Module 3

Module:3 Relational Database Design

Database Design – Schema Refinement - Guidelines for Relational Schema – Functional dependencies - Axioms on Functional Dependencies- Normalization: First, Second and Third Normal Forms - Boyce Codd Normal Form, Multi-valued dependency and Fourth Normal form - Join dependency and Fifth Normal form

- **Reference :**
- ***R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7th Edition, 2016***
- ***A. Silberschatz, H. F. Korth & S. Sudarshan, Database System Concepts, McGraw Hill, 7th Edition 2019.***

What is Functional Dependency in DBMS?

- A functional dependency is a relationship between two sets of attributes in a relational database.
- if a set of attributes X can uniquely determine another set of attributes Y,

  We say that Y is functionally dependent on X.

  This is denoted as X → Y.
- **Example:**

  consider a Student table with the following attributes

Student(StudentID, StudentName, Course, Instructor)

If each course is taught by **only one** instructor, then the attribute Course can determine the attribute Instructor.

Course→Instructor

This means if we know the course, we can determine the instructor uniquely.

Functional Dependency

In any relation, a functional dependency $\alpha \rightarrow \beta$ **holds** if-
Two tuples having **same value** of **attribute α** also have **same value** for attribute **β.**

Mathematically,

If α and β are the two sets of attributes in a relational table R where,

$$\alpha \subseteq R$$
$$\beta \subseteq R$$

Then, for a functional dependency to exist from α to β,

$$\text{If } t1[\alpha] = t2[\alpha], \text{ then } t1[\beta] = t2[\beta]$$

α	**β**
t1[α]	t1[β]
t2[α]	t2[β]
…….	…….

$\mathbf{f_d : \alpha \rightarrow \beta}$

Hold in FD

In the context of **functional dependencies (FDs)** in **relational database theory**, the terms **"holds"** and **"satisfies"** have specific meanings related to how a relation (i.e., a table) conforms to a given functional dependency.

"Holds"

A functional dependency **$X \rightarrow Y$ holds** in a relation **R** if **for every pair of tuples t1 and t2 in R**, whenever:

$t1[X] = t2[X]$

it follows that:

$t1[Y] = t2[Y]$

In simpler terms: **The FD is always true** for the data in that relation.

Example:

Consider a relation Students(StudentID, Name, Major), and suppose the data is:

StudentID	Name	Major
1001	Alice	CS
1002	Bob	Math
1001	Alice	CS

Here, the FD StudentID → Name **holds**, because whenever StudentID is the same (1001), the Name is also the same (Alice).

Satisies in FD

A relation R satisfies a set of functional dependencies F

- if all FDs in F **hold** in R.
- "satisfies" as applying to a **set of FDs**
- while "holds" applies to a **single FD.**

Example:

Let F = { StudentID → Name, StudentID → Major }

The relation Students satisfies F if both of those FDs hold in the data.

FD holds :A specific FD is true for all rows in a relation

Relation **satisfies FD(s):** A relation satisfies a set of FDs if all of them hold in it

Importance of Functional Dependencies in Database Design

- Functional dependencies are essential for database **normalization**
- To reduce redundancy and **improve data integrity** in databases.
- **Prevent anomalies** during database updation.
- Help in **identifying the correct schema** design by ensuring that each attribute is stored in the appropriate table.

Types of Functional Dependencies

There are 2 types of functional dependencies in DBMS.

1. Trivial Functional Dependency

A functional dependency $X \rightarrow Y$ is trivial if **Y is a subset of X.**

Example: StudentID, Course→Course

2. Non-Trivial Functional Dependency

A functional dependency $X \rightarrow Y$ is non-trivial if **Y is not a subset of X.**

Example: Course →Instructorid, course

Armstrong's Axioms?

Armstrong's Axioms are a set of rules used to infer all the functional dependencies on a relational database. They are:

- **Reflexivity**: If Y is a subset of X, then X → Y.
- **Augmentation**: If XZ → YZ, then XZ → YZ for any Z.
- **Transitivity:** If X → Y and Y → Z, then X → Z.
- Additional derived rules include:
- **Union**: If X → Y and X → Z, then X → YZ.
- **Decomposition**: If X → YZ, then X → Y and X → Z.
- **Pseudo Transitivity**: If X → Y and YZ → W, then XZ→ W.

How to Identify Functional Dependencies?

To identify functional dependencies, you can:

- **Analyze Data**: Look for patterns and relationships in sample data.
- **Understand Business Rules**: Comprehend the business rules and constraints governing the data.
- **Consult Documentation**: Use ER diagrams and schema definitions to identify potential dependencies.
- **Use SQL Queries**: Write queries to check if certain attributes consistently determine others.

Rules for Functional Dependency

- A functional dependency X → Y will always hold if all the values of **X are unique** (different) irrespective of the values of Y.
- **Example-**

Consider the following table

A	B	C	D	E
5	4	3	2	2
8	5	3	2	1
1	9	3	3	5
4	7	3	3	8

The following functional dependencies will always hold since all **the values of attribute 'A' are unique.**

A → B A+ =(A,B,C,D,E)

- A → BC
- A → CD
- A → BCD
- A → DE
- A → BCDE

In general, we can say following functional dependency will always hold

A → Any combination of attributes A, B, C, D, E

Similar will be the case for attributes B and E

Rules for Functional Dependency

- A functional dependency $X \rightarrow Y$ will always hold if all the values of Y are same irrespective of the values of X.
- **Example-**

Consider the following table

A	B	C	D	E
8	5	3	2	2
8	5	3	2	1
1	9	3	3	5
4	7	3	3	8

The following functional dependencies will always hold since all the values of attribute 'C' are same-

$A \rightarrow C$

$AB \rightarrow C$

$ABDE \rightarrow C$

$DE \rightarrow C$

$AE \rightarrow C$

In general, we can say following functional dependency will always hold true.

Rules for Functional Dependency

In general, a functional dependency $\alpha \rightarrow \beta$ always holds,

If **either all values of α** are unique or **if all values of β** are same or both

NOTE:

1.For a functional dependency $X \rightarrow Y$ to hold, if two tuples in the table agree on the value of attribute X, then they must also agree on the value of attribute Y.

2. For a functional dependency $X \rightarrow Y$, violation will occur only when for two or more same values of X, the corresponding Y values are different.

Equivalence of Two Sets of Functional Dependencies

Two different sets of functional dependencies for a given relation may or may not be equivalent.
If F and G are the two sets of functional dependencies, then following 3 cases are possible-

Case-01: F covers G ($F \supseteq G$)
Case-02: G covers F ($G \supseteq F$)
Case-03: Both F and G cover each other ($F = G$)

Equivalence of Two Sets of Functional Dependencies

Case-01: Determining Whether F Covers G-

Following steps are followed to determine whether F covers G or not-

Step-01: Take the functional dependencies of set G into consideration.

- For each functional dependency X+ → Y, find the **closure of X u**sing the functional dependencies of **set G.**

Step-02: Take the functional dependencies of set G into consideration.

- For each functional dependency X → Y, find the **closure of X** using the functional dependencies of **set F.**

Step-03: **Compare the results** of Step-01 and Step-02.

- If the functional dependencies of set F has determined all those attributes that were determined by the functional dependencies of set G, then it means F covers G.

Equivalence of Two Sets of Functional Dependencies

Case-02: Determining Whether G Covers F-

Following steps are followed to determine whether G covers F or not-

Step-01:

Take the functional dependencies of set **F into consideration**.

For each functional dependency X → Y, find the **closure of X** using the functional dependencies of **set F**.

Step-02:

Take the functional dependencies of set **F into consideration**.

For each functional dependency X → Y, find the **closure of X** using the functional dependencies of **set G.**

Step-03:

Compare the results of Step-01 and Step-02.

If the functional dependencies of set G has determined all those attributes that were determined by the functional dependencies of set F, then it means G covers F.

Thus, we conclude G covers F (**G ⊇ F**) otherwise not.

Equivalence of Two Sets of Functional Dependencies

Case-02: Determining Whether G Covers F-

Following steps are followed to determine whether G covers F or not-

Step-01:

Take the functional dependencies of set **F into consideration**.

For each functional dependency X → Y, find the **closure of X** using the functional dependencies of **set F**.

Step-02:

Take the functional dependencies of set **F into consideration**.

For each functional dependency X → Y, find the **closure of X** using the functional dependencies of **set G.**

Step-03:

Compare the results of Step-01 and Step-02.

If the functional dependencies of set G has determined all those attributes that were determined by the functional dependencies of set F, then it means G covers F.

Thus, we conclude G covers F (**G ⊇ F**) otherwise not.

Equivalence of Two Sets of Functional Dependencies

Case-03: Determining Whether Both F and G Cover Each Other

If F covers G and G covers F, then both F and G cover each other.
Thus, if both the above cases hold true, we conclude both F and G cover each other (F = G).

PRACTICE PROBLEM BASED ON EQUIVALENCE OF FUNCTIONAL DEPENDENCIES-

Problem-

A relation R (A , C , D , E , H) is having two functional dependencies sets
F and G as shown-

Set F

$A \rightarrow C$

$(AC) \rightarrow D$

$E \rightarrow AD$

$E \rightarrow H$

Set G-

$A \rightarrow CD$

$E \rightarrow AH$

Which of the following holds true?

(A) $G \supseteq F$

(B) $F \supseteq G$

(C) $F = G$

(D) All of the above

Solution-

Determining whether F covers G-

Step-01:

$(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow CD$ using set G

$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AH$ using set G

Step-02:

$(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow CD$ using set F

$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AH$ using set F

Step-03:

Comparing the results of Step-01 and Step-02, we find-

Functional dependencies of set F can determine all the attributes which have been determined by the functional dependencies of set G.

Thus, we conclude F covers G i.e. $F \supseteq G$.

Determining whether G covers F-

Step-01:

$(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow C$ using set F
$(AC)^+ = \{ A , C , D \}$ // closure of left side of $AC \rightarrow D$ using set F
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AD$ and $E \rightarrow H$ using set F

Step-02:

$(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow C$ using set G
$(AC)^+ = \{ A , C , D \}$ // closure of left side of $AC \rightarrow D$ using set G
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AD$ and $E \rightarrow H$ using set G

Step-03:

Comparing the results of Step-01 and Step-02, we find-
Functional dependencies of set G can determine all the attributes which have been determined by the functional dependencies of set F.
Thus, we conclude G covers F i.e. $G \supseteq F$.

Determining whether G covers F-

Step-01:

$(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow C$ using set F
$(AC)^+ = \{ A , C , D \}$ // closure of left side of $AC \rightarrow D$ using set F
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AD$ and $E \rightarrow H$ using set F

Step-02:

$(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow C$ using set G
$(AC)^+ = \{ A , C , D \}$ // closure of left side of $AC \rightarrow D$ using set G
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AD$ and $E \rightarrow H$ using set G

Step-03:

Comparing the results of Step-01 and Step-02, we find-
Functional dependencies of set G can determine all the attributes which have been determined by the functional dependencies of set F.
Thus, we conclude G covers F i.e. $G \supseteq F$.

Determining whether both F and G cover each other-

From Step-01, we conclude F covers G.
From Step-02, we conclude G covers F.
Thus, we conclude both F and G cover each other i.e. F = G.

Thus, Option (D) is correct.

Definition of Canonical Cover

- A canonical cover is a **simplified and reduced** version of the given set of functional dependencies.
- It is also called as **Irreducible set.**

Characteristics-

- Canonical cover **is free** from all the **extraneous functional** dependencies.
- The **closure of canonical cover** is same as that of the given set of functional dependencies.
- Canonical cover is **not unique** and may be more than one for a given set of functional dependencies.

Need of Canonical Cover:

- Working with the set containing extraneous functional dependencies increases the **computation time**.
- Therefore, the given **set is reduced** by eliminating the useless functional dependencies.
- This **reduces the computation time** and working with the irreducible set becomes easier.

Steps To Find Canonical Cover

Step-01:

Write the given set of functional dependencies in such a way that each functional dependency contains exactly one attribute on its right side.

Example-

The functional dependency X → **YZ** will be written as-

X → Y

X → Z

Step-02:

Consider each functional dependency one by one from the set obtained in Step-01.

- Determine whether **it is essential or non-essential**.

To determine whether a functional dependency is essential or not,

compute the closure of its left side-

- Once by considering that the particular functional dependency is **present in the set**
- Once by considering that the particular functional dependency is **not present in the set**

Steps To Find Canonical Cover

Then following two cases are possible-

Case-01: Results Come Out to be Same-

If results come out to be same,

- It means that the presence or absence of that functional dependency
- does not create any difference.
- **Thus, it is non-essential.**
- Eliminate that functional dependency from the set.

NOTE

- Eliminate the non-essential functional dependency from the set as soon as it is discovered.
- Do not consider it while checking the essentiality of other functional dependencies.

Steps To Find Canonical Cover

Case-02: Results Come Out to be Different-

If results come out to be different,

❖ It means that the presence or absence of that functional dependency creates a difference.

❖ **Thus, it is essential.**

❖ Do not eliminate that functional dependency from the set.

❖ Mark that functional dependency as essential.

Steps To Find Canonical Cover

Step-03:

- Consider the newly obtained set of functionaldependencies **after performing Step-02.**
- Check if there is any functional dependency that contains **more than one attribute on its left side**.

Then following two cases are possible:

Case-01: No-

- There exists **no functional dependency** containing **more than one attribute on its left side.**
- In this case, the set obtained in Step-02 is **the canonical cover.**

Case-02: Yes

- There exists at **least one functional dependency** containing **more than one attribute on its left side.**
- In this case, consider all such functional dependencies one by one.
- Check **if their left side can be reduced.**

Problem-

The following functional dependencies hold true for the relational scheme

R (W , X , Y , Z)

$X \rightarrow W$

$WZ \rightarrow Y$

$Y \rightarrow WXZ$

Write the irreducible equivalent for this set of functional dependencies.

Solution- **Step-01:**

Write all the functional dependencies such that each contains **exactly one attribute on its right side-**

$X \rightarrow W$

$WZ \rightarrow X$

$WZ \rightarrow Y$

$Y \rightarrow W$

$Y \rightarrow X$

$Y \rightarrow Z$

Step-02: Check the essentiality of each functional dependency one by one.

For $X \rightarrow W$:

- Considering $X \rightarrow W$, $(X)^+ = \{ X , W \}$
- Ignoring $X \rightarrow W$, $(X)^+ = \{ X \}$ Now,
- Clearly, the two results are different.
- Thus, we conclude that $X \rightarrow W$ is essential and can not be eliminated.

For $WZ \rightarrow X$:

Considering $WZ \rightarrow X$, $(WZ)^+ = \{ W , X , Y , Z \}$

- Ignoring $WZ \rightarrow X$, $(WZ)^+ = \{ W , X , Y , Z \}$ Now,
- Clearly, the two results are same.
- Thus, we conclude that $WZ \rightarrow X$ is non-essential and can be eliminated.

Eliminating $WZ \rightarrow X$, our set of functional dependencies reduces to-

$X \rightarrow W$

$WZ \rightarrow Y$

$Y \rightarrow W$

$Y \rightarrow X$

For $WZ \to Y$:

Considering $WZ \to Y$, $(WZ)^+ = \{ W , X , Y , Z \}$

Ignoring $WZ \to Y$, $(WZ)^+ = \{ W , Z \}$

Now,

Clearly, the two results are different.

Thus, we conclude that $WZ \to Y$ is essential and can not be eliminated.

For $Y \to W$:

Considering $Y \to W$, $(Y)^+ = \{ W , X , Y , Z \}$

Ignoring $Y \to W$, $(Y)^+ = \{ W , X , Y , Z \}$

Now,

Clearly, the two results are same.

Thus, we conclude that $Y \to W$ is non-essential and can be eliminated.

Eliminating $Y \to W$, our set of functional dependencies reduces to-

$X \to W$

$WZ \to Y$

$Y \to X$

$Y \to Z$

For $Y \to X$:

Considering $Y \to X$, $(Y)^+ = \{ W , X , Y , Z \}$

- Ignoring $Y \to X$, $(Y)^+ = \{ Y , Z \}$ Now,
- Clearly, the two results are different.
- Thus, we conclude that $Y \to X$ is essential and can not be eliminated.

For $Y \to Z$:

Considering $Y \to Z$, $(Y)^+ = \{ W , X , Y , Z \}$

- Ignoring $Y \to Z$, $(Y)^+ = \{ W , X , Y \}$

Now,

- Clearly, the two results are different.
- Thus, we conclude that $Y \to Z$ is essential and can not be eliminated.

From here, our essential functional dependencies are-

$X \to W$

$WZ \to Y$

$Y \to X$

$Y \to Z$

Step-03:

Consider the functional dependencies having more than one attribute on their left side.
Check if their left side can be reduced.

In our set,
Only $WZ \rightarrow Y$ contains more than one attribute on its left side.
Considering $WZ \rightarrow Y$, $(WZ)^+ = \{ W , X , Y , Z \}$

Now,
Consider all the possible subsets of WZ.
Check if the closure result of any subset matches to the closure result of WZ.
$(W)^+ = \{ W \}$
$(Z)^+ = \{ Z \}$

Clearly,
None of the subsets have the same closure result same as that of the entire left side.
Thus, we conclude that we can not write $WZ \to Y$ as $W \to Y$ or $Z \to Y$.
Thus, set of functional dependencies obtained in step-02 is the canonical cover.

Finally, the canonical cover is-
$X \to W$
$WZ \to Y$
$Y \to X$
$Y \to Z$

Decomposition of a Relation-

The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

Properties of Decomposition-

The following two properties must be followed when decomposing a given relation-

1. Lossless decomposition-

Lossless decomposition ensures-

- No information is lost from the original relation during decomposition.
- When the sub relations are joined back, the same relation is obtained that was decomposed.
- Every decomposition must always be lossless.

Dependency preservation ensures-

- None of the functional dependencies that holds on the original relation are lost.
- The sub relations still hold or satisfy the functional dependencies of the original relation.

1. Lossless Join Decomposition-

- Consider there is a relation R which is decomposed into sub relations R_1 , R_2 , …. , R_n.
- This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.
- For lossless join decomposition, we always have-

where $\bowtie$ **is a natural join** operator

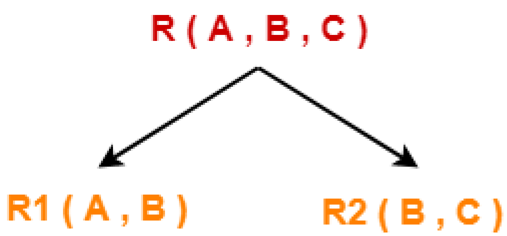
Example-

Consider the following relation R(A , B , C)-

R(A , B , C)

A	B	C
1	2	1
2	5	3
3	3	3

Consider this relation is decomposed into two sub relations R_1(A , B) and R_2(B , C)- The two sub relations are-



B	C
2	1
5	3
3	3

B	C
2	1
5	3
3	3

Now, let us check whether this decomposition is lossless or not.

For lossless decomposition, we must have- $\mathbf{R_1 \bowtie R_2 = R}$

Now, if we perform the natural join ($\bowtie$) of the sub relations R_1 and R_2 , we

This relation is same as the original relation R.

Thus, we conclude that the above decomposition is **lossless join decomposition.**

R(A , B, C)

A	B	C
1	2	1
2	5	3
3	3	3

NOTE-

- Lossless join decomposition is also known as **non-additive join decomposition.**
- This is because the resultant relation after joining the sub relations is same as the decomposed relation.
- **No extraneous tuples** appear after joining of the sub relations.

<u>2. Lossy Join Decomposition-</u> known as **careless decomposition.**

Consider there is a relation R which is decomposed into sub relations R_1 , R_2 , …. , R_n.

- This decomposition is called lossy join decomposition when the join of the sub relations **does not result in the same** relation R that was decomposed.
- The natural join of the sub relations is always found to have some extraneous tuples.
- For lossy join decomposition, we always have- $R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots \bowtie R_n \supset R$

where $\bowtie$ is a natural join operator

<u>Example-</u>

Consider the following relation R(A , B , C)-

R(A , B , C)

A	B	C
1	2	1
2	5	3
3	3	3

Consider this relation is decomposed into two sub relations as R_1(A , C) and R_2(B , C)-

R (A , B , C)

R1 (A , C)

A	C
1	1
2	3
3	3

R2 (B , C)

B	C
2	1
5	3
3	3

Now, let us check whether this decomposition is lossy or not. For lossy decomposition, we must have- $R_1 \bowtie R_2 \supset R$

A	B	C
1	2	1
2	5	3
2	3	3
3	5	3
3	3	3

This relation is not same as the original relation R and **contains some extraneous tuples.**

Clearly, $R_1 \bowtie R_2 \supset R$.

Thus, we conclude tha the above decomposition is **lossy join decomposition.**

Dependency Preservation Question 1:

Consider a schema R(A, B, C, D) and following functional dependencies.

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow B$

Then decomposition of R into $R_1(A, B)$, $R_2(B, C)$ and $R_3(B, D)$ is ______

1. Dependency preserving and lossless join.
2. Lossless join but not dependency preserving.
3. Dependency preserving but not lossless join.
4. Not dependency preserving and not lossless join.

__**Lossless join:**__ If there is **no loss of information** by **replacing** a relation **R with** two relation schema **R1 & R2**, then join can be said as Lossless decomposition.

That means, after **natural join R1 & R2**, we will get exactly the same relation **R.**

Some **properties** of lossless decomposition are:

1. R1 ∩ R2 = R1 or **R1 ∩ R2 = R2**

2. R1 U R2 = R

3. R1 ∩ R2 = super key of either R1 or R2

__**Decomposition preservation:**__ Decomposition D = **{ R1,R2..R_n }** of relation **R** is dependency preserving with respect to **Functional dependency F** if :

Closure of Union of all functional dependencies with respect to each relation is equivalent to closure of F.

In other words, If relation **R has FD F**, its decomposed relations **R1 & R2 has FD F1 & F2** respectively then, **F' = F1 U F2 & $F'^+ = F^+$** .

EXPLANATION:

Lossless: Consider relation **R1(A,B) & R2(B,C)**:

R1 ∩ R2 -> $(B)^+$ -> { B,C,D } {B is common attribute in both relation so find its **closure**}

Since B^+ derives relation R2, Hence, relation **R1'(A,B,C)** is **lossless.**

Now consider relation **R1'(A,B,C) & R3(B,D)**:

R1' ∩ R3 -> $(B)^+$ -> { B,C,D } {B is common attribute in both relation so find its **closure**}

Since B^+ derives relation R3, Hence relation **R(A,B,C,D)** is **lossless.**

Decomposition:

Here,

R1(A,B) -> { A->B }, --- FD1

R2(B,C) -> { B->C }, --- FD2

R3(B,D) -> { D->B } --- FD3

F' = FD1 U FD2 U FD3

= { A->B, B->C, D->B **}**

Now find closure of **F'** & **F :**

F'^+ = {A,B,C,D} & **F^+** = {A,B,C,D}

Since the closure of F' also preserve dependency C -> D,

Hence, given decomposition of R into R_1(A, B), R_2(B, C) and R_3(B, D) is dependency preserving and lossless join.