Unit-3

Cont…

Functional dependency

- A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets.
- A functional dependency is denoted by an arrow "→". The functional dependency X → Y, where X is a set of attributes that is capable of determining the value of Y.
- The attribute set on the left side of the arrow, X is called Determinant, while on the right side, Y is called the Dependent.
- Functional dependencies are used to mathematically express relations among database entities and are very important to understand advanced concepts in Relational Database System

roll_no	name	dept_name	dept_building
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

- From the table we can conclude some valid functional dependencies:

roll_no → { name, dept_name, dept_building },
Here, roll_no can determine values of fields name, dept_name and dept_building,

roll_no → dept_name , Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.

dept_name → dept_building , Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building

More valid functional dependencies: roll_no → name, {roll_no, name} ⇢ {dept_name, dept_building}, etc.

Armstrong's axioms/properties of functional dependencies:

- Reflexivity: If Y is a subset of X, then X→Y holds by reflexivity rule
  - For example, {roll_no, name} → name is valid.
- Augmentation: If X → Y is a valid dependency, then XZ → YZ is also valid by the augmentation rule.
  - For example, If {roll_no, name} → dept_building is valid, hence {roll_no, name, dept_name} → {dept_building, dept_name} is also valid.→
- Transitivity: If X → Y and Y → Z are both valid dependencies, then X→Z is also valid by the Transitivity rule.
  - For example, roll_no → dept_name & dept_name → dept_building, then roll_no → dept_building is also valid.

Types of Functional dependencies

- Trivial functional dependency
- Non-Trivial functional dependency
- Multivalued functional dependency
- Transitive functional dependency

1. 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

2. 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

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

3. Multivalued 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.
- It is represented by ->> Double Arrow
- Ex.
- In the previous table DNumber -> Dname, Dnumber -> Dlocation But there is dependency between Dname and Dlocation, So Dnumber ->> DLocation

4. Transitive Functional Dependency

- In transitive functional dependency, dependent is indirectly dependent on determinant.
- i.e. If a → b & b → c, then according to axiom of transitivity, a → c. This is a transitive functional dependency

- Here, enrol_no → dept and dept → building_no,
- Hence, according to the axiom of transitivity,
- enrol_no → building_no is a valid functional dependency.
- This is an indirect functional dependency, hence called Transitive functional dependency.

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Transitive FD example

Roll_Number	Pin_Code	City_Name
011	450331	Burhanpur
012	450001	Khandwa
013	456001	Ujjain
014	452020	Indore

From above table,
Roll_Number --> Pin_Code and Pin_Code --> City_Name hold.
Than Roll_Number --> City_Name is a transitive FD.

- Normalization:
  - It is the process of decomposing unsatisfactory "bad“ relations by breaking up their attributes into smaller relations
  - Normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- Normal form:
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

Definitions of Keys and Attributes Participating in Keys

- A super key of a relation schema R = {A1, A2, ..., An} is a set of attributes S subset-of R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]
- A Candidate key K is a super key with the additional property that removal of any attribute from K will cause K not to be a super key any more.

OR

- Minimal Super Key with unique and not null values can be a candidate key
- One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary Keys / Alternate Keys

- A Prime attribute must be a member of some candidate key
- A Nonprime attribute is not a prime attribute— that is, it is not a member of any candidate key

Normalization

- First Normal Form: This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.
- Eg.

Car_model	Color	Price
1	Red, Grey, Black	5lk , 6lk , 4.5lk
2	Red, Yellow, White	5.5 lk , 4lk, 6lk
3	Black, White	7lk , 6lk
4	White, Grey	6lk , 7lk

- Convert into 1st NF as:

  {Car_Model, Color, Price} with multiple records for Car Model

Here in 1st NF we don't create any Primary key.

Cont…

- Second Normal Form:
- Every non-prime attribute should be fully functionally dependent on prime key attribute.
- That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.

Eg.

Stud_Proj: {stud_id, name, proj_id, proj_name}

Student: {stud_Id, name, proj_Id,}

Project: {proj_id, proj_name}

Cont…

- Third Normal Form:

  For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency,

  $X \rightarrow A$, then either

  - X is a superkey or
  - A is prime attribute.

StudentID	StudentName	CourseID	CourseName	CourseFee	InstructorName	InstructorPhone
1	Amit	C101	DBMS	5000	Prof. Mehta	9876543210
2	Rina	C102	Python	6000	Prof. Shah	9876543222
3	Kiran	C101	DBMS	5000	Prof. Mehta	9876543210

Step 1: Check 1NF

•All attributes contain atomic values. Already in 1NF.

Step 2: Check 2NF

•Primary Key could be (StudentID, CourseID).

•CourseName, CourseFee, InstructorName, InstructorPhone depend only on CourseID,

•not on StudentID → **partial dependency**.

•So, split into two tables:

StudentID	StudentName	CourseID
1	Amit	C101
2	Rina	C102
3	Kiran	C101

CourseID	CourseName	CourseFee	InstructorName	InstructorPhone
C101	DBMS	5000	Prof. Mehta	9876543210
C102	Python	6000	Prof. Shah	9876543222

Step 3: Check 3NF

- In COURSE table, **InstructorPhone depends on InstructorName**, not directly on CourseID.
- This is a **transitive dependency**.

StudentID	StudentName	CourseID
1	Amit	C101
2	Rina	C102
3	Kiran	C101

Decompose to 3NF

We create a separate **INSTRUCTOR Table**.

CourseID	CourseName	CourseFee	InstructorName
C101	DBMS	5000	Prof. Mehta
C102	Python	6000	Prof. Shah

InstructorName	InstructorPhone
Prof. Mehta	9876543210
Prof. Shah	9876543222