Relational Database Design
UNIT-5

Functional Dependency:

- ➔ 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**.

Example:

Consider a simple example of a table that contains information about students and their courses:

StudentID	StudentName	CourseID	CourseName
1	Alice	C101	Math
2	Bob	C102	Physics
3	Alice	C101	Math

- StudentID functionally determines StudentName (StudentID → StudentName). If you know the StudentID, you can uniquely determine the StudentName.
- CourseID functionally determines CourseName (CourseID → CourseName). If you know the CourseID, you can uniquely determine the CourseName.

Types of Functional Dependency:

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency
5. Fully Functional Dependency
6. Partial Functional Dependency

Faculty : Shubham Upadhyay

1. **Trivial Functional Dependency:**
    - ➔ In Trivial Functional Dependency, a dependent is always a subset of the determinant. i.e. If $X \rightarrow Y$ and Y is the subset of X, then it is called trivial functional dependency.
    - ➔ In other words, $X \rightarrow Y$ if Y is already contained within X. Trivial dependencies do not provide any new information about the database.
    - ➔ **If the set of attributes Y is a subset of the set of attributes X.**

2. **Non Trivial Functional Dependency:**
    - ➔ In Non-trivial functional dependency, the dependent is strictly not a subset of the determinant. i.e. If $X \rightarrow Y$ and Y is not a subset of X, then it is called Non-trivial functional dependency.

3. **Multivalued Functional Dependency:**
    - ➔ In Multivalued functional dependency, entities of the dependent set are not dependent on each other. i.e. If **a** $\rightarrow$ **{b, c}** and there exists **no** functional dependency between **b and c**, then it is called a multivalued functional dependency.

4. **Transitive Functional Dependency:**
    - ➔ A transitive functional dependency is an indirect relationship between attributes in a relational database.
    - ➔ Specifically, if you have a functional dependency $X \rightarrow Y$ and another functional dependency $Y \rightarrow Z$ a transitive dependency implies that $X \rightarrow Z$.

Faculty: Shubhm Upadhyay

Explanation

Consider a relation RRR with attributes AAA, BBB, and CCC:

1. If A→B
2. And B→C
3. Then, A→C is a transitive dependency.

Example

Assume we have a relation Employee with the following attributes:

- EmployeeID (Primary Key)
- DepartmentID
- DepartmentName

The functional dependencies might be:

1. EmployeeID → DepartmentID (An employee belongs to a department)
2. DepartmentID → DepartmentName (Each department has a unique name)

Because of these dependencies, we can infer that:

- EmployeeID → DepartmentName

Here, EmployeeID → DepartmentName is a transitive dependency because DepartmentID is the intermediate attribute linking EmployeeID to DepartmentName.

5. Fully Functional Dependency:

➔ A functional dependency X→Y is said to be fully functional if removing any attribute from X means that the dependency no longer holds. In other words, Y is fully functionally dependent on X if Y is functionally dependent on the whole of X and not on any proper subset of X.

Example: Consider a relation R with attributes {A,B,C}\{A, B, C\}{A,B,C} and the following functional dependency:

- {A,B}→C C{A,B}→C

If C depends on both A and BBB together and not on AAA or BBB alone, then CCC is fully functionally dependent on {A,B}\{A, B\}{A,B}.

6. Partial Functional Dependency:

➔ A functional dependency X→YX is partial if there is a proper subset X′ of X such that X′→YX'. In other words, Y is partially dependent on X if it depends on just a part of X.

Example: Consider a relation R with attributes {A,B,C}\{A, B, C\}{A,B,C} and the following functional dependencies:

- {A,B}→C
- A→C

Here, C is partially dependent on {A,B} because it is already dependent on A alone, which is a subset of {A,B}

Armstrong's Axioms/Inference Rules:

➔ Armstrong's Axioms, also known as inference rules for functional dependencies in relational databases, are a set of rules used to infer all the functional dependencies on a relational database.

➔ They were developed by William W. Armstrong in 1974.

Rules:

1. **Reflexivity:** If A is a set of attributes and B is a subset of A, then A holds B. If B⊆A then A→B.

2. **Augmentation:** If A→B holds and Y is the attribute set, then AY→BY also holds. That is adding attributes to dependencies, does not change the basic dependencies.
3. **Transitivity:** Same as the transitive rule in algebra, if **A→B** holds and **B→C** holds, then **A→C** also holds. **A→B** is called A functionally which determines B.

Secondary Rules

These rules can be derived from the above axioms.

- **Union:** If A→B holds and A→C holds, then A→BC holds. If X→Y and X→Z then X→YZ.
- **Composition:** If A→B and X→Y hold, then AX→BY holds.
- **Decomposition:** If A→BC holds then A→B and A→C hold. If X→YZ then X→Y and X→Z.
- **Pseudo Transitivity:** If A→B holds and BC→D holds, then AC→D holds. If X→Y and YZ→W then XZ→W.
- **Self Determination:** It is similar to the Axiom of Reflexivity, i.e. A→A for any A.

Closure of FD:

➔ closure of functional dependencies refers to the set of all functional dependencies that can be logically inferred from a given set of functional dependencies.

➔ The closure essentially includes all functional dependencies that hold within the database.

➔ This is often denoted as F+ for a set F of functional dependencies.

Steps to Find Closure of Functional Dependencies:

1. Start with the given set of functional dependencies, F.

2. Apply inference rules (such as Armstrong's Axioms) to derive new functional dependencies:
    - **Reflexivity:** If $Y \subseteq X$, then $X \to Y$.
    - **Augmentation:** If $X \to Y$, then $XZ \to YZ$ for any set of attributes Z.
    - **Transitivity:** If $X \to Y$ and $Y \to Z$, then $X \to Z$.
    - **Union:** If $X \to Y$ and $X \to Z$, then $X \to YZ$.
    - **Decomposition:** If $X \to YZ$, then $X \to Y$ and $X \to Z$.
    - **Pseudotransitivity:** If $X \to Y$ and $YZ \to W$, then $XZ \to W$.
3. Iterate this process until no more new functional dependencies can be inferred. The result is the closure $F+$.

Closure of Attributes:

➔ The closure of a set of attributes X(denoted as X^+) with respect to a set of functional dependencies F is the set of all attributes that are functionally determined by X using the functional dependencies in F.

➔ **EXAMPLES:**

Decomposition:

When we divide a table into multiple tables or divide a relation into multiple relations, then this process is termed Decomposition.

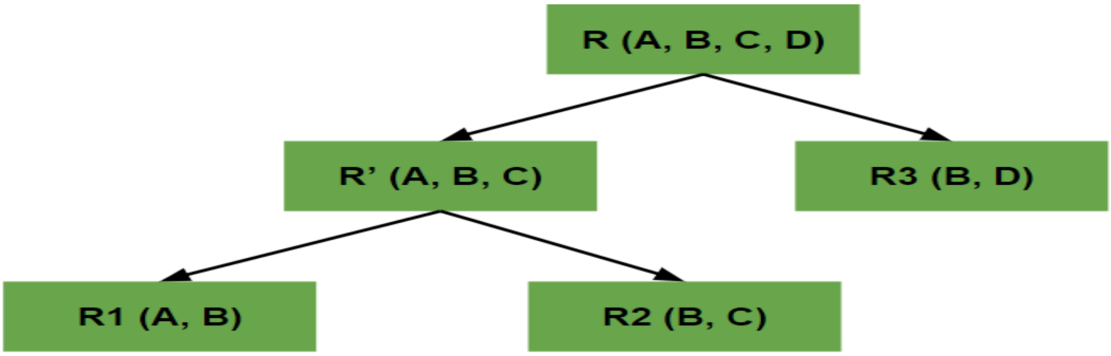
Types of Decomposition

There are two types of Decomposition:

- Lossless Decomposition
- Lossy Decomposition

Lossless Decomposition: The process in which where we can regain the original relation R with the help of joins from the multiple relations formed after decomposition.

Faculty: Shubham Upadhyay



- ➔ The lossless decomposition tries to ensure following things:
- ➔ While regaining the original relation, no information should be lost.
- ➔ If we perform join operation on the sub-divided relations, we must get the original relation.

Example:

There is a relation called R(A, B, C)

A	B	C
55	16	27
48	52	89

Now we decompose this relation into two sub relations R1 and R2

R1(A, B)

A	B
55	16
48	52

R2(B, C)

B	C
16	27
52	89

After performing the Join operation we get the same original relation

A	B	C
55	16	27
48	52	89

Faculty: Shubham Upadhyay

Lossy Decomposition:

- Lossy decomposition means when we perform join operation on the sub-relations it doesn't result to the same relation which was decomposed.
- After the join operation, we always found some extraneous tuples.

Database Anomalies:

- Database anomalies refer to issues that arise in a database when data is inserted, updated, or deleted.
- These anomalies can occur when the database schema is not properly normalized, leading to redundancy, inconsistency, and inefficiency.
- There are three primary types of database anomalies:

1. **Insertion Anomaly**
2. **Update Anomaly**
3. **Deletion Anomaly**

Normalization:

- Normalization is the process of minimizing redundancy from a relation or set of relations.
- Redundancy in relation may cause insertion, deletion, and update anomalies.

Important Points Regarding Normal Forms in DBMS:

1. **First Normal Form (1NF)**
2. **Second Normal Form (2NF)**
3. **Third Normal Form (3NF)**
4. **Boyce-Codd Normal Form (BCNF)**
5. **Fourth Normal Form (4NF)**
6. **Fifth Normal Form (5NF)**

Faculty: Shubham Upadhyay

First Normal Form (1NF):

➔ This is the most basic level of normalization. In 1NF, each table cell should contain only a single value, and each column should have a unique name.

➔ The first normal form helps to eliminate duplicate data and simplify queries.

A table is in 1 NF if:

- There are only Single Valued Attributes.
- Attribute Domain does not change.
- There is a unique name for every Attribute/Column.
- The order in which data is stored does not matter.

Example 1:

Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Second Normal Form (2NF):

➔ A relation that is in First Normal Form.

➔ The second Normal Form (2NF) is based on the concept of fully functional dependency.

➔ **All the non-prime attribute should be fully functional dependent on the candidate key.**

Third Normal Form (3NF):

➔ A relation is in the third normal form, if there is no transitive dependency for non-prime attributes as well as it is in the second normal form.

Faculty: Shubham Upadhyay

➔ A relation is in 3NF if at least one of the following conditions holds in every non-trivial function dependency X –> Y.
  - X is a super key.
  - Y is a prime attribute (each element of Y is part of some candidate key)

Boyce-Codd Normal Form (BCNF):

➔ BCNF is a stricter form of 3NF that ensures that each determinant in a table is a candidate key.
➔ In other words, BCNF ensures that each non-key attribute is dependent only on the candidate key.

Rules for BCNF

Rule 1: The table should be in the 3rd Normal Form.

Rule 2: X should be a super/candidate key for every functional dependency (FD) X−>Y in a given relation.

Fourth Normal Form (4NF):

A relation R is in 4NF if and only if the following conditions are satisfied:

1. It should be in the Boyce-Codd Normal Form (BCNF).

2. The table should not have any Multi-valued Dependency.

Fifth Normal Form/Projected Normal Form (5NF)

A relation R is in 5NF if and only if it satisfies the following conditions:

1. R should be already in 4NF.

2. It cannot be further non loss decomposed (join dependency).