Relational Database Design

Ram Datta Bhatta

Functional Dependency

- ✓ Functional Dependency (FD) is a constraint that determines the **relation of one attribute to another attribute** in the Database Management System(DBMS).
- ✓ It expresses a relationship between attributes, where the value of one attribute (or a set of attributes) uniquely determines the value of another attribute (or set of attributes).
- ✓ A functional dependency is denoted by $X \rightarrow Y$ where X and Y are sets of attributes in a relation. The FD $X \rightarrow Y$ means that if two tuples (rows) in the relation have the same value for X, then they must also have the same value for Y. In other words, X functionally determines Y.

EmpID	EmpName	Salary	City
1	Ram	15000	Pok
2	Shyam	20000	Ktm
3	Gopal	22000	Dharan

Here, EmpID→EmpName

Types of Functional Dependency

1. **Trivial Functional Dependency**:
   - A functional dependency $X \rightarrow Y$ is trivial if $Y \subseteq X$.
   - Example: $\{A, B\} \rightarrow \{A\}$ is trivial because A is a subset of $\{A, B\}$.
2. **Non-Trivial Functional Dependency**:
   - A functional dependency $X \rightarrow Y$ is non-trivial if Y is not a subset of X.
   - Example: $A \rightarrow B$ is non-trivial if B is not a part of A.
3. **Full Functional Dependency**:
   - A functional dependency $X \rightarrow Y$ is a full functional dependency if removing any attribute from X means the dependency does not hold anymore.
   - Example: $\{A, B\} \rightarrow C$ is a full functional dependency if neither $A \rightarrow C$ nor $B \rightarrow C$ holds.

4. **Partial Functional Dependency**:

    - A functional dependency $X \rightarrow Y$ is partial if some attribute can be removed from X and the dependency still holds.
    - Example: In $\{A, B\} \rightarrow C$, if $A \rightarrow C$ holds, then $\{A, B\} \rightarrow C$ is a partial dependency.

5. **Transitive Functional Dependency**:

    - A transitive dependency occurs when there is a set of attributes Z such that $X \rightarrow Z$ and $Z \rightarrow Y$.
    - Example: If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ is a transitive dependency.

Closure properties of functional dependencies

- The Closure Properties of FDs refers to the set of all functional dependencies that can be logically inferred from a given set of FDs using a set of inference rules.
- The closure properties help determine all possible FDs that can be derived from a given set of FDs.
- These properties, or rules, ensure that any FD that logically follows from the original set can be identified.

Closure Properties

1. **Reflexivity (Trivial Functional Dependency):**
   - If $Y \subseteq X$, then $X \rightarrow Y$.
   - Example: If $\{A, B\}$, then $\{A, B\} \rightarrow \{A\}$.
2. **Augmentation (Attribute Augmentation):**
   - If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z.
   - Example: If $\{A\} \rightarrow \{B\}$, then $\{A, C\} \rightarrow \{B, C\}$.
3. **Transitivity (Transitive Rule):**
   - If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.
   - Example: If $\{A\} \rightarrow \{B\}$ and $\{B\} \rightarrow \{C\}$, then $\{A\} \rightarrow \{C\}$.

Closure Properties

4. **Union (Additive Rule)**:
   - If $X \to Y$ and $X \to Z$, then $X \to YZ$.
   - Example: If $\{A\} \to \{B\}$ and $\{A\} \to \{C\}$, then $\{A\} \to \{B, C\}$.
5. **Decomposition (Projective Rule)**:
   - If $X \to YZ$, then $X \to Y$ and $X \to Z$.
   - Example: If $\{A\} \to \{B, C\}$, then $\{A\} \to \{B\}$ and $\{A\} \to \{C\}$.
6. **Pseudotransitivity**:
   - If $X \to Y$ and $WZ \to X$, then $WZ \to Y$.
   - Example: If $\{A\} \to \{B\}$ and $\{C, D\} \to \{A\}$, then $\{C, D\} \to \{B\}$.

Closure of Attributes

- The closure of a set of attributes in a relational database schema is the set of all attributes that can be functionally determined by that set of attributes using a given set of functional dependencies.
- The closure of a set of attributes, denoted as X^+, is the set of all attributes that can be functionally determined by X using a given set of functional dependencies (FDs).
- In other words, it represents all the attributes that are dependent on X directly or indirectly.

Question: Find the Closure of Attributes

$R(A, B, C, D, E)$ and the functional dependencies:

1. $A \to D$
2. $D \to B$
3. $B \to C$
4. $E \to B$

Closure of Attributes

Closure of A (A^+)

1. Start with $A^+ = \{A\}$.
2. Apply $A \to D$: $A^+ = \{A, D\}$.
3. Apply $D \to B$: $A^+ = \{A, D, B\}$.
4. Apply $B \to C$: $A^+ = \{A, D, B, C\}$.

No more FDs can be applied, so the closure of A is $A^+ = \{A, D, B, C\}$.

Closure of D (D^+)

1. Start with $D^+ = \{D\}$.
2. Apply $D \to B$: $D^+ = \{D, B\}$.
3. Apply $B \to C$: $D^+ = \{D, B, C\}$.

No more FDs can be applied, so the closure of D is $D^+ = \{D, B, C\}$.

Closure of B (B^+)

1. Start with $B^+ = \{B\}$.
2. Apply $B \to C$: $B^+ = \{B, C\}$.

No more FDs can be applied, so the closure of B is $B^+ = \{B, C\}$.

Closure of E (E^+)

1. Start with $E^+ = \{E\}$.
2. Apply $E \to B$: $E^+ = \{E, B\}$.
3. Apply $B \to C$: $E^+ = \{E, B, C\}$.

No more FDs can be applied, so the closure of E is $E^+ = \{E, B, C\}$.

Closure of C (C^+)

1. Start with $C^+ = \{C\}$.

No FDs can be applied, so the closure of C is $= \{C\}$.

Closure of AE ($(AE)^+$)

1. Start with $(AE)^+ = \{A, E\}$.
2. Apply $A \to D$: $(AE)^+ = \{A, E, D\}$.
3. Apply $D \to B$: $(AE)^+ = \{A, E, D, B\}$.
4. Apply $E \to B$: B is already in the set.
5. Apply $B \to C$: $(AE)^+ = \{A, E, D, B, C\}$.

No more FDs can be applied, so the closure of AE is $(AE)^+ = \{A, E, D, B, C\}$.

Normalization

- Normalization is a process in database design that decomposes a table into multiple tables to minimize redundancy and eliminate anomalies such as insertion, deletion, and update anomalies.
- The main objective is to:

✓ **Eliminate Duplicate Data:** Reduce storage space and improve data consistency by minimizing data redundancy.

✓ **Avoid Anomalies:** Ensure that changes to data (insertions, deletions, and updates) do not lead to inconsistencies

Anomalies

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatate Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Let’s consider database design

Teacher

tid	tname	address	qualification	subjectid	subname	fm	pm
1	Ram	ktm	PhD	100	C prpgramming	100	32
1	Ram	ktm	PhD	101	C++ programming	100	32
2	Shyam	ktm	BE	102	Java	100	32
3	Sita	brt	BE	102	Java	100	32

What are the Problems in this design?

Solution: Normalization

Option II

Teacher

tid	tname	address	qualification
1	Ram	ktm	PhD
2	Shyam	ktm	BE
3	Sita	brt	BE
4	Gita	ktm	BE

Subject

subjectid	subname	fm	pm
100	C prpgramming	100	32
101	C++ programming	100	32
102	Java	100	32
103	DBMS	100	32

Teaches

tid	subjectid
1	100
1	101
2	102
3	102

Advantages

- **Eliminates Redundancy**: By breaking down tables into smaller, related tables, normalization reduces duplicate data, saving storage space.
- **Prevents Anomalies**: Ensures that updates to data are consistent and do not lead to anomalies such as partial updates or conflicting data entries.
- **Enhances Query Performance**: Optimizes query performance by reducing the amount of data that needs to be scanned and processed, especially for large databases.
- **Supports Data Consistency**: Ensures that the same data is not stored in multiple places, reducing the risk of data discrepancies and promoting consistency.
- **Reduces Storage Costs**: By minimizing redundant data, normalization can lead to reduced storage costs, especially in large databases.
- **Enhances Security**: Enables more precise access control by organizing data into distinct tables, making it easier to restrict access to sensitive information.

Types of Normal Forms

First Normal Form (1NF) :

A relation is in 1NF if it contains an atomic value.

Second Normal Form (2NF):

A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

Third Normal Form (3NF):

A relation will be in 3NF if it is in 2NF and no **transition dependency** exists.

Boyce Code Normal Form (BCNF):

A stronger definition of 3NF is known as Boyce Codd's normal form.

First Normal Form (1 NF)

- ✓ A table is in 1NF if it contains only **atomic values** (indivisible) and there are no repeating groups (values of single type)
- ✓ It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

rollno	name	subjects
1	Ram	Math, Physics
2	Sita	Chemistry, Math
3	Gita	Physics, Math
4	Shyam	Math, Biology

This table is not in 1NF

The **subjects** column contains multiple values, which violates the rule of atomicity in 1NF.
Each cell must hold a single value, not a list or set of values

Converting into 1NF

Creating a separate row for each subject a student is enrolled in. Now, the combination of rollno and subject is the primary key of the table. Here, the design is not perfect, still remains redundancy.

rollno	name	subjects
1	Ram	Math, Physics
2	Sita	Chemistry, Math
3	Gita	Physics, Math
4	Shyam	Math, Biology

rollno	name	subject
1	John	Math
1	John	Physics
2	Alice	Chemistry
2	Alice	Math
3	Bob	Physics
3	Bob	Math

Example: 1 NF

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

Atomic Domain

- An atomic domain means that each value in a column of a table is simple and indivisible (can’t be broken down into smaller parts).
- In following table, the PhoneNumbers column is not atomic because it can contain multiple phone numbers separated by a comma
- To make the domains atomic, we should ensure that each column has only one value per row.

StudentID	Name	PhoneNumbers
11	Alice	0134455555, 014555555
2	Bob	34444555

Second Normal Form (2NF)

A relation is in Second Normal Form (2NF) **if it is in First Normal Form (1NF) and all non-key attributes are fully functionally dependent on the primary key**. If any attribute is only partially dependent on the primary key (i.e., it depends on a part of a composite primary key rather than the whole key), then the table is not in 2NF

rollno	name	subject	roomNo	Building
1	Ram	Math	101	Main
1	Ram	Physics	102	Science
2	Sita	Chemistry	103	Science
2	Sita	Math	101	Main
3	Gita	Physics	102	Science
3	Gita	Math	101	Main
4	Shyam	Math	101	Main
4	Shyam	Biology	104	Science

This table is not in 2NF. Here, **roomNo** is dependent only in **subject** i.e. not whole composite key

Converting into 2NF

To convert the table to 2NF, we need to remove partial dependencies. We can achieve this by creating two tables: one for the **student_subject** relationship and another for the **subject_room** relationship.

Student

rollno	name	subject	roomNo	Building
1	Ram	Math	101	Main
1	Ram	Physics	102	Science
2	Sita	Chemistry	103	Science
2	Sita	Math	101	Main
3	Gita	Physics	102	Science
3	Gita	Math	101	Main
4	Shyam	Math	101	Main
4	Shyam	Biology	104	Science

Student_Subject

rollno	name	subject
1	Ram	Math
1	Ram	Physics
2	Sita	Chemistry
2	Sita	Math
3	Gita	Physics
3	Gita	Math
4	Shyam	Math
4	Shyam	Biology

Subject_room

subject	roomNo	Building
Math	101	Main
Physics	102	Science
Chemistry	103	Science
Biology	104	Science

Alternative Design for 2 NF

Student

rollno	name	subject	roomNo	Building
1	Ram	Math	101	Main
1	Ram	Physics	102	Science
2	Sita	Chemistry	103	Science
2	Sita	Math	101	Main
3	Gita	Physics	102	Science
3	Gita	Math	101	Main
4	Shyam	Math	101	Main
4	Shyam	Biology	104	Science

Student

rollno	name
1	Ram
2	Sita
3	Gita
4	Shyam

Student_Subject

rollno	subject
1	Math
1	Physics
2	Chemistry
2	Math
3	Physics
3	Math
4	Math
4	Biology

Subject_Room

subject	roomNo	Building
Math	101	Main
Physics	102	Science
Chemistry	103	Science
Biology	104	Science

Third Normal Form (3NF)

- A relation is in Third Normal Form (3NF) if **it is in Second Normal Form (2NF) and it has no transitive dependencies.**
- In other words, the non-key attributes must not depend on other non-key attributes i.e., Every non-key attribute must be directly dependent on the primary key.

Subject_Room

<u>subject</u>	roomNo	Building
Math	101	Main
Physics	102	Science
Chemistry	103	Science
Biology	104	Science

This relation is not in 3 NF as roomNo→Building

Converting into 3NF

Subject_Room

subject	roomNo	Building
Math	101	Main
Physics	102	Science
Chemistry	103	Science
Biology	104	Science

Subject_Room

subject	roomNo
Math	101
Physics	102
Chemistry	103
Biology	104

Room_Building

roomNo	building
101	Main
102	Science
103	Science
104	Science

The **building** attribute is dependent on **roomNo**, and **roomNo** is dependent on **subject**. To eliminate this transitive dependency, we need to create a new table that captures the relationship between **roomNo** and **building**

Boyce-Codd Normal Form (BCNF)

- A relation is in Boyce-Codd Normal Form (BCNF) if it is in Third Normal Form (3NF) and, for every functional dependency $X \rightarrow Y$, X is a candidate key.
- In other words, for every functional dependency, the left-hand side (LHS) must be a candidate key.
- BCNF is the advance version of 3NF. It is stricter than 3NF.

Not in BCNF

The functional dependency **Realease_Year_Month** → **Release_Year** does not satisfy BCNF because **Realease_Year_Month** is not a superkey

Release Year	Popularity_Ranking	MovieName	Realease_Year_Month
2022	1	A	2022-01
2022	2	B	2022-03
2022	3	C	2022-01
2023	1	D	2023-05
2023	2	E	2023-01
2023	3	F	2023-09

The candidate keys: {MovieName}, {Release_Year, Popularity_Ranking}, {Realease_Year_Month, Popularity_Ranking}

The functional dependency is: Realease_Year_Month → Release_Year

It is in BCNF

Release Year	Popularity_Ranking	MovieName	Realease_Month
2022	1	A	1
2022	2	B	3
2022	3	C	1
2023	1	D	5
2023	2	E	1
2023	3	F	9

References

https://www.youtube.com/watch?v=VWnKUKH4tLg

https://www.javatpoint.com/dbms-boyce-codd-normal-form

Thank you.