in the relation but **provide additional information** about the entities represented by the relation.

Non-prime attributes are **dependent on the primary key or candidate keys** for their values.

Functional Dependency

Functional dependencies (FDs) define the **relationship between attributes in a table**.

They specify how **one attribute or set of attributes can determine the value of another** attribute in the same table.

In other words, knowing the value of the first attribute(s) allows you to uniquely determine the value of the second attribute(s).

Functional Dependency

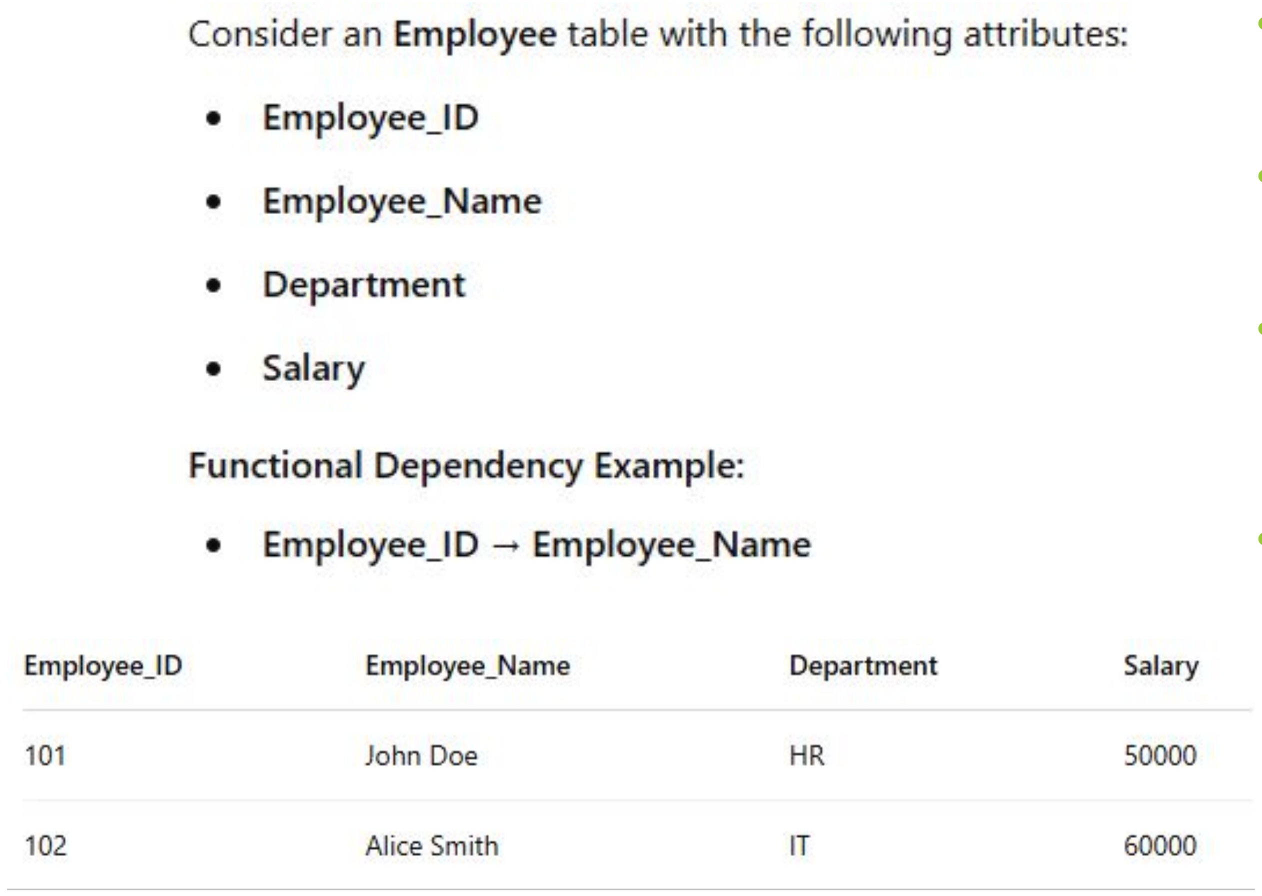
A dependency is a constraint that governs or defines the relationship between two or more attributes. Also know as Functional Dependency.

In a database, it happens when information recorded in the same table uniquely determines other information stored in the same table.

This may also be described as a relationship in which knowing the value of one attribute (or collection of attributes) in the same table tells you the value of another attribute (or set of attributes).

It's critical to understand database dependencies since they serve as the foundation for database normalization.

Functional Dependency



Consider an **Employee** table with the following attributes:

- **Employee_ID**
- **Employee_Name**
- **Department**
- **Salary**

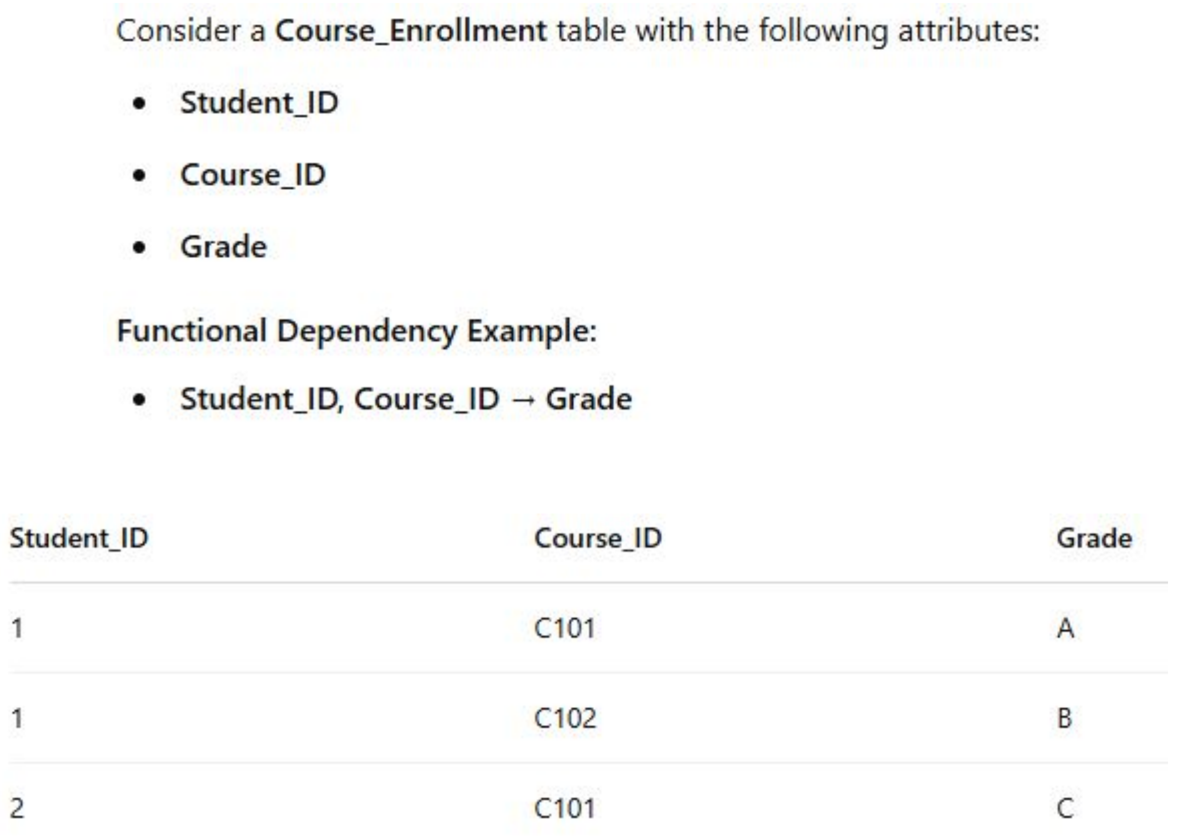
Functional Dependency Example:

- **Employee_ID → Employee_Name**

Employee_ID	Employee_Name	Department	Salary
101	John Doe	HR	50000
102	Alice Smith	IT	60000

- The Employee_ID uniquely determines the Employee_Name.
- If you know the Employee_ID, you can find the corresponding Employee_Name.
- For instance, if Employee_ID = 101, you can be sure that the Employee_Name is "John Doe" (assuming unique Employee_IDs).
- Here, **Employee_ID** functionally determines **Employee_Name**.

Functional Dependency



Consider a **Course_Enrollment** table with the following attributes:

- Student_ID
- Course_ID
- Grade

Functional Dependency Example:

- Student_ID, Course_ID → Grade

Student_ID	Course_ID	Grade
1	C101	A
1	C102	B
2	C101	C

- The combination of **Student_ID** and **Course_ID** uniquely determines the **Grade** of the student in that course.
- If you know the **Student_ID** and **Course_ID**, you can uniquely determine the **Grade**.
- For Student_ID = 1 and Course_ID = C101, you can say that the Grade is A.
- This is a functional dependency where **Student_ID and Course_ID** together determine Grade.

Functional Dependency

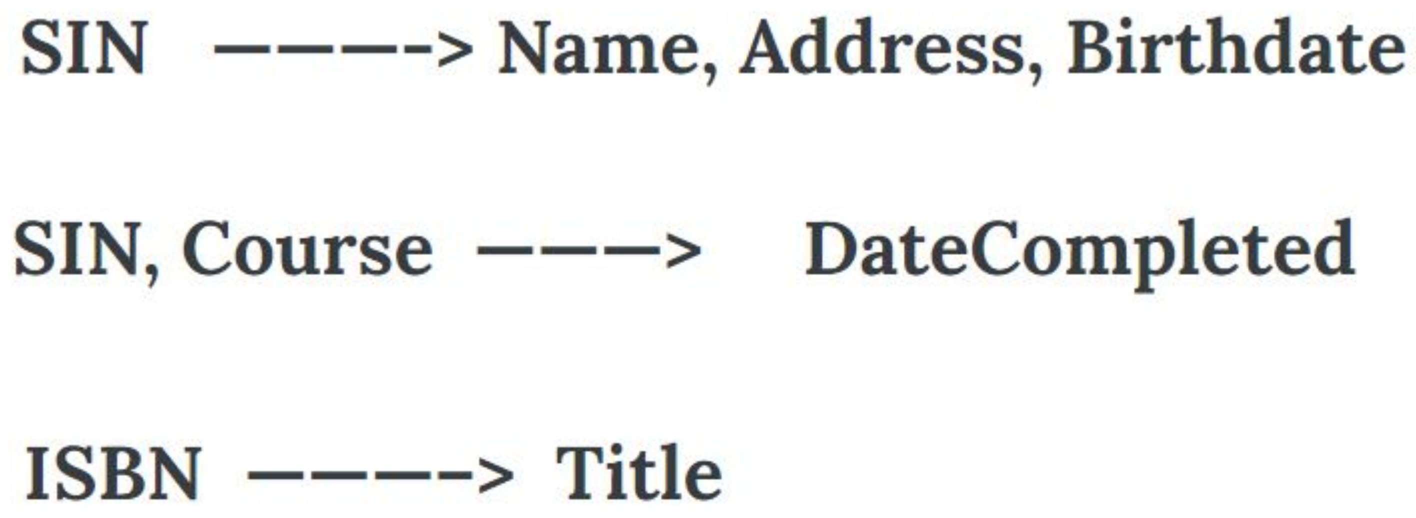
- In a relational database management, functional dependency is a concept that specifies the relationship between two sets of attributes where one attribute determines the value of another attribute.
- It is denoted as **X** → **Y,** where the attribute set on the left side of the arrow, **X** is called **Determinant**, and **Y** is called the **Dependent**.
- For example, if we have a table with attributes "A," "B," and "C," and attribute "A" determines the values of attributes "B" and "C," you would denote it as
- **A -> B, C**
- This notation indicates that the value(s) in attribute "A" determines the value(s) in attributes "B" and "C."
- In other words, if you know the value of "A," you can determine the values of "B" and "C."

Functional Dependency

If the determinant X value is unique (different) then the dependent Y could have any value meaning:

- for same X , value of Y should be same.
- for different X, value of Y could be same or different.
- If a table contains an attribute as a primary key, then all other attributes in that table must be fully functionally dependent on the primary key.
- If X–>Y holds true then Y–>X may or may not hold.

Functional Dependency



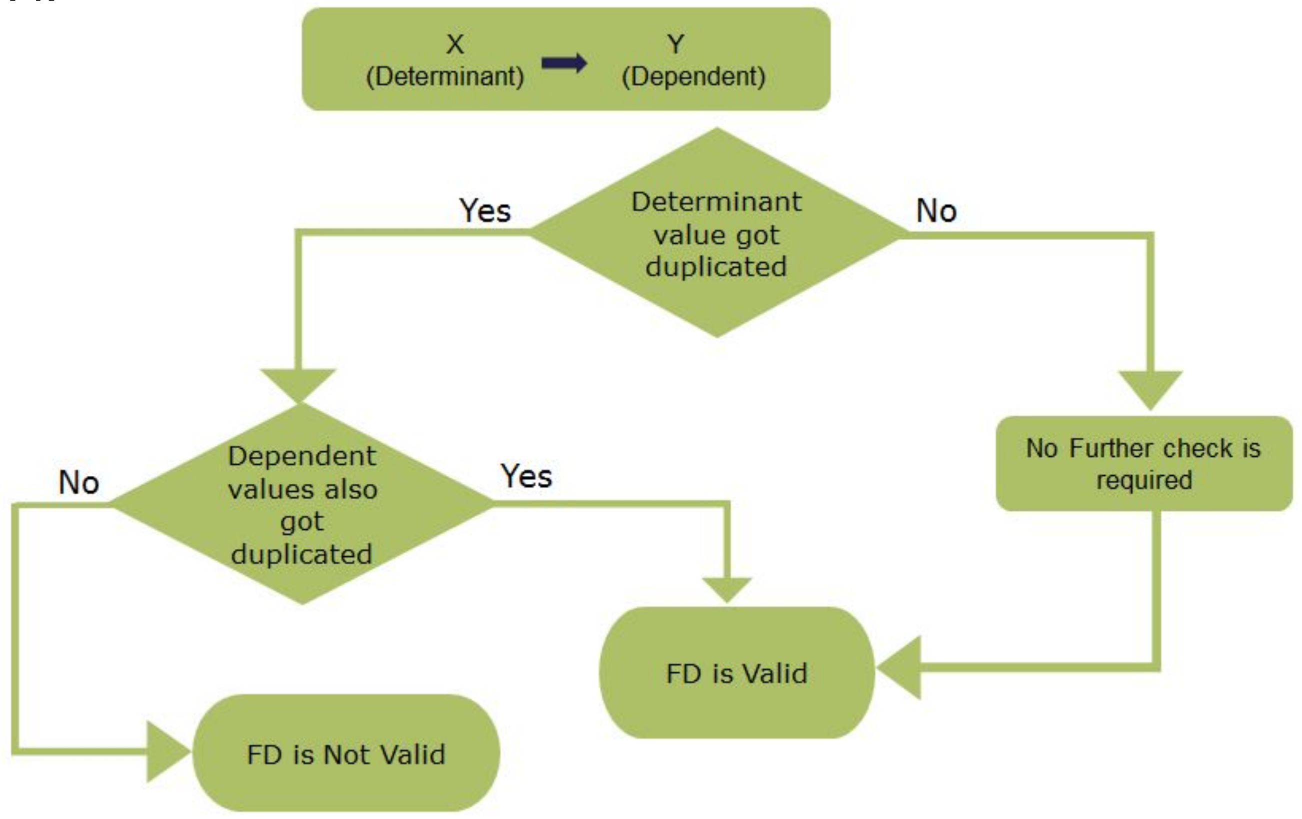
SIN ————> Name, Address, Birthdate

SIN, Course ———> DateCompleted

ISBN ————> Title

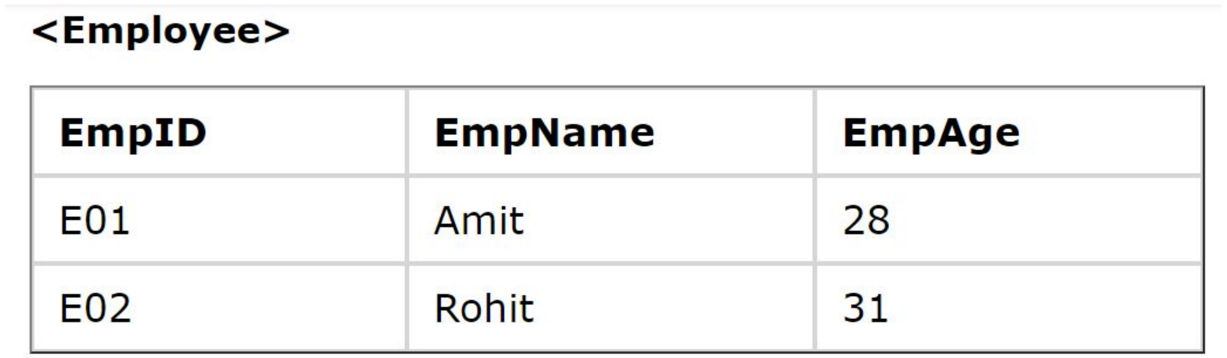
Functional Dependency

- Flow chart used for F.D validation check:


X (Determinant) ➡ Y (Dependent)
Determinant value got duplicated
Yes
No
Dependent values also got duplicated
No
Yes
No Further check is required
FD is Valid
FD is Not Valid

Full Functional Dependency

- When all non-key attributes in a relation depend on the key attribute (simple or composite) then this is called full functional dependency.


<Employee>

EmpID	EmpName	EmpAge
E01	Amit	28
E02	Rohit	31

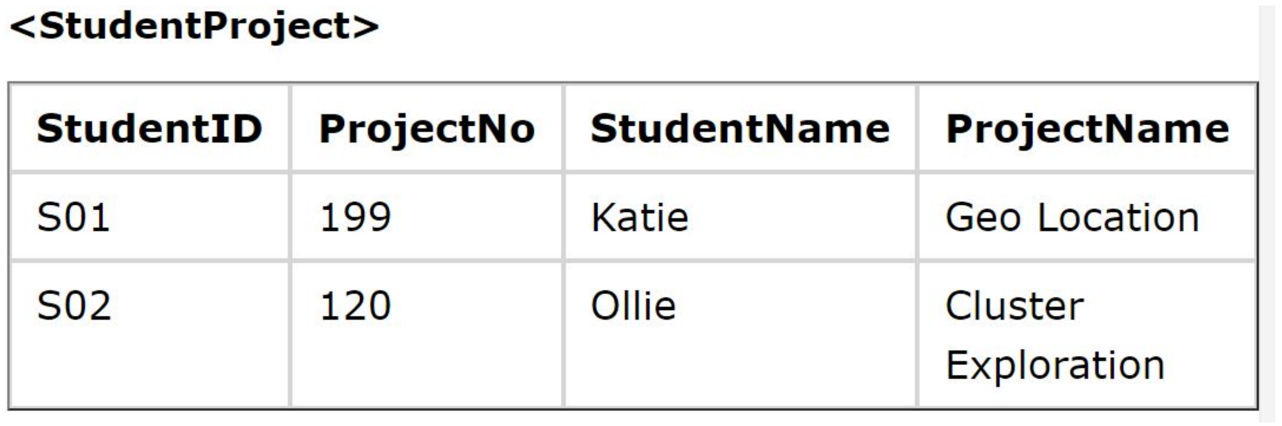
- In the above table, EmpName is functionally dependent on EmpID because EmpName can take only one value for the given value of EmpID:

Partial Functional Dependency

- A functional dependency X->Y is a partial dependency if Y is functionally dependent on X and Y can be determined by any proper subset of X. For example, we have a relationship AC->B, A->D, and D->B. Here A is alone capable of determining B, which means B is partially dependent on AC.
- **Partial dependency** implies is a situation where a non-prime attribute(An attribute that does not form part of the determinant(Primary key/Candidate key)) is functionally dependent to a portion/part of a primary key/Candidate key.

Partial Functional Dependency

- Example



<StudentProject>

StudentID	ProjectNo	StudentName	ProjectName
S01	199	Katie	Geo Location
S02	120	Ollie	Cluster Exploration

- The prime key attributes are **StudentID** and **ProjectNo.**
- As stated, the non-prime attributes i.e. **StudentName** and **ProjectName** should be functionally dependent on part of a candidate key, to be Partial Dependent.
- The **StudentName** can be determined by **StudentID** that makes the relation Partial Dependent.
- The **ProjectName** can be determined by **ProjectID**, which that the relation Partial Dependent.

Transitive Functional Dependency

- When an indirect relationship causes functional dependency it is called Transitive Dependency.
- If P -> Q and Q -> R is true, then P-> R is a transitive dependency.
- A transitive functional dependency in DBMS is a relational database table's relationship between attributes (columns). It occurs when one attribute's value determines another's value through an intermediary (a third) attribute.

Transitive Functional Dependency

- Consider a database table called "Student_Info" with the following attributes:

Student_ID (unique identifier for each student),

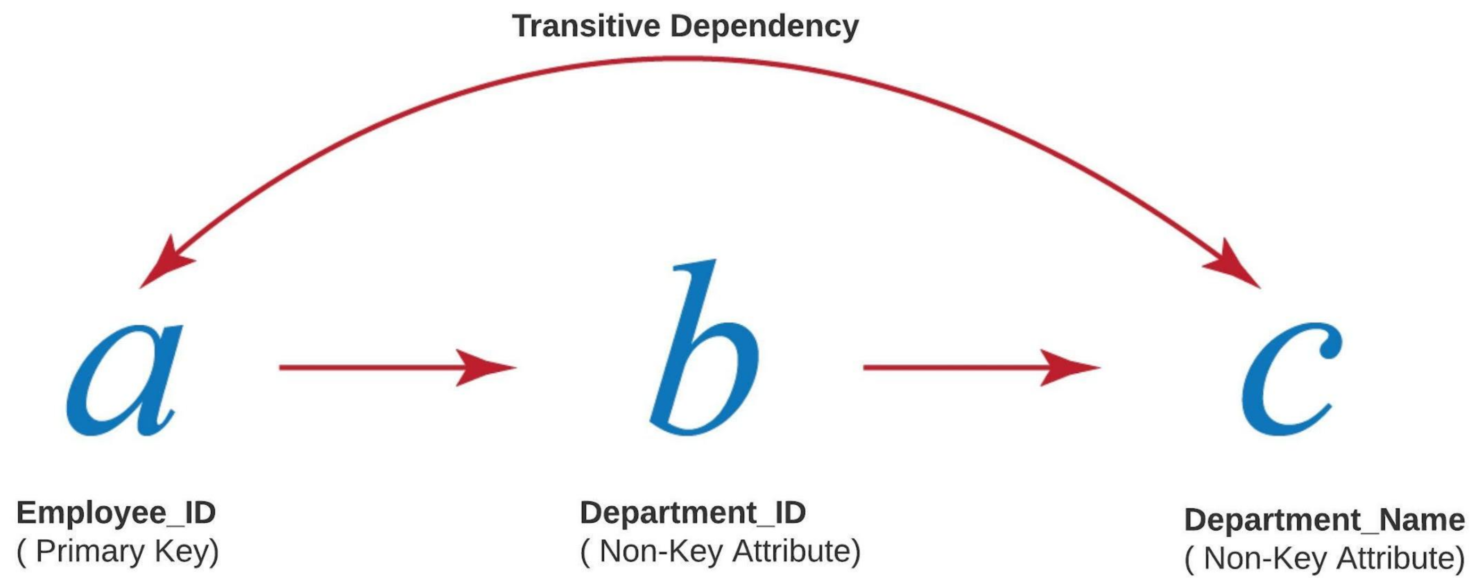
Student_Name,

Student_Address

Student_City.

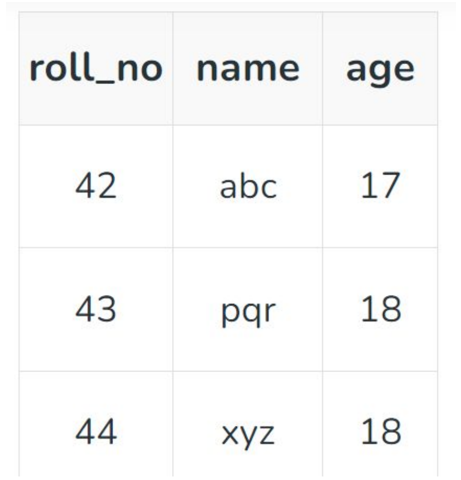
- In this example, we can assume that Student_Address depends on Student_City, and Student_City depends on Student_ID. This creates a transitive dependency in DBMS, where Student_ID indirectly determines Student_Address.

Transitive Functional Dependency



Trivial Functional Dependency

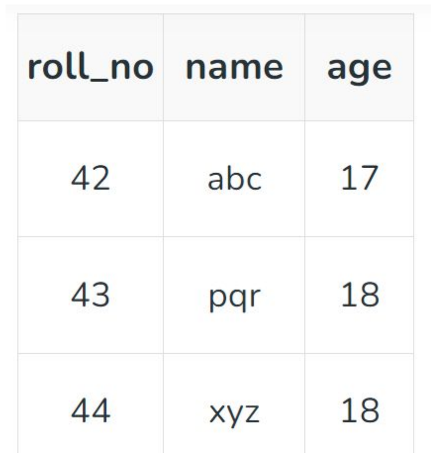
- In Trivial Functional Dependency, a dependent is always a subset of the determinant. i.e. If X → Y and Y is the subset of X, then it is called trivial functional dependency.
- Here, **{roll_no, name}** → **name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}.** Similarly, **roll_no** → **roll_no** is also an example of trivial functional dependency.



roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Non-Trivial Functional Dependency

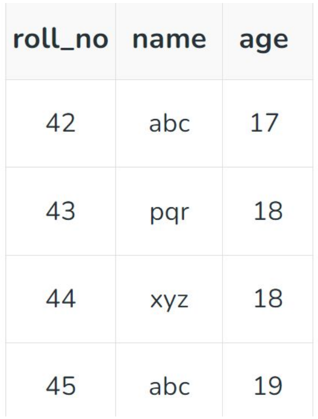
- In **Non-trivial functional dependency**, the dependent is strictly not a subset of the determinant. i.e. If **X** → **Y** and **Y is not a subset of X**, then it is called Non-trivial functional dependency.
- Here, **roll_no → name** is a non-trivial functional dependency, since the dependent **name** is **not a subset of** determinant **roll_no.** Similarly, **{roll_no, name}** → **age** is also a non-trivial functional dependency, since **age** is **not a subset of {roll_no, name}**



roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Multi-Valued Functional Dependency

- In Multivalued functional dependency, entities of the dependent set are not dependent on each other. i.e. **If a → {b, c}** and there exists no functional dependency between b and c, then it is called a multivalued functional dependency.
- Here, roll_no → {name, age} is a multivalued functional dependency, since the dependents name & age are not dependent on each other(i.e. name → age or age → name doesn't exist !)

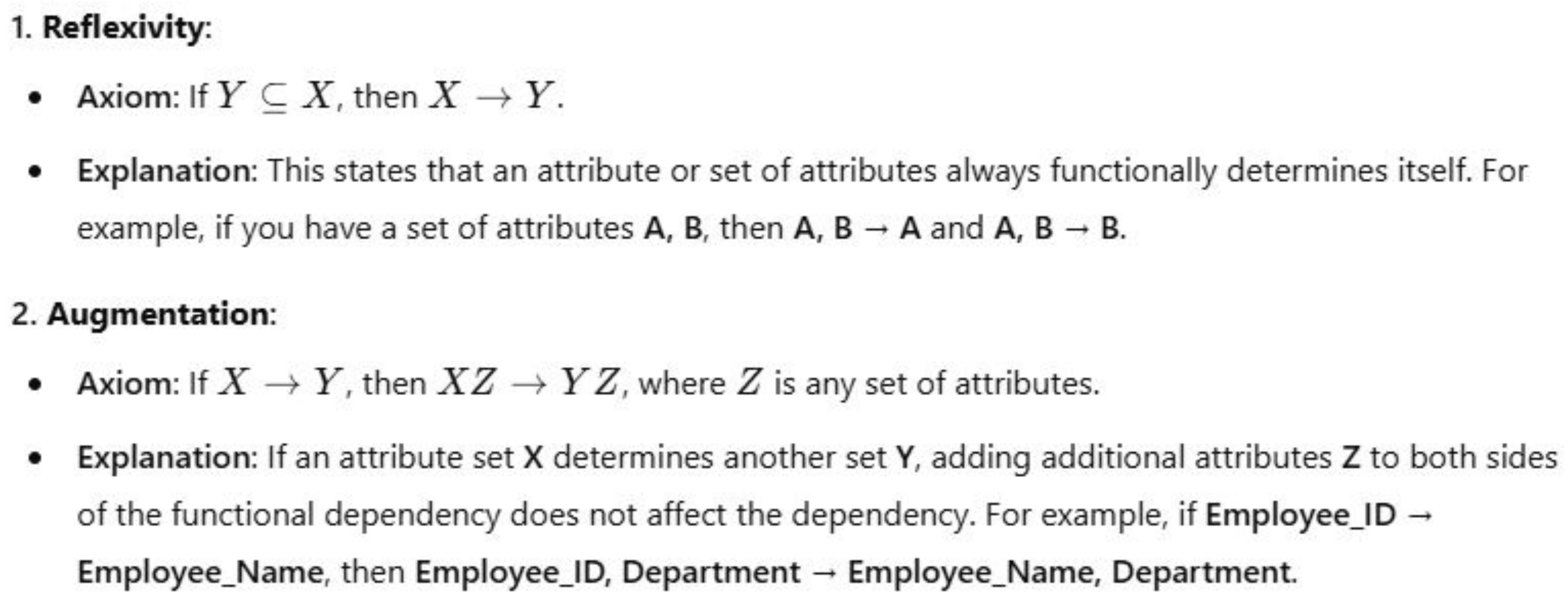


roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Armstrong’s axioms/properties of functional dependencies

- Armstrong's axioms are a set of inference rules in relational databases.
- They help derive additional functional dependencies from a given set.
- William W. Armstrong formulated these axioms.
- They provide a formal basis for reasoning about dependencies in a schema.
- Useful for understanding relationships between attributes.
- Integral in the process of database normalization and optimization.

Armstrong’s axioms/properties of functional dependencies



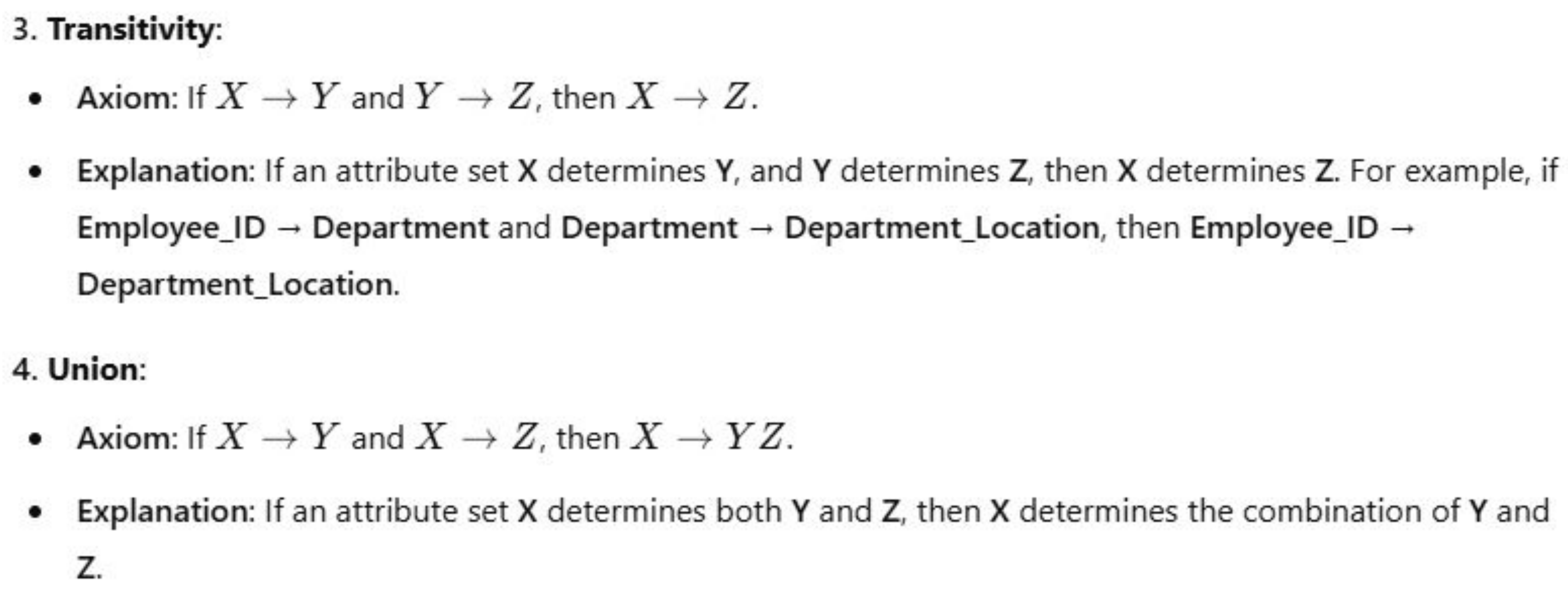
1. **Reflexivity**:

- **Axiom:** If $Y \subseteq X$, then $X \to Y$.
- **Explanation:** This states that an attribute or set of attributes always functionally determines itself. For example, if you have a set of attributes **A, B**, then **A, B → A** and **A, B → B**.

2. **Augmentation**:

- **Axiom:** If $X \to Y$, then $XZ \to YZ$, where Z is any set of attributes.
- **Explanation:** If an attribute set **X** determines another set **Y**, adding additional attributes **Z** to both sides of the functional dependency does not affect the dependency. For example, if **Employee_ID → Employee_Name**, then **Employee_ID, Department → Employee_Name, Department**.

Armstrong’s axioms/properties of functional dependencies



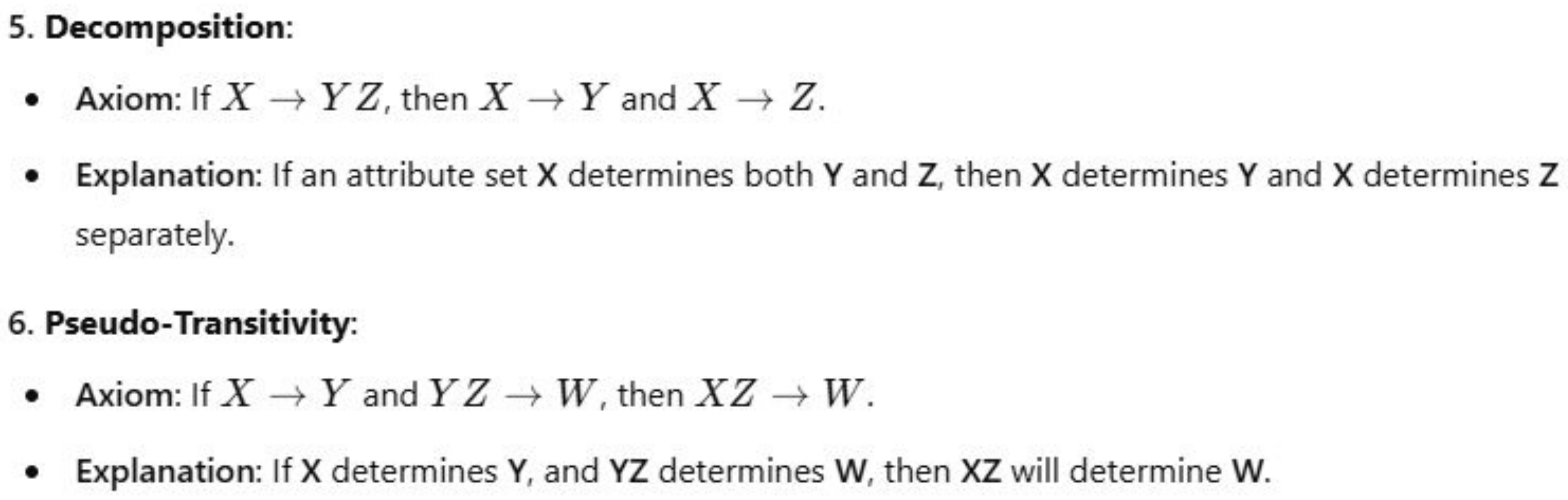
3. **Transitivity**:

- **Axiom**: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.
- **Explanation**: If an attribute set **X** determines **Y**, and **Y** determines **Z**, then **X** determines **Z**. For example, if **Employee_ID → Department** and **Department → Department_Location**, then **Employee_ID → Department_Location.**

4. **Union**:

- **Axiom**: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
- **Explanation**: If an attribute set **X** determines both **Y** and **Z**, then **X** determines the combination of **Y** and **Z.**

Armstrong’s axioms/properties of functional dependencies



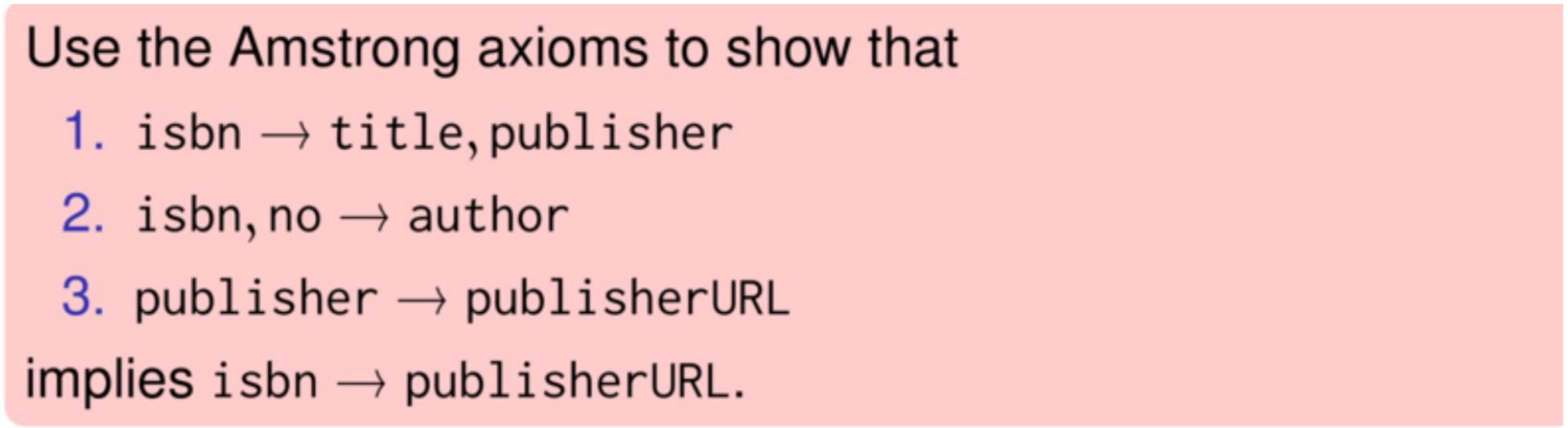
5. **Decomposition**:

- **Axiom**: If $X \to YZ$, then $X \to Y$ and $X \to Z$.
- **Explanation**: If an attribute set **X** determines both **Y** and **Z**, then **X** determines **Y** and **X** determines **Z** separately.

6. **Pseudo-Transitivity**:

- **Axiom**: If $X \to Y$ and $YZ \to W$, then $XZ \to W$.
- **Explanation**: If **X** determines **Y**, and **YZ** determines **W**, then **XZ** will determine **W**.

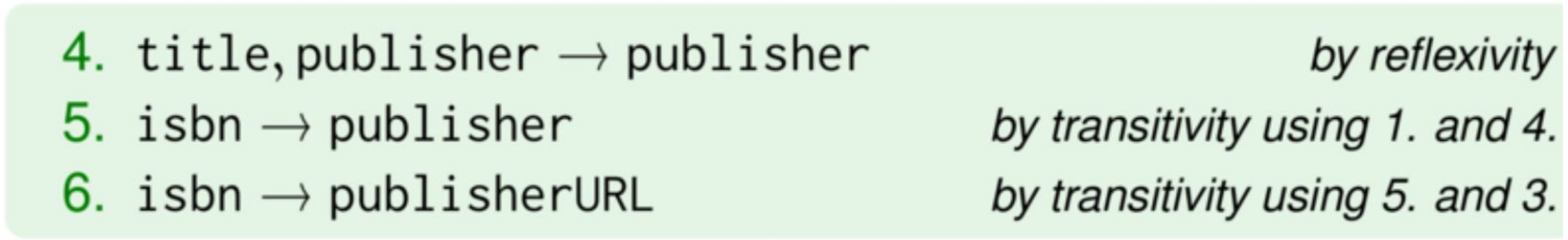
Rules of Axioms



Use the Amstrong axioms to show that

1. isbn → title, publisher
2. isbn, no → author
3. publisher → publisherURL

implies isbn → publisherURL.



4. title, publisher → publisher *by reflexivity*
5. isbn → publisher *by transitivity using 1. and 4.*
6. isbn → publisherURL *by transitivity using 5. and 3.*

Task

- Draw tables and illustrate FD such a full , partial, transitive, trivial and non trivial.
- Give examples of axioms and proof with example rules of axioms from given table.

Roll No	Name	Marks	Dept	Course
1	Ali	70	CS	C1
2	Sana	68	EE	C1
…	…	…	…	…