Database Management System
Module 4

Lect 1 : FUNCTIONAL DEPENDANCY

Module 4

- Different anomalies in designing a database, The idea of normalization, Functional dependency, Armstrong's Axioms (proofs not required), Closures and their computation, Equivalence of Functional Dependencies (FD), Minimal Cover (proofs not required). First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Boyce Codd Normal Form (BCNF), Lossless join and dependency preserving decomposition, Algorithms for checking Lossless Join (LJ) and Dependency Preserving (DP) properties.

Functional Dependency

Sl	Name	Address
1	A	A1
2	B	A2
3	C	A3
3	C	A4
4	D	A5
5	E	A6

- Sl → Name
  - Name is Functionally dependent on Sl
  - Sl determines Name

Functional Dependency

- A **functional dependency** is a constraint between two sets of attributes from the database.
- It describes relationship between attributes in a relation
- A functional dependency, denoted by

  $X \rightarrow Y$,

  between two sets of attributes X and Y in a Relation R specifies a constraint that, for any two tuples t1 and t2 ,

  if $t1[X] = t2[X]$ then $t1[Y] = t2[Y]$.

  i,e any values in X attribute is same then the corresponding values in Y attribute must also be same.

- When X→Y
  - X is known as Determinent
  - Y is known as Dependent

Functional Dependency

Sl	Name	Address
1	A	A1
2	B	A2
3	C	A3
3	C	A4
4	D	A5
5	E	A6

- If $X \rightarrow Y$, if $t1[X] = t2\ [X]$ then $t1[Y]=t2[Y]$
- Example
  - Sl → Name - True
  - Sl → Address - False
  - Name → Address - False

Roll No	Name	Marks	Dept	Course
1	Smith	78	CSE	C1
2	Ramesh	60	EE	C1
3	Smith	78	CSE	C2
4	Ramesh	60	EE	C3
5	John	80	IT	C3
6	Bevin	80	EC	C2

- Here the following Functional Dependency Exist
  - RollNo -> Name (since all values in roll no is unique)
  - RollNo -> Marks (since all values in roll no is unique)
  - Rollno,Name -> Marks (Sine Rollno Name combination is not repeating
- Here the following Functional Dependency does not Exist
  - Name -> Rollno (Names like Smith & Ramesh is repeating)
  - Dept-> Course (Dept CSE & EE Repeating)
  - Course->Dept (Course C1,C2 C3 is repeating)

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

- Check which FD exist
  - Teacher → Course
  - Course → Text
  - Text → course
  - Teacher,course → Text

Types of Functional Dependency

1. **Trivial Functional Dependency**
2. **Non Trivial**

- **<u>Trivial Functional Dependency</u>**
- A functional dependency X → Y is trivial if Y ⊆ X (⊆ means Subset)

<u>Roll No</u>	Name	Marks	Dept	Course
1	Smith	78	CSE	C1
2	Ramesh	60	EE	C1
3	Smith	78	CSE	C2
4	Ramesh	60	EE	C3
5	John	80	IT	C3
6	Bevin	80	EC	C2

eg1: Rollno, Name → Name

- Here Name in RHS is subset of LHS

Eg2 Rollno → Rollno

Non Trivial Functional Dependency

- A functional dependency $X \rightarrow Y$ is non trivial if X Intersection $Y = \Phi$
- Eg : Rollno $\rightarrow$ Name (Since there is nothing common)

Inference rules or Axioms

- Functional Dependency is a property defined by someone who knows the semantics of the attributes of Relation R
- There are a set of **inference rules or axioms** that can be used to infer new dependencies from a given set of dependencies

Inference rules or Axioms

1. IR1 : Reflexive Rule

   $X \rightarrow X$ or $X \rightarrow Y$, if $Y \subseteq X$

2. IR2 : Transitivity

   If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

   eg: If Name → Marks and Marks →Dept then Name → Dept

3. IR3 : Augmentation Rule :

   If $X \rightarrow Y$ then $XZ \rightarrow YZ$

   eg : if Rollno → Name then we can also write

   Rollno, Marks → Name, Mark

Inference rules IR1 through IR3 are known as **Armstrong's inference rules.**

4. IR4 : Decomposition Rule

If $X \rightarrow Y Z$ then $X \rightarrow Y$ and $X \rightarrow Z$

Decomposition is applied only in RHS

5. IR5: Union or additive Rule

If $X \rightarrow Y$, $X \rightarrow Z$ then $X \rightarrow YZ$

6. IR6: Pseudotransitive rule

If $X \rightarrow Y$, $WY \rightarrow Z$ then $WX \rightarrow Z$

Closure of Functional Dependencies

- A closure of a set of functional dependency defines all the FDs that can be derived from a given set of FDs.
- Given 'F', set of functional dependencies for a relation schema R closure of F is denoted by F^+
- Armstrong's inference axioms can be used to compute F^+ of F

Example 1

Consider a releation R(A,B,C,D,E) having Functional Dependency {$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$}

(i) Compute Closure of A

(ii) Compute Closure of AB

(iii) Compute Closure of CD

Soln : <u>Closure of A</u>

Closure of A or A^+ is computed by taking all the Functional dependency of A

A^+ = { A, (Since $A \rightarrow A$ [Reflexive Property])

B, (Since $A \rightarrow B$)

C, (Since $B \rightarrow C$)

D, (Since $C \rightarrow D$)

E }(Since $D \rightarrow E$)

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

Closure of AD

AD^+ =A (Since A→A [Reflexive Property])
=D (Since D→D [Reflexive Property])
=B (Since A→B)
=C (Since B→C)
=D (Since C→D)
=E (Since D→E)

Therefore AD^+ = {A,D,B,C,E}

Closure of CD

CD^+ = C (Since C→C[Reflexive Property]
= D (since D→D[Reflexive Property]
=E (Since D→E)

Therefore CD^+= {C,D,E}

Example 2

- consider the relation schema EMP_PROJ; from the semantics of the attributes, we specify the following set *F* of functional dependencies that should hold on EMP_PROJ:

$$
\begin{aligned}
F = \{&\text{Ssn} \to \text{Ename},\\
&\text{Pnumber} \to \{\text{Pname}, \text{Plocation}\},\\
&\{\text{Ssn}, \text{Pnumber}\} \to \text{Hours}\}
\end{aligned}
$$

$$
\begin{aligned}
&\{\text{Ssn}\}^+ = \{\text{Ssn}, \text{Ename}\}\\
&\{\text{Pnumber}\}^+ = \{\text{Pnumber}, \text{Pname}, \text{Plocation}\}\\
&\{\text{Ssn}, \text{Pnumber}\}^+ = \{\text{Ssn}, \text{Pnumber}, \text{Ename}, \text{Pname}, \text{Plocation}, \text{Hours}\}
\end{aligned}
$$

Database Management System
Module 4

Lect 2 :

→EQUIVALENCE OF SETS OF FUNCTIONAL DEPENDENCIES

→ MINIMAL COVER

EQUIVALENCE OF SETS OF FUNCTIONAL DEPENDENCIES

E = $\{A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E\}$

F = $\{A \rightarrow BC, D \rightarrow AE\}$

- A set of functional dependencies E and F is Equivalent if
  - **E covers F and F covers E.**

- E covers F means that all the Functional dependency in F can be inferred from E , (i.e whether E is covering functional dependencies of F)

- F covers E means that all the Functional dependency in E can be inferred from F (i.e whether F is covering functional dependencies of E)

Computing F Covers E

- We can determine whether F covers E by calculating X+ with respect to F for each FD $X \rightarrow Y$ in E, and then checking whether this X+ includes the attributes in Y

F={A→B, C→E, D→B}

E={C→D, D→E}

Compute C+ & D+ wrt F

Check C+ include D & D+ include E

- Given two sets F and E of FDs for a relation.

E = {A→B, AB→C, D→AC, D→E }

F = {A→BC, D→AE }

Are the two sets equivalent?

Soln : if E = F then We have to check whether E covers F and F covers E.

E Covers F

Given E = {$A \rightarrow B$, $AB \rightarrow C$, $D \rightarrow AC$, $D \rightarrow E$ }

F = {$A \rightarrow BC$, $D \rightarrow AE$ }

$A \rightarrow BC$ A+ = {A $\underline{B}$ $\underline{C}$} A+ includes B and C	**$D \rightarrow AE$** D+ = { D $\underline{A}$ C $\underline{E}$ B} D+ includes A and E

Therefore E covers F

F Covers E

Given E = {A→B, AB→C, D→AC, D→E }

F = {A→BC, D→AE }

A→B A+ = {A $\underline{\text{B}}$ C} A+ includes B **AB→C** AB+ = {A B $\underline{\text{C}}$} AB+ includes C	**D→AC** D+ = {D $\underline{\text{A}}$ E B $\underline{\text{C}}$ } D+ includes A C **D→E** D+= { D A $\underline{\text{E}}$ B C } D+ includes E

Therefore E covers F

Since E Covers F and F covers E ,
E is equivalent to F

Example 2

Check X is equivalent to Y

Given $X = \{A \rightarrow B\}$

$Y = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

Soln : if X = Y then We have to check whether X covers Y and Y covers X.

X covers Y

X= {A → B }

Y = {A→ B, B→ C, A→ C}

A→ B A+ = { A B} A+ includes B **B→C** B+ = { B } B+ does not includes C	**A→C** A+ = { A B} A+ does not includes C

Here X covers Y is not valid So X & Y are not equivalent

MINIMAL COVER

- If a Functional Dependency F is given, then F' is Minimal cover of this FD set if F' does not have
  - Redundant Attributes
  - Redundant Functional Dependency

Steps

1. In Every Functional Dependency right hand side must contain only single attribute
   - eg if A →BC can be applied decomposition rule as A→B ,A→C

2. (a) If Functional Dependency has multiple attributes on LHS, Remove Extraneous/redundant attributes
   - eg : if FD contains F' :{AB→C, ….A→C} the B can be removed

   (b) IF there is any trivial Functional Dependency , that can be removed

   Eg : {AB → B} is trivial since RHS & LHS have attributes in common

3. Remove redundant Functional Dependency By using the transitive rule
   - eg: **E' : {B → A, D → A, B → D}**
   - By using the transitive rule on B → D and D → A, we derive B → A. Hence B → A is redundant and can be removed

Example 1

- Let the given set of FDs be E : {$B \rightarrow A$, $A \rightarrow D$, $AB \rightarrow D$}. Find the minimal cover of E.

Soln :

Step 1 : Check if RHS of Functional Dependency contain only single attribute

Here All above dependencies have only one attribute on the right-hand side , so we have completed step 1.

E : {$B \rightarrow A$, $A \rightarrow D$, $AB \rightarrow D$}.

Step 2 : Check if LHS of Functional Dependency has only single attributes

AB $\rightarrow$ D has two attributes on LHS. Check whether it can be replaced by B$\rightarrow$D or A$\rightarrow$D.

Thus A$\rightarrow$D is redundant will not consider so AB $\rightarrow$ D may be replaced by B $\rightarrow$ D.

E' : {B $\rightarrow$ A, A $\rightarrow$ D, B $\rightarrow$ D}

- **Step 3** Remove redundant Functional Dependency By using the transitive rule

- By using the transitive rule on $B \rightarrow A$ and $A \rightarrow D$, we derive $B \rightarrow D$ Hence $B \rightarrow D$ is redundant and can be removed

Therefore, the minimal cover of E is

E' = {B → A, A → D, }

Example 2

F={A $\rightarrow$B, C$\rightarrow$B, D$\rightarrow$ABC, AC$\rightarrow$D}. Find the minimal cover

STEP 1:

A$\rightarrow$B, C$\rightarrow$B, D$\rightarrow$A, D$\rightarrow$B, D$\rightarrow$C, AC$\rightarrow$D

STEP 2:

Here AC$\rightarrow$D, there is no FD with A$\rightarrow$D or C$\rightarrow$D. so AC$\rightarrow$D must be retained.

A$\rightarrow$B, C$\rightarrow$B, D$\rightarrow$A, D$\rightarrow$B, D$\rightarrow$C, AC$\rightarrow$D

STEP 3:

- In $\underline{A \rightarrow B}$, $C \rightarrow B$, $\underline{D \rightarrow A}$, $D \rightarrow B$, $D \rightarrow C$, $AC \rightarrow D$
- By using the transitive rule $D \rightarrow A$, $A \rightarrow B$ implies $D \rightarrow B$ is redundant and can be removed
- Therefore, the minimal cover of F is $F' = \{A \rightarrow B, C \rightarrow B, D \rightarrow B, D \rightarrow C, AC \rightarrow D\}$

Example 3

- F= {AB → C, C→AB, B→ C, ABC→ AC, A→ C, AC→B}. Find minimal cover

Soln :

STEP1 : Remove multiple attribute from RHS

{AB → C, C→A, C→B, B→ C ,ABC→ A, ABC→ C, A→ C, AC→B}

- STEP2 : Remove multiple attribute from RHS

{AB → C, C→A, C→B, B→ C ,ABC→ A, ABC→ C, A→ C, AC→B}

  - For AB → C there exist A→C, B→C so AB→ C can be removed
  - For AC → B there exist C→ B, so AC → B can be replaced by A→ B
  - ABC→ A, ABC→ C are trivial so can be removed

- After Step 2 FD will be

{C→A, C→B, B→ C ,A→ C, A→B}

{C→A, C→B, B→ C ,A→ C, A→B}

- STEP 3 : Applying transitivity rule

A →C, C→B == > A→B, so A→ B can be removed

- So minimal cover is
  {C→A, C→B, B→ C ,A→ C}

Database Management System
Module 4

Lect 3 :

→ Definition of Keys and Attributes participating in Keys
→ Normalization

Definitions of Keys and Attributes Participating in Keys

SUPERKEY

- A **superkey** of a relation schema *R* is a set of attributes with the property that no two tuples can have same value
- A **key** *K* is a superkey with the additional property that removal of any attribute from *K* will cause *K* not to be a superkey any more.

Book Id	Name	Author
B1	ABC	A1
B2	EFG	A2
B3	XYZ	A3
B4	ABC	A1
B5	PQR	A3

Here following can be super key

<Bookid>

<Bookid,Name>

<Bookid,Author>

< Bookid,Name,Author>

From < Bookid,Name,Author> if we remove Bookid it will no more be a super key because <Name,Author> cannot be used to identify a unique record.

Candidate Key

- It is a Superkey whose subset is not a super key
- It is a minimal super key.

A	B	C
1	1	1
2	1	2
3	2	1
4	2	2
5	3	1

Super key = A
Super Key = A B
Super Key = A C
Super Key = BC
Super Key = ABC

- <u>AB is a Super Key</u>
  - Subset of AB is A,B
  - A is a super key but B is not super key
  - **So AB is not candidate Key**
- <u>AC is a Super Key</u>
  - Subset of AC is A,C
  - A is a super key but C is not super key
  - **So AC is not candidate Key**
- <u>BC is a Super Key</u>
  - Subset of BC is B,C
  - B and C is not super key
  - **So AB is candidate Key**

'A' is a Super Key as well as candidate key because A cannot have subset

- <u>ABC is a Super Key</u>
  - Subset of ABC is A,B,C,AB,AC,BC
  - A is a super key
  - B is not a super key
  - C is not a super key
  - AB is a super key
  - AC is a super key
  - BC is a super key
  - So ABC is not a Super Key
- <u>A is Super Key</u>
  - Since A does not have any subset, it is candidate key
- Therefore Candidate Keys = {A,BC}

- Candidate Key is Minimal super key
  - Minimal means - as minimum as possible
  - Eg : A,B,C is a Superkey
  - Removing C it becomes A, B which is a SK
  - Removing B it becomes A, which is a SK
  - A cannot be further decomposed so A is Candidate Key

Prime & Non Prime Attributes

Prime Attribute

- An attribute of relation schema *R* is called a **prime attribute** of *R* if it is a member of *some candidate key* of *R*
- Eg: Prime Attribute = {A,B,C}

Non Prime Attribute

- An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key

- <u>Primary Key</u>

One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called secondary keys.

Foreign Key

Key that is referring to the primary key of another table

Sl	Name	Dno
1	A	D1
2	B	D2
3	C	D3
4	D	D4
5	E	D5

Dno	DName
D1	CSE
D2	ECE
D3	EEE
D4	AE
D5	CE

Normalization

- **Normalization** is the process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of
  - Minimizing redundancy and
  - Minimizing the insertion, deletion, and update anomalies.

Roll no	Name	Dept	Building	Room no.
1	Rahul	CSE	B1	403
2	Ramesh	CSE	B1	403
3	Sujith	CE	B2	703
4	Nikesk	CE	B2	703
5	Akash	ECE	B1	503
6	Arun	CE	B2	703

- **Insertion Anomaly** – Occurs when certain attribute cannot be inserted into the Relation without the presence of other attributes
- **Update Anomaly** – We cannot modify a particular data without making the same modification for all the similar data.
- **Deletion anomaly -** occurs if we delete a record that may contain attribute that should not be deleted

Solution

- Solution to above problems is to decompose the table into two

Roll no	Name	Dept
1	Rahul	CSE
2	Ramesh	CSE
3	Sujith	CE
4	Nikesk	CE
5	Akash	ECE
6	Arun	CE

Dept	Building	Room no.
CSE	B1	403
CE	B2	703
ECE	B1	503

Normalization

- First Normal Form(1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce Codd Normal Form (BCNF)

Database Management System
Module 4

Lect 4 : FIRST NORMAL FORM (1NF)

FIRST NORMAL FORM

- Domain of an attribute must include only atomic (single, indivisible) values.
- ATOMIC – single/Indivisible

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Three methods to achieve first normal form for such a relation.

Method 1

- Remove the attribute that violates 1NF and place it in a separate relation along with the primary key of the relation

Dname	Dnumber	Mgr_ssn
Research	5	33344555
Administrator	4	987654321
Headquarters	1	888665555

Dnumber	Dlocation
5	Bellaire
5	Sugarland
5	Houston
4	Stafford
1	Houston

Method 2

- 2 Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of department

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

Method 3

3. Create a separate column for each mutli valued attribute.

Dname	Dnumber	Dmg_ssn	Dlocations1	Dlocations 2	Dlocations3
Research	5	33344555	Bellaire	Sugarland	Houston
Administrator	4	987654321	Stafford	Nil	Nil
Headquarters	1	888665555	Houston	Nil	Nil

Finding Candidate Keys

- Given R(ABCD), F= {A→B, B→C, C→A}

1. **Find the closure of all the attributes which will be definitely the superkey**

$(ABCD)^+ = \{A\ B\ C\ D\}$ is a super key

2. Try to minimize the SK by looking at the FD

Given F= {A→B, B→C, C→A}

SK + {A B C D}

Since A→B is given B can be removed from SK

therefore SK = {A,C,D}

To check {A,C,D} is still a SK Find $(ACD)^+$

if it contains all attributes then it is a SK

ACD^+ ={A,C,D,B} .

Therefore {A C D} is a SK

Since A→B, B→C = A→C, C can be removed from SK {A C D}

SK = {A,D}

AD+ = {A D B C}

Therefore {A D } is a Super Key

3. Find the candidate key (CK) by taking the closure of subset of SK and check whether it is not a SK

SK ={A D}

Subset of AD is A, D. Find A+ and B+

A+ = {A, B, C} ---- Not SK

D+ = {D} -----   Not SK

- Therefore **A D is a Candidate Key**
- **Prime attributes = {A,D}**

4. Check if any more Candidate Keys are there – If any of the Prime attribute is present in RHS of Functional dependency

Prime attributes = {A,D}

F= {A→B, B→C, C→A}

There can be more CK

From SK = {A D} replace A with C

So SK = {C D}

Closure of subset of CD

C+ = {C A B} ----- Not SK

D+ = { D} ----- Not SK

Therefore CD is a Candidate Key

Prime Attribute {C, D}

Prime Attribute = {C, D}

F= {A→B, B→C, C→A}

There can be more CK

In SK ={C D} replace C with B

So SK = {B D}

Closure of subset of BD

B+ = {B C A} ----- Not SK

D+ = { D} ----- Not SK

Therefore BD is a Candidate Key

Prime Attribute {B, D}

Prime Attribute = {B, D}

F= {A→B, B→C, C→A}

There can be more CK

In SK ={B D} replace B with A

So SK = {A D}

We have already found {A D } as CK

There fore candidate keys are =

CK ={AD, CD, BD}

Prime Attributes = {A,B,C,D}

Example 2

R(ABCD) F= {A→B, D→E}. Find candidate key

(ABCD)+ = {A B C D} ---- SuperKey

Since A→B , B can be removed

SK = {A C D}

Checking {A C D} is CK

Subset of closure of A C D is

A+ = { A B } ----- Not SK

C+ = {C} --------- Not SK

D+ = {D E} --------Not SK

AC+ = { A C B } -- Not SK

AD+ = {A D E } --- Not SK

CD+ = { C D E} --- Not SK

Therefore {A C D } is candidate Key

Prime Attribute = {A, C, D}

There can only one Candidate key because RHS of FD does not contain any Prime attribute

Example 3

R(ABCD) with FD {AB → CD , D→B, C→A}

Soln

Database Management System
Module 4

Lect 5 : SECOND NORMAL FORM (2NF)

SECOND NORMAL FORM (2NF)

- **Second normal form (2NF)** is based on the concept of full functional dependency.
- A functional dependency $X \rightarrow Y$ is a **FULL FUNCTIONAL DEPENDENCY** if removal of any attribute A from X means that the dependency does not hold any more; that is, $(X - \{A\})$ does *not* functionally determine Y.
- Eg:, {Ssn, Pnumber} →Hours is a full dependency (Ssn →Hours or Pnumber→Hours does not holds)

- A functional dependency $X \rightarrow Y$ is a **PARTIAL DEPENDENCY** if some attribute A can be removed from X and the dependency still holds; that is, $(X - \{A\}) \rightarrow Y$.
- the dependency {Ssn, Pnumber}→Ename is partial because Ssn→Ename holds.

Definition 1:

There are mainly two condition for 2NF

1. The Relation must be in First Normal Form .
2. There is no Partial Dependency Exist
   - Partial Dependency means that Proper subset of candidate key will determine the Non Prime Attribute
   - I.e **Proper subset of candidate key → Non Prime Attribute** **should** not exist

Example 1

- Consider a Relation R(A,B,C,D,E,F) with FDs { AD→C, A→B, D→E} Suppose the candidate key is A & D the non prime attributes will be B,C,E & F
- Soln :

AD→C (Satisfies 2NF bcaz NPA C is Fully Functional Dependent on all Candidate keys)

A→B (violates 2NF bcaz subset of candidate key A try to determine a NPA)

D→E (violates 2NF bcaz subset of candidate key A try to determine a NPA)

Example 2

- Consider a Relation R(A,B,C,D,E,F) with Functional Dependencies {A→B, B→C, C→D, D→E}

Soln :

<u>Finding Candidate Key</u>

$(ABCDEF)^+$ = ABCDEF ----→ is a Superkey(SK)

Since A→B so B can be removed from the SK

SK =ACDEF

Since A→B,B→C ➔ A→C, C is removed

SK = ADEF

Since A→C, C→D ➔ A→D so D can be removed

SK = AEF

Since A→D, D→E ➔ A→E so E can be removed

SK = AF

- Checking if SK {A F} be candidate key

(Check closure subset of candidate key not a SK)

Given FD ={$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$}

A^+ = {A B C D E} --- not SK

F^+= {F} ---- not SK

So A F is a candidate key

Prime Attributes = {A, F}

Non Prime Attributes = {B,C,D,E}

Since Prime attributes are not found in RHS of FD , there will not be any more candidate keys

Checking for 2NF

1. The relation is already in 1NF

2. To Check for Partial Dependency check whether **Proper subset of candidate key → Non Prime Attribute** does not exit

Here the given Functional Dependency are {A→B, B→C, C→D, D→E}

Candidate Key= {AF}

- Non Prime Attribute= **{B,C,D,E}**

A→B violates 2NF (CK subset → NPA)

B→C satisfies 2NF (NPA → NPA)

C→ D satisfies 2NF (NPA → NPA)

D→ E satisfies 2NF (NPA → NPA)

To Decompose this relation to make it 2NF

- Given R(A B C D E F)
- Here the Functional Dependency $A \rightarrow B$ does not satisfy 2NF so the relation R has to decomposed into two R1 & R2
  - R1(A B C D E F) with FDs $\{B \rightarrow C, C \rightarrow D, D \rightarrow E\}$
  - R2(A B) with FD $\{ A \rightarrow B \}$

Eg 2: Check whether the following relation is in 2NF

R(A,B,C,D,E,F) FD= {A→BCDEF, BC→ADEF, B→F, D→E}

Soln :

$(ABCDEF)^{+}$ = ABCDEF is Superkey

Since A→BCDEF so therefore BCDEF can be removed

Superkey = {A}

Since superkey does not have any subset {A} can be candidate key

Therefore candidate key = {A}

Prime Attribute = {A}

Non Prime Attribute = {B,C,D,E,F}

FD= {A→BCDEF, BC→ADEF, B→F, D→E}

- Since prime Attribute {A} is present in RHS of BC→ADEF so there will be more candidate keys

SK {A } can be replaced by {B C}

SK = {B C}

<u>Checking BC as candidate Key</u>

(closure of subset of SK is not a SK)

B+ = {B F} ---- not SK

C+ = { C }------ not SK

Therefore { B C } is a candidate key

Prime attribute = {B,C}

There is no more candidate keys

Therefore

Candidate keys = {A B C}

Prime Attribute = {A, B, C}

Non Prime Attribute = {D, E, F}

- <u>Checking for 2NF</u>

Given FD= {A→BCDEF, BC→ADEF, B→F, D→E}

Candidate key = {A B C}

Prime Attribute = {A, B, C}

Non Prime Attributes {D,E,F}

- A→BCDEF --- satisfies 2NF (CK → NPA)
- BC→ADEF --- satisfies 2NF (CK → NPA)
- B→F ---- violates 2NF (subset of CK → NPA}
- D→ E ---- satisfies 2NF (NPA→ NPA)

- Table decomposition to make it 2NF

Table R(A,B,C,D,E,F)

FD= {A→BCDEF BC→ADEF, B→F, D→E}

can be decomposed into

R1(A,B,C,D,E) with FD: A→BCDE, BC→ADE, D→E

R2 (B,F) with FD: B→ F

Database Management System
Module 4

Lect 6 :
->Second Normal Form (3NF)
-> Boyce Codd Normal Form (BCNF)

THIRD NORMAL FORM (3NF)

- **Third normal form (3NF)** is based on the concept of *transitive dependency*
- When a NPA dependent on NPA (NPA→NPA). This is known as **Transitive Dependency**

Definition 1

- **A relation schema R is in 3NF**
  - If it satisfies 2NF
  - Transitive Dependency does not exist
    i.e no Non Prime Attribute is dependent on another Non Prime Attribute (**NPA → NPA**)

Definition 2

- **A Relation R is in 3NF if a non trivial Functional Dependency X→Y then**
  - **X is a Super key**

    or
  - **Y is a Prime attribute**

- Consider a Relation R(A,B,C,D) with FD : {A→B, B→C, C→D}. Check whether the relation is in 3NF

Soln :

<u>Finding candidate keys</u>

$(ABCD)^+$ ={A,B,C,D} so ABCD is Super key

Since A→B then B can be removed from super key

Since A→B, B→C => A→C then C can be removed from super key

Since A→C, C→D => A→D then D can be removed from super key

Finally the Superkey becomes (A)

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

The super key A does not have further subset so A is a candidate key

Prime Attribute ={A}

There is no more candidate keys because RHS of FD does not have any prime attributes

Candidate Key= A

Prime Attribute ={A}

Non Prime Attribute ={B,C,D}

- <u>Checking for 2NF</u>

FD : {A→B, B→C, C→D}.

Candidate Key= A

Prime Attribute ={A}

Non Prime Attribute ={B,C,D}

A→B --- Satisfies 2NF (CK → NPA)

B→ C --- Satisfies 2NF (NPA → NPA)

C→ D ---- Satisfies 2NF (NPA → NPA)

- <u>Checking for 2NF</u>

FD : {A→B, B→C, C→D}.

Candidate Key= A

Prime Attribute ={A}

Non Prime Attribute ={B,C,D}

A→B --- Satisfies 3NF (CK → NPA)

B→ C --- Violates 3NF (NPA → NPA)

C→ D ---- Violates 3NF (NPA → NPA)

Table decomposition

R(ABCD) is decomposed into

R1(<u>A</u> B C) with FD : $A \rightarrow B$

R2(<u>B</u> C) with FD : $B \rightarrow C$

R3(<u>C</u> D) with FD : $C \rightarrow D$

Eg 2: Check whether the following relation is in 3NF

$R(A,B,C,D,E,F)$ FD= $\{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow F, D \rightarrow E\}$

Soln :

$(ABCDEF)^+ = ABCDEF$ is Superkey

Since $A \rightarrow BCDEF$ so therefore BCDEF can be removed

Superkey = {A}

Since superkey does not have any subset {A} can be candidate key

Therefore candidate key = {A}

Prime Attribute = {A}

Non Prime Attribute = {B,C,D,E,F}

FD= {A→BCDEF, BC→ADEF, B→F, D→E}

Since prime Attribute {A} is present in RHS of BC→ADEF so there will be more candidate keys

SK {A } can be replaced by {B C}

Therefore SK = {B C}

Checking BC as candidate Key

(closure of subset of SK is not a SK)

B+ = {B F} ---- not SK

C+ = { C }------ not SK

Therefore { B C } is a candidate key

Prime attribute = {B,C}

There is no more candidate keys

Therefore

Candidate keys = {A B C}

Prime Attribute = {A, B, C}

Non Prime Attribute = {D, E, F}

- <u>Checking for 2NF</u>

Given FD= {A→BCDEF, BC→ADEF, B→F, D→E}

Candidate key = {A B C}

Prime Attribute = {A, B, C}

Non Prime Attributes {D,E,F}

- A→BCDEF --- satisfies 2NF (CK → NPA)
- BC→ADEF --- satisfies 2NF (CK → NPA)
- B→F ---- violates 2NF (subset of CK → NPA}
- D→ E ---- satisfies 2NF (NPA→ NPA)

- Table decomposition to make it 2NF

Table R(A,B,C,D,E,F)

FD= {$A \rightarrow BCDEF$ $BC \rightarrow ADEF$, $B \rightarrow F$, $D \rightarrow E$}
can be decomposed into

R1(A,B,C,D,E) with FD: $A \rightarrow BCDE$, $BC \rightarrow ADE$, $D \rightarrow E$

R2 (B,F) with FD: $B \rightarrow F$

Checking for 3NF

R1(A,B,C,D,E) with FD : $A \rightarrow BCDE$, $BC \rightarrow ADEF$ $D \rightarrow E$

Candidate key = {A B C}

Prime Attribute = {A, B, C}

Non Prime Attributes {D,E,F}

- $A \rightarrow BCDE$ --- satisfies 3NF (CK → NPA)
- $BC \rightarrow ADE$ --- satisfies 3NF (CK → NPA)
- D→ E ---- violates 3NF (NPA→ NPA)

- R2 (B,F) with FD: B→ F

Candidate key = {A B C}

Prime Attribute = {A, B, C}

Non Prime Attributes {D,E,F}

- B→ F --- satisfies 3NF (CK → NPA)

Table decomposition

R1(A,B,C,D,E) and R2 R2 (B,F) is decomposed into

- R11(A B C D) with FD : $A \rightarrow BCD$, $BC \rightarrow AD$
- R12(D E) with FD : $D \rightarrow E$
- R2(B F) with FD : $B \rightarrow F$

BOYCE-CODD NORMAL FORM (BCNF)

- Stricter than 3NF
- Every relation in BCNF is also in 3NF; however, a relation in 3NF is not necessarily in BCNF

Definition

- **A relation schema R is in BCNF if whenever a non-trivial functional dependency $X \rightarrow Y$, then X is a superkey of R.**
- In other words, a relation must only have candidate key in the LHS

i.e **If X → Y**

Then X must be SuperKey

Eg 1 Consider a relation R(A,B,C) with Functional Dependency {A→B, B→C, C→A}

Finding Candidate Keys

(ABC)+ = {ABC}

since A→B, B can be eliminated

Since A→B, B→C => A→C so C can be eliminated

so the superkey becomes (A)

A+ ={ABC}

So A is a Candidate Key.

Prime Attribute = {A}

There are more candidate keys because A is available in the RHS of Functional Dependency **C**→A **.**

A can be replaced by C

SK = C

So C is a Candidate Key.

Prime Attribute = {C}

There are more candidate keys because C is available in the RHS of Functional Dependency **B**→C

So C can be replaced by B

SK = B

so B is a candidate key

- Candidate Key = {A B C}
- Prime Attribute = {A,B,C}
- Non Prime Attribute = nil

- Check for BCNF

FD : {A→B, B→C, C→A}

Candidate Key = {A B C}

Prime Attribute = {A,B,C}

A→B satisfies BCNF (A is a Superkey)

B→ C satisfies BCNF (B is a Superkey)

C→ A satisfies BCNF (C is a Superkey)

Therefor the relation is in BCNF