MODULE 4

NORMALIZATION

CO – Students will be able to demonstrate the Relational Database Design concepts.

EDULINE FOR CSE STUDENTS

Prepared By Mr. EBIN PM, AP

RELATIONAL DATABASE DESIGN

- Each relation schema consists of a number of attributes, and the relational database schema consists of a number of relation schemas.
- Relational database design ultimately produces a set of relations.
- The implicit goals of the design activity are information preservation and minimum redundancy.

Prepared By Mr. EBIN PM, AP,CU EDULINE 2

ANOMALIES IN A DATABASE

- Consider the following relational schema pertaining to the information about a student maintained by a university.

STDINF (Name, Course, Phone_No, Major, Professor, Grade)

Name	Course	Phone_no	Major	Professor	Grade
Jones	353	237-4539	Comp.sci	Smith	A
Ng	329	427-7390	Chemistry	Turner	B
Jones	388	237-4539	Comp.sci	Clark	B
Martin	456	388-5783	Physics	James	A
Dulles	293	371-6259	Decision Sci.	Cook	C
Duke	491	823-7293	Maths	Lamb	B
Duke	356	823-7293	Maths	Bond	In prog
Jones	492	237-4539	Comp.sci	Cross	In prog
Baxter	379	839-0827	English	Broes	C

- The relation schema STDINF can lead to several undesirable problems.

1. REDUNDANCY

- The aim of the database system is to reduce redundancy, meaning that information is to be stored only once.
- Storing information several times leads to the waste of storage space and an increase in the total size of the data stored.
- In the above relation the Major and Phone_no of a student are stored several times in the data base; once for each course that is or was taken by a student.

2. UPDATE ANOMALIES

- multiple copies of the same fact may lead to update anomalies or inconsistencies when an update is made and only some of the multiple copies are updated.
- Thus a change in the Phone_no of Jones must be made, for consistency, in all tuples pertaining to the student Jones.
- If one of the three tuples of the figure is not changed to reflect the new Phone_no of Jones, there will be an inconsistency in the data.

3. INSERTION ANOMALIES

- If this is the only relation in the database showing the association between the faculty member and the course he or she teaches, the fact that a given Professor is teaching a given Course cannot be entered in the database unless a student is registered in the Course.

4. DELETION ANOMALIES

- If the only student registered in a given Course discontinues the course, the information as to which Professor is offering the course will be lost if this is the only relation in the database showing the association between a faculty member and the course she or he teaches.
- If another relation in the database also establishes the relationship between a Course and a Professor who teaches that course, the deletion of the last tuple in "STDINF" for a given course will not cause the information about the course's teacher to be lost.

- Redundancy, storing the same information several times in a database, can result in update anomalies (all copies need to be updated), insertion anomalies (certain information cannot be stored unless other information is stored as well), and deletion anomalies (deleting some information means loss of other information as well).
- We can reduce redundancy by replacing a relation schema R with several smaller relation schemas.
- This process is called **decomposition.**

Prepared By Mr. EBIN PM, AP , CU EDULINE 7

FUNCTIONAL DEPENDENCY

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- It is basically a many-to-one relationship from one set of attributes to another with in a given relational variable.
- Given two sets of attributes X and Y , Y is said to be functionally dependent on X if a given value for each attribute in X uniquely determines the value of the attribute in Y.(i.e., each X value is associated with exactly one Y value.)

Prepared By Mr. EBIN PM, AP EDULINE 8

- Functional dependency is represented by an arrow sign (→) that is, **X→Y**, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.
- We can say that the FD X→Y is satisfied on the relation R if the cardinality of $\Pi_Y (\sigma_{X=X} (R))$ is at most one.
- A functional dependency X→Y is said to be trivial if Y is a subset of X , otherwise non trivial.
- Trivial FDs always hold.
- If an FD X → Y holds, where x intersect Y = Φ, it is said to be a completely non-trivial FD.

Account_No	Loan_No	Amount
101	L1	5000
102	L2	5000
103	L3	5000
104	L2	5000

Here Loan_No → Amount functional dependency is exist.

ARMSTRONG'S AXIOMS

- If F is a set of functional dependencies then the closure of F, denoted as F+, is the set of all functional dependencies logically implied by F ($F \models x \rightarrow y$).
- Armstrong's Axioms (Inference axioms) are a set of rules that, when applied repeatedly, generates a closure of functional dependencies.

Reflexivity: $x \rightarrow x$

Augmentation : $(x \rightarrow y) \models (xz \rightarrow y \text{ and } xz \rightarrow yz)$

Transitivity : $(x \rightarrow y \text{ and } y \rightarrow z) \models (x \rightarrow z)$

Additivity(union) : $(x \rightarrow y \text{ and } x \rightarrow z) \models (x \rightarrow yz)$

Projectivity (decomposition) : $(x \rightarrow yz) \models (x \rightarrow y \text{ and } x \rightarrow z)$

Psudo transitivity : $(x \rightarrow y \text{ and } yz \rightarrow w) \models (xz \rightarrow w)$

CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES

- The set of functional dependencies that is logically implied (involve) by F is called the closure of F and is written as F+

Example:

- Let F = {A→B, B→C}. Then F+ can be finding using the above axioms

F+ = {A→B, B→C, A→C, A→BC , AC→BC}

Prepared By Mr. EBIN PM, AP EDULINE 13

PRIME ATTRIBUTE & NONPRIME ATTRIBUTE

- An attribute, which is a part of the prime-key, is known as a prime attribute.
- An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

Example:

If R (ABCDEH) and F= {A→BC, CD→E, E→C, AH→D}, then AH is the only candidate key of R.

The attributes A and H are prime and the attributes B, C, D and E are nonprime.

Prepared By Mr. EBIN PM, AP EDULINE 14

PARTIAL DEPENDENCY

- Consider the relation R (name, course, grade, phone_ no, major, course_dept).
- The functional dependencies are F {name→phone_no,major ; course→course_dept ; name course→grade}.
- Name course is a candidate key. Name and course are prime attributes. Grade is fully functionally dependent on the candidate key. Phone_no, course_dept and major are partially dependent on the candidate key.

Another example

- Given R (ABCD) and F {AB→C, B→D}, the key of this relation is AB and D is partially dependent on the key.

TRANSITIVE DEPENDENCY

- Given R (ABCDE) and functional dependencies f {AB→C, B→D, C→E}.
- AB is the key and E is transitively dependent on the key, since AB→C→E.

NORMALIZATION & NORMAL FORMS

- Normalization is a method to remove all these anomalies and bring the database to a consistent state.
- Given a relation schema, we need to decide whether it is a good design or whether we need to decompose it into smaller relations.
- Such a decision must be guided by an understanding of what problems, if any, arise from the current schema.
- To provide such guidance, several normal forms have been proposed.
- If a relation schema is in one of these normal forms, we know that certain kinds of problems cannot arise.

❖FIRST NORMAL FORM (1NF)

- First Normal Form is defined in the definition of relations (tables) itself.
- This rule defines that all the attributes in a relation must have atomic domains.
- The values in an atomic domain are indivisible units.
- A database schema is in 1NF; if every relation schema included in the database schema is in 1NF.

unnormalized relation

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

- We re-arrange the relation (table) as below, to convert it to First Normal Form.
- Each attribute must contain only a single value from its predefined domain.

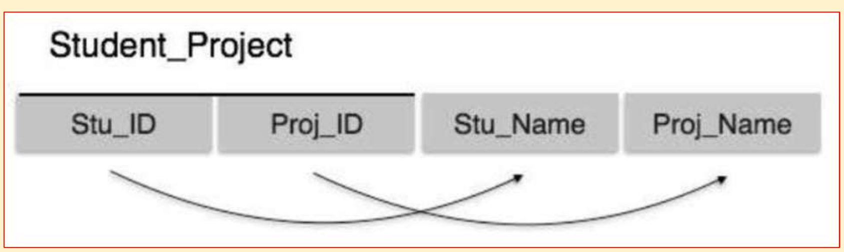
Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

Relation in 1NF

Prepared By Mr. EBIN PM, AP EDULINE 19

❖SECOND NORMAL FORM (2NF)

- If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.
- A second normal form does not permit partial dependency between a non-prime attribute and the relation keys.

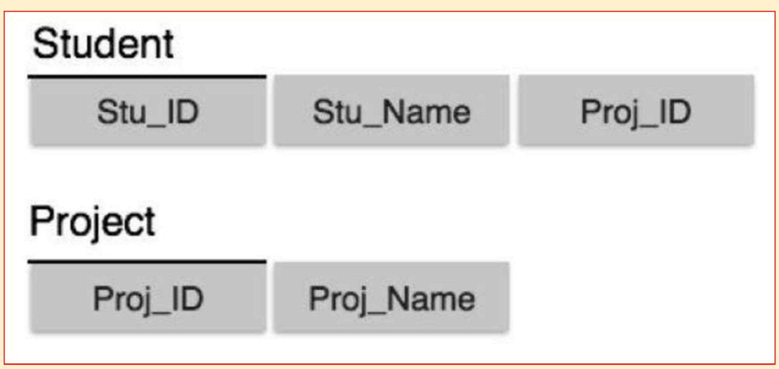


Relation not in 2NF

Prepared By Mr. EBIN PM, AP EDULINE 20

- We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID.
- According to the rule, non-key attributes, i.e., Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently.
- This is called partial dependency, which is not allowed in Second Normal Form.

- We broke the relation in two as depicted in the above picture. So there exists no partial dependency.



Relation in 2NF

❖THIRD NORMAL FORM (3NF)

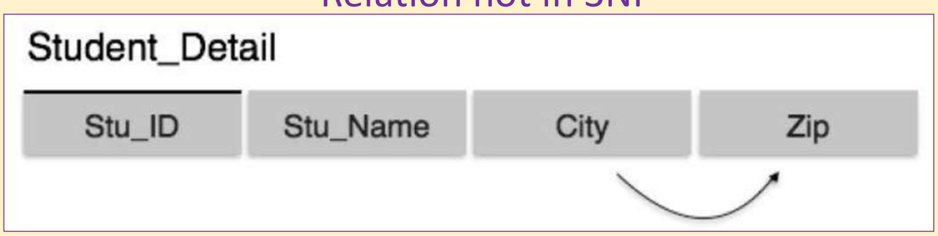
- For a relation to be in Third Normal Form it must be in Second Normal form and the following must satisfy:

➢No non-prime attribute is transitively dependent on prime key attribute.

➢For any non-trivial functional dependency, X → A, then either:

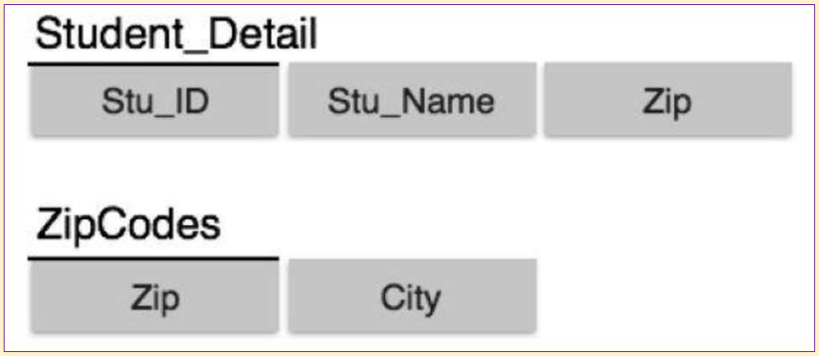
- X is a super key or,
- A is prime attribute (each element of A is part of some candidate key)

Relation not in 3NF



- We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute.
- We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a super key nor is City a prime attribute.
- Additionally, Stu_ID→ Zip → City, so there exists transitive dependency.

- To bring this relation into third normal form, we break the relation into two relations as follows:



Relation in 3NF

- Consider the following Table **TEACHER_DETAILS**

ID	NAME	SUBJECT	STATE	COUNTRY
29	Lalita	English	Gujrat	INDIA
33	Ramesh	Geography	Punjab	INDIA
49	Sarita	Mathematics	Maharashtra	INDIA
78	Zayed	History	Bihar	INDIA

- The candidate key is **ID**
- The functional dependency { ID->NAME, ID->SUBJECT, ID->STATE, STATE->COUNTRY }

- For the above relation, ID->STATE, STATE->COUNTRY is transitive dependency.
- This does not satisfy the conditions of the Third Normal Form.
- So in order to transform it into Third Normal Form, we need to break the table into two tables in total and we need to create another table for STATE and COUNTRY with STATE as the primary key.

TEACHER_DETAILS:

ID	NAME	SUBJECT	STATE
29	Lalita	English	Gujrat
33	Ramesh	Geography	Punjab
49	Sarita	Mathematics	Maharashtra
78	Zayed	History	Bihar

STATE_COUNTRY:

STATE	COUNTRY
Gujrat	INDIA
Punjab	INDIA
Maharashtra	INDIA
Bihar	INDIA

Example:

- Consider relation R(A, B, C, D, E) and the FDs are given below

  A -> BC,

  CD -> E,

  B -> D,

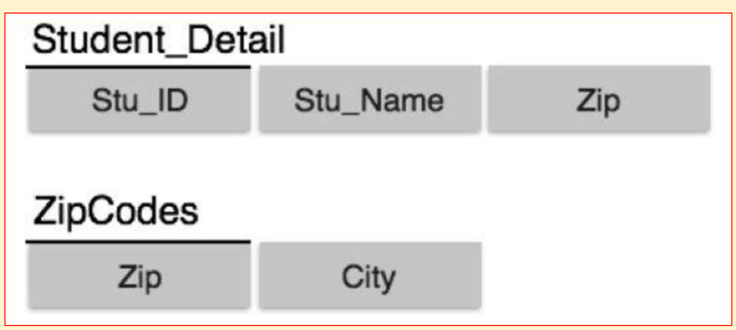
  E -> A

- All possible candidate keys in above relation are {A, E, CD, BC}
- All attribute are on right sides of all functional dependencies are prime.

❖BOYCE CODD NORMAL FORM (BCNF)

- Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that

➢For any non-trivial functional dependency, X → A, X must be a super-key. (Check whether the left side of each dependency in F is a super key)



- In the above image, Stu_ID is the super-key in the relation Student_Detail and Zip is the super-key in the relation ZipCodes. So,

  Stu_ID → Stu_Name, Zip and

  Zip → City

  Which confirms that both the relations are in BCNF

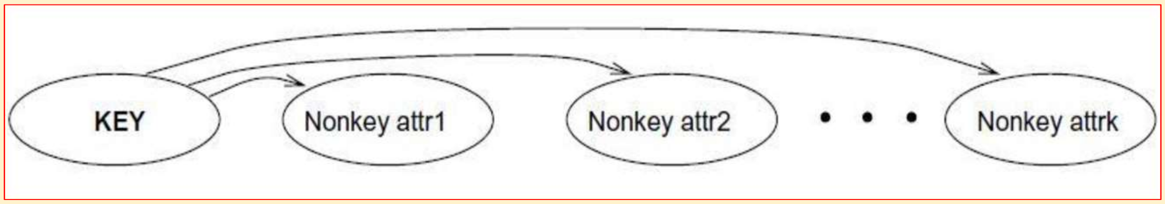


Fig: FDs in a BCNF relation

RELATIONAL DATABASE DESIGN

- The criteria for design are the following:

➢The design is **content preserving**; if the original relation can be derived from the relations resulting from the design process. Since the join operation is used in deriving the original relation from its decomposed relations; this criterion is also called lossless join decomposition.

➢The relation design is **dependency preserving**; if the original set of constraints can be derived from the dependencies in the output of the design process.

- The relation design is **free from interrelation join constraints**; if there are no dependencies that can only be derived from the join of two or more relations in the output of the design process.
- Since join operation is a computationally expensive process, it is desirable that the database design be free of such interrelation join constraints

Lossless Join Decomposition

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

- For a decomposition to be lossless, it should hold the following conditions

1. Union of Attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

   Att(R1) U Att(R2) = Att(R)

2. Intersection of Attributes of R1 and R2 must not be NULL.

   Att(R1) ∩ Att(R2) ≠ Φ

3. Common attribute **must be a key** for at least one relation (R1 or R2)

   Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)

Example:

A relation R(A,B,C,D) with FD set {A->BC} is decomposed into R1(ABC) and R2(AD). This is lossless join decomposition because

- First rule holds true as Att(R1) ∪ Att(R2)=(ABC) ∪ (AD)= (ABCD) = Att(R)
- Second rule holds true as Att(R1) ∩ Att(R2) = (ABC) ∩ (AD) ≠ Ø
- Third rule holds true as Att(R1) ∩ Att(R2) = A is a key of R1(ABC) because A->BC is given.

❖Dependency Preserving Decomposition

- If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 or must be derivable from combination of FD's of R1 and R2.
- For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of R1(ABC).

Question:

Consider a schema R(A,B,C,D) and functional dependencies A->B and C->D. Then the decomposition of R into R1(AB) and R2(CD) is

A. dependency preserving and lossless join

B. lossless join but not dependency preserving

C. dependency preserving but not lossless join

D. not dependency preserving and not lossless join

Ans:

For lossless join decomposition, these three conditions must hold true:

R(A,B,C,D) R1(AB) R2(CD)

1. Att(R1) U Att(R2) = ABCD = Att(R)
2. Att(R1) ∩ Att(R2) = Φ, which violates the condition of lossless join decomposition. Hence the decomposition is not lossless.

For dependency preserving decomposition,

- A->B can be ensured in R1(AB) and C->D can be ensured in R2(CD). Hence it is dependency preserving decomposition.

So, the correct option is C.

➢ Advantages of Lossless Decomposition in DBMS

- It eliminates data redundancy
- Improves the efficiency of the database
- Reduces storage space requirements
- Enables effective maintenance of the database
- Preserves all the information from the original table
- Prevents data inconsistencies

ALGORITHM TO CHECK IF DECOMPOSITION IS LOSSY OR LOSSLESS

Step 1 – Create a table with M rows and N columns

M= number of decomposed relations.

N= number of attributes of original relation.

Step 2 – If a decomposed relation Ri has attribute A then

Insert a symbol (say 'a') at position (Ri,A)

Step 3 – Consider each FD X->Y

If column X has two or more symbols then

Insert symbols in the same place (rows) of column Y.

Step 4 – If any row is completely filled with symbols then

Decomposition is lossless.

Else

Decomposition is lossy.

Example:

Consider R(A,B,C,D,E)

F:{A->B, BC->E, ED->A}

R is decomposed into R1(AB) and R2(ACDE). Check the decomposition is lossy or lossless.

Step 1

	A	B	C	D	E
R1					
R2					

Step 2

	A	B	C	D	E
R1	a	a			
R2	a		a	a	a

Step 3

Now let us insert symbol ‘a’ for A->B in second column, second row

	A	B	C	D	E
R1	a	a			
R2	a	a	a	a	a

R2 is completely filled => decomposition is lossless.

Q : Consider a relation schema R(X Y Z W P) is decomposed into R1(X Y) and R2(Z W). determine whether the above R1 and R2 are Lossless or Lossy?

- Not satisfied as Attribute(R1) U Attribute (R2) (X Y Z W) ≠ Attribute (R) = (X Y Z W P)
- Hence relation R (X Y Z W P) decomposed into R1(X Y) and R2(Z W) is a Lossy decomposition.

Q: R = (A, B, C, D, E). We decompose it into R1 = (A, B, C), R2 = (A, D, E). The set of functional dependencies is: A → BC, CD → E, B → D, E → A. Show that this decomposition is a lossless-join decomposition.

R1 ∩ R2 = A

(A → BC)

this is a lossless-join decomposition.

How To Find the Closure of a Set of Attributes (With Examples)

Q: Relation R(A, B, C, D, E).

functional dependencies: {A->B, B->C, C->D, D->E}. Find {A}+

Ans:

- First, we add A to {A}+.
- What columns can be determined given A? We have A -> B, so we can determine B. Therefore, {A}+ is now {A, B}.
- What columns can be determined given A and B? We have B -> C in the functional dependencies, so we can determine C. Therefore, {A}+ is now {A, B, C}.

Prepared By Mr. EBIN PM, AP EDULINE 47

- Now, we have A, B, and C. What other columns can we determine? Well, we have C -> D, so we can add D to {A}+.
- Now, we have A, B, C, and D. Can we add anything else to it? Yes, since D -> E, we can add E to {A}+.
- We have used all of the columns in R and we have all used all functional dependencies.

{A} + = {A, B, C, D, E}

Q: We are given R(A, B, C, D, E, F).

The functional dependencies are {AB->C, BC->AD, D->E, CF->B}.

What is {A, B}+?

Prepared By Mr. EBIN PM, AP EDULINE 48

- We start with {A, B}.
- What columns can we determine, given A and B? We have AB -> C, so we can add C to {A, B}+.
- We now have A, B, and C. What other columns can we determine? We have BC -> AD. We already have A in {A, B}+, so we can add D.
- So, we now have A, B, C, and D. What else can we add? We have D -> E, so we can add E to {A, B}+.
- Now {A, B}+ is {A, B, C, D, E}. Can we add anything else? No. We have one more functional dependency in our set that we did not use: CF -> B. We can't use this dependency because F is not in {A, B}+.
- Thus, **{A, B}+ = {A, B, C, D, E}**

Prepared By Mr. EBIN PM, AP EDULINE 49

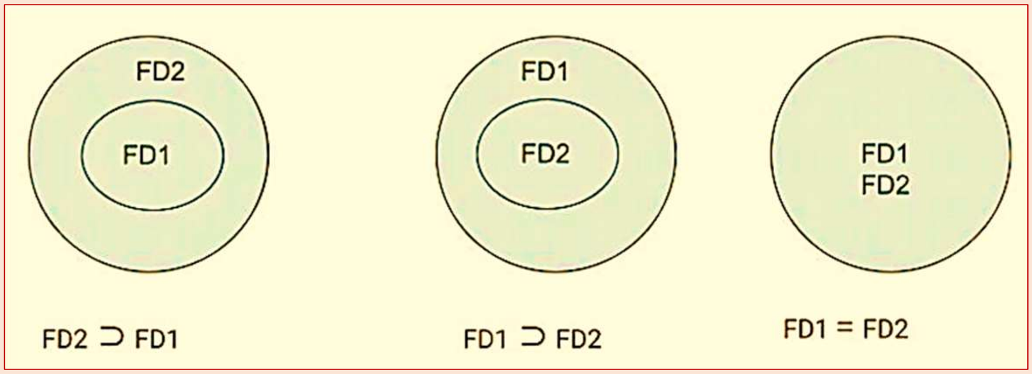
Equivalence Of Functional Dependencies

- Two FDs F and G sets over schema R are equivalent if **F+ = G+**
- It means that if every functional dependency of F is in G+ and every functional dependence of G is in F+, then we would say that the sets of functional dependencies F and G are 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)

Prepared By Mr. EBIN PM, AP EDULINE 50



➢Case-01: Determining Whether F Covers G

Step-01:

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

➢**Case-02: Determining Whether G Covers F**

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.

Example:

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

Set F

A → C, AC → D, E → AD, E → H

Set G

A → CD, E → AH

Which of the following holds true?

(A) G ⊇ F

(B) F ⊇ G

(C) F = G

(D) All of the above

Set F

A → C, AC → D, E → AD, E → H

Set G

A → CD, E → AH

➢Determining whether F covers G-

Step-01:

(A)+ = { A , C , D } // closure of left side of A → CD using set G

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

Step-02:

(A)+ = { A , C , D } // closure of left side of A → CD using set F

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

- Comparing the results of Step-01 and Step-02
- 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 ⊇ G.**

➢ Determining whether G covers F-

Set F
A → C, AC → D, E → AD, E → H
Set G
A → CD, E → AH

Step-01:

(A)+ = { A , C , D } // closure of left side of A → C using set F

(AC)+ = { A , C , D } // closure of left side of AC → D using set F

(E)+ = { A , C , D , E , H } // closure of left side of E → AD and E → H using set F

Step-02:

(A)+ = { A , C , D } // closure of left side of A → C using set G

(AC)+ = { A , C , D } // closure of left side of AC → D using set G

(E)+ = { A , C , D , E , H } // closure of left side of E → AD and E → H using set G

- 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 ⊇ 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.

CANONICAL COVER

- The canonical cover is defined as a simplified and reduced version of a given functional dependency.
- The canonical cover is free from all irrelevant functional dependencies
- The closure of canonical cover is the same as that of a given set of functional dependency
- The canonical cover is not unique because there may be more than one canonical cover for a given set of functional dependency.
- It is called **irreducible** a set of functional dependencies.

➢Need

- 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.

Example:

Relational scheme R (W , X , Y , Z) and the FDs are

{ X → W , WZ → XY , Y → WXZ }. Write the irreducible equivalent (canonical cover)for this set of functional dependencies.

Step-01:

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

  X → W , WZ → X , WZ → Y, Y → W, Y → X, Y → Z

Step-02:

- Check the essentiality of each functional dependency one by one.

For X → W:

- Considering X → W, (X)+ = { X , W }
- Ignoring X → W, (X)+ = { X }

X → W
WZ → X
WZ → Y
Y → W
Y → X
Y → Z

Now,

- Clearly, the two results are different.
- Thus, we conclude that X → W is essential and can not be eliminated.

For WZ → X:

X → W
WZ → X
WZ → Y
Y → W
Y → X
Y → Z

- Considering WZ → X, (WZ)+ = { W , X , Y , Z }
- Ignoring WZ → X, (WZ)+ = { W , X , Y , Z }

Now,

- Clearly, the two results are same.
- Thus, we conclude that WZ → X is non-essential and can be eliminated.
- Eliminating WZ → X, our set of functional dependencies reduces to

X → W , WZ → Y, Y → W, Y → X, Y → Z

- Now, we will consider this reduced set in further checks.

Prepared By Mr. EBIN PM, AP EDULINE 63

For WZ → Y:

X → W
WZ → Y
Y → W
Y → X
Y → Z

- Considering WZ → Y, (WZ)+ = { W , X , Y , Z }
- Ignoring WZ → Y, (WZ)+ = { W , Z }

Now,

- Clearly, the two results are different.
- Thus, we conclude that WZ → Y is essential and can not be eliminated.

Prepared By Mr. EBIN PM, AP EDULINE 64

For Y → W:

X → W
WZ → Y
Y → W
Y → X
Y → Z

- Considering Y → W, (Y)+ = { W , X , Y , Z }
- Ignoring Y → W, (Y)+ = { W , X , Y , Z }

Now,

- Clearly, the two results are same.
- Thus, we conclude that Y → W is non-essential and can be eliminated.
- Eliminating Y → W, our set of functional dependencies reduces to

X → W , WZ → Y, Y → X, Y → Z

- Now, we will consider this reduced set in further checks.

Prepared By Mr. EBIN PM, AP EDULINE 65

For Y → X:

X → W
WZ → Y
Y → X
Y → Z

- Considering Y → X, (Y)+ = { W , X , Y , Z }
- Ignoring Y → X, (Y)+ = { Y , Z }

Now,

- Clearly, the two results are different.
- Thus, we conclude that Y → X is essential and can not be eliminated.

Prepared By Mr. EBIN PM, AP EDULINE 66

For Y → Z:

- Considering Y → Z, (Y)+ = { W , X , Y , Z }
- Ignoring Y → Z, (Y)+ = { W , X , Y }

Now,

- Clearly, the two results are different.
- Thus, we conclude that Y → Z is essential and can not be eliminated.
- From here, our essential functional dependencies are

X → W, WZ → Y, Y → X, Y → Z

X → W
WZ → Y
Y → X
Y → Z

Prepared By Mr. EBIN PM, AP EDULINE 67

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 → Y contains more than one attribute on its left side.
- Considering WZ → 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.

Prepared By Mr. EBIN PM, AP EDULINE 68

(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 → Y as W → Y or Z → Y.
- Thus, set of functional dependencies obtained in step-02 is the canonical cover.

X → W, WZ → Y, Y → X, Y → Z

Prepared By Mr. EBIN PM, AP EDULINE 69