25CS305
Database Management Systems

M. NARMADHA,

Assistant Professor,

Computer Science and Engineering

UNIT-IV

1. Schema Refinement:

- The Schema Refinement refers to refine the schema by using some technique. The best technique of schema refinement is **decomposition.**
- ***Normalization or Schema Refinement is a technique of organizing the data in the database.***
- ***It is a systematic approach of decomposing tables to eliminate data redundancy*** and undesirable characteristics like Insertion, Update and Deletion Anomalies.

- ***Redundancy*** refers to repetition of same data or duplicate copies of same data stored in different locations.

- **Anomalies:** Anomalies refers to the problems occurred after poorly planned and unnormalized databases where all the data is stored in one table which is sometimes called a flat file database.

- **Anomalies or problems facing without normalization (problems due to redundancy) :**

- Anomalies refers to the problems occurred after poorly planned and unnormalized databases where all the data is stored in one table which is sometimes called a flat file database.

- Let us consider such type of schema –

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k
Primary Key(SID,CID)				

- Here all the data is stored in a single table which causes redundancy of data or say anomalies as SID and Sname are repeated once for same CID . Let us discuss anomalies one by one.
- Due to redundancy of data we may get the following problems, those are-

1. **insertion anomalies :** It may not be possible to store some information unless some other information is stored as well.
2. **redundant storage:** some information is stored repeatedly

3. update anomalies: If one copy of redundant data is updated, then inconsistency is created unless all redundant copies of data are updated.

4. deletion anomalies: It may not be possible to delete some information without losing some other information as well.

- **Problem in updation / updation anomaly** –
- If there is updation in the fee from 5000 to 7000, then we have to update FEE column in all the rows, else data will become inconsistent.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	~~5k~~
S2	A	C1	C	~~5k~~
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

7k
7k

Costly Operation

More IO Cost

- **Insertion Anomaly and Deletion Anomaly-** These anomalies exist only due to redundancy, otherwise they do not exist.

- **Insertion Anomalies**: New course is introduced C4, But no student is there who is having C4 subject.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

NULL	NULL	CA	DB	12k

To Insert that Row, It is Required to Put Dummy Data..

- Because of ... r dummy data.

Therefore,

xx	xx	CA	DB	12k

Deletion Anomaly :

Deletion of S3 student cause the deletion of course.

Because of deletion of some data forced to delete some other useful data.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
~~S3~~	~~B~~	~~C2~~	~~C~~	~~10k~~
~~S3~~	~~B~~	~~C2~~	~~JAVA~~	~~15k~~

- Solutions To Anomalies : Decomposition of Tables – Schema Refinement

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

SID	Sname	CID
S1	A	C1
S2	A	C1
S1	A	C2
~~S3~~	~~B~~	~~C2~~
~~S3~~	~~B~~	~~C3~~

PK(SID,CID)

Deletion Anamoly Removed

CID	CNAME	FEE
C1	C	~~5k~~ 7k
C2	C	10k
C3	JAVA	15k
C4	DB	12k

(Updation Anamoly Removed

Insertion Anamoly Removed

PK(CID)

There are some Anomalies in this again –

Updation Anamoly

SID	Sname	CID
S1	~~A~~ (AA)	C1
S2	~~A~~ (AA)	C1
S1	~~A~~ (AA)	C2
S3	B	C2
S3	B	C3
S4	B	xx

A student having no course is enrolled. We have to put dummy data again.

Deletion Anamoly as C2 course is alloted to some students

CID	CNAME	FEE
C1	C	5k
~~C2~~	~~C~~	~~10k~~
C3	JAVA	15k
C4	DB	12k

- ***What is the Solution ??***
- ***Solution : decomposing into relations as shown below***

R1

SID	Sname

R2

SID	CID

R3

CID	Cname	Fee

->TO AVOID REDUNDANCY and problems due to redundancy, we use refinement technique called **DECOMPOSITION.**

Decomposition:- Process of decomposing a larger relation into smaller relations.

->Each of smaller relations contain subset of attributes of original relation.

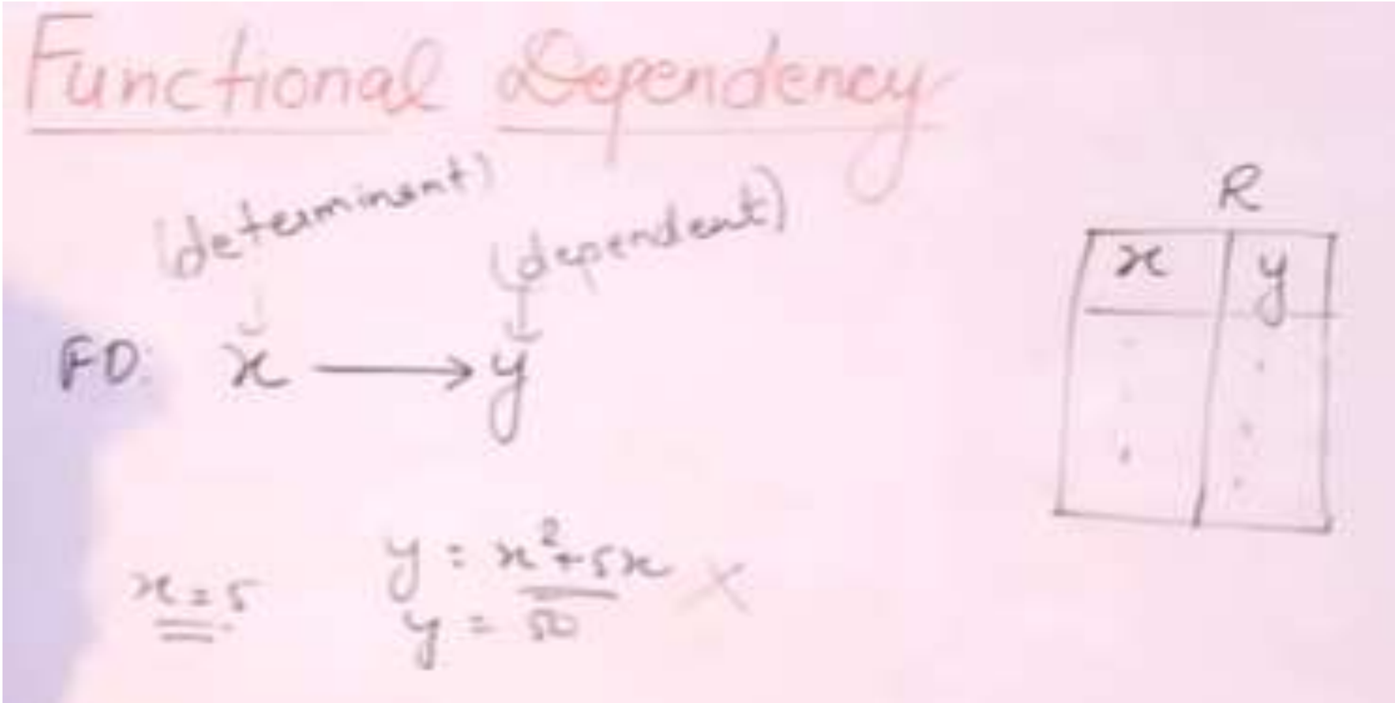
- **Functional dependencies:**

→ Functional dependency is a relationship that exist when one attribute uniquely determines another attribute.

→ Functional dependency is a form of integrity constraint that can identify schema with redundant storage problems and to suggest refinement.

→ A functional dependency A→B in a relation holds true if two tuples having the same value of attribute A also have the same value of attribute B

- **IF t1.X=t2.X then t1.Y=t2.Y** where t1,t2 are tuples and X,Y are attributes.


Functional Dependency
(determinant)
(dependent)
FD: x → y
R
x
y
x=5
y = x2+5x
y = 50

X	Y
1	1
2	1
3	2
4	3

X	Y
1	1
2	1
3	2
4	3

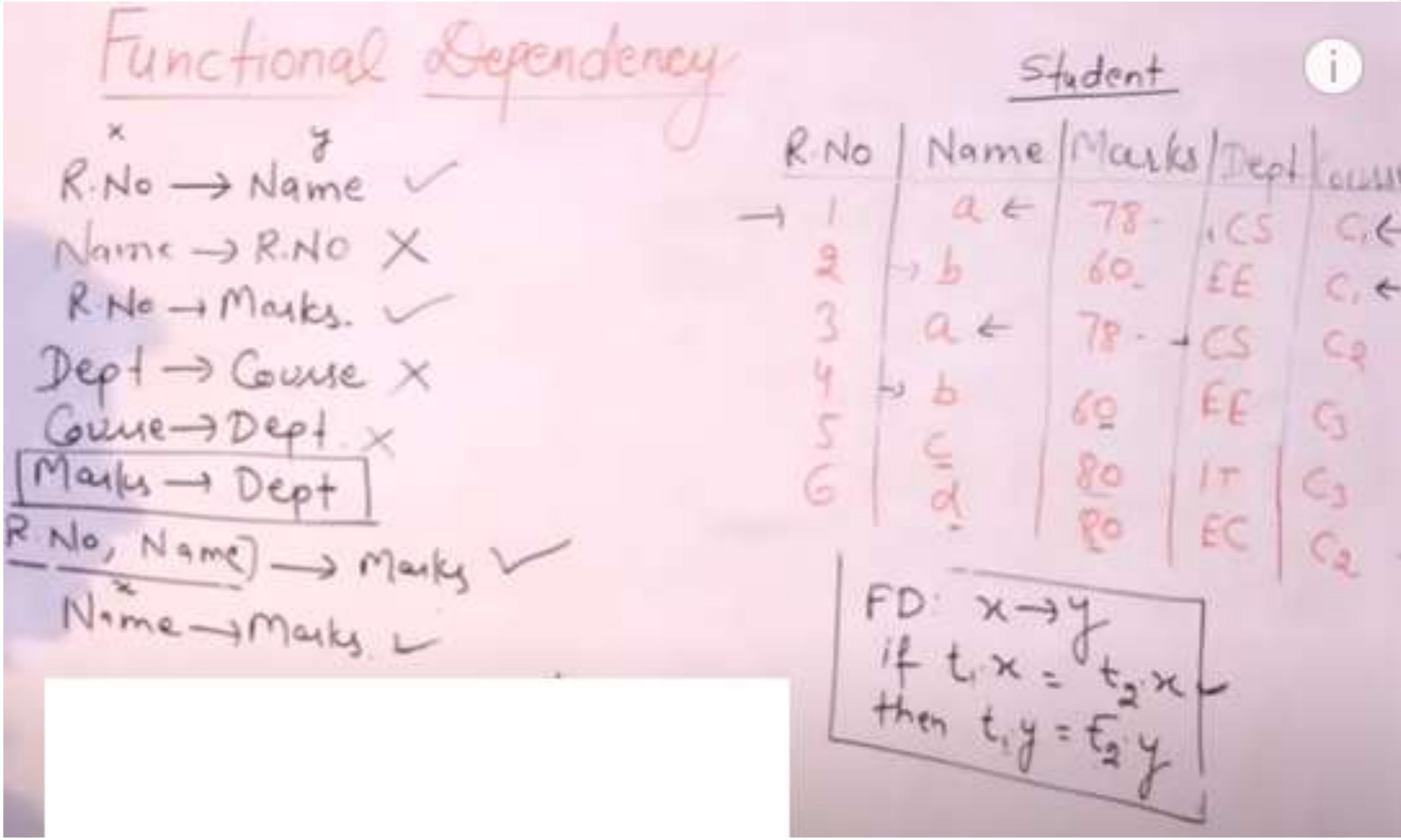
X	Y
1	1
2	1
3	2
4	3
2	5

$\underline{x = 5}$ $\quad y = x^2$

$y = 25$

$\rightarrow x = 5 \quad y = \underline{25}$

$- x = \underline{5} \quad y = 25 > 25$


Functional Dependency
R.No → Name ✓
Name → R.No ✗
R.No → Marks. ✓
Dept → Course ✗
Course → Dept ✗
Marks → Dept
R.No, Name → Marks ✓
Name → Marks ✓
Student
R.No | Name | Marks | Dept | Course
1 a 78 CS C1
2 b 60 EE C1
3 a 78 CS C2
4 b 60 EE C3
5 c 80 IT C3
6 d 80 EC C2
FD: x → y
if t1.x = t2.x
then t1.y = t2.y

- **Reasoning about functional dependencies:**
- **Armstrong Axioms :**
- Armstrong axioms defines the set of rules for reasoning about functional dependencies and also to infer all the functional dependencies on a relational database.

- **Types of Functional Dependency**
  - Trivial
  - Non Trivial

Types of functional dependencies:

1) **Trivial functional dependency**:-If $X \rightarrow Y$ is a functional dependency where Y subset X, these type of FD's called as trivial functional dependency.

2) **Non-trivial functional dependency**:-If $X \rightarrow Y$ and Y is not subset of X then it is called non-trivial functional dependency.

3) **Completely non-trivial functional dependency**:-If $X \rightarrow Y$ and $X \cap Y = \Phi$(null) then it is called completely non-trivial functional dependency.

Armstong axioms rules or inference rules:

Primary axioms:

1. Axiom of reflexivity –

If A is a set of attributes and B is subset of A, then A holds B. If $B \subseteq A$ then $A \to B$ This property is trivial property.

- FD: X → Y
- If y⊆ X
- X → X

Armstong axioms rules or inference rules:

Primary axioms:

2. **Axiom of augmentation –**

If $A \to B$ holds and Y is attribute set, then $AY \to BY$ also holds. That is adding attributes in dependencies, does not change the basic dependencies. If $A \to B$, then $AC \to BC$ for any C.

- If X → Y then XA → YA

Rno -> Name then (Rno,Marks) -> (Name, Marks)

ROLLNO	NAME	MARKS	DEPT

Armstong axioms rules or inference rules:

Primary axioms:

3. **Axiom of transitivity –**

   Same as the transitive rule in algebra, if $A \to B$ holds and $B \to C$ holds, then $A \to C$ also holds. $A \to B$ is called as A functionally that determines B. If $X \to Y$ and $Y \to Z$, then $X \to Z$

- If (X → Y & Y → Z) then X → Z

- Ram is sibling of syam
- Syam is sibling of mohan

Name → Marks
Marks → Dept

Name → Dept ✓

RNo	Name	Marks	Dept	Course
1	a	78	CS	C_1
2	b	60	EE	C_1
3	a	78	CS	C_2
4	b	60	EE	C_3
5	c	80	IT	C_2

RNo	Name	Marks	Dept	Course
1	a	78	CS	C_1
2	b	60	EE	C_1
3	a	78	CS	C_2
4	b	60	EE	C_3
5	c	80	IT	C_2
6	d	80	EC	C_4

secondary or derived axioms:

1. **Union –**
   If $A \to B$ holds and $A \to C$ holds, then $A \to BC$ holds. If $X \to Y$ and $X \to Z$ then $X \to YZ$

- If X → Y & X → Z then X → Y Z

Rno -> Name & Rno -> Marks then Rno ->(Name,Marks)

ROLLNO	NAME	MARKS	DEPT

secondary or derived axioms:

3. Decomposition -

If $A \rightarrow BC$ holds then $A \rightarrow B$ and $A \rightarrow C$ hold. If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

- If $X \rightarrow Y Z$ then $X \rightarrow Y$ & $X \rightarrow Z$
- If $XY \rightarrow Z$ then $X \rightarrow Z$ & $Y \rightarrow Z$

ROLLNO	NAME	MARKS	DEPT

secondary or derived axioms:

4. Pseudo Transitivity -

If $A \rightarrow B$ holds and $BC \rightarrow D$ holds, then $AC \rightarrow D$ holds. If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$.

- If ($X \rightarrow Y$ and $YZ \rightarrow A$ then $XZ \rightarrow A$

Rno -> Name & Name,Marks -> Dept then Rno,Marks -> Dept

ROLLNO	NAME	MARKS	DEPT

secondary or derived axioms:

2. Composition –

If $A \rightarrow B$ and $X \rightarrow Y$ holds, then $AX \rightarrow BY$ holds.

ROLLNO	NAME	MARKS	DEPT

- **Candidate Key:**
- Candidate Key is minimal set of attributes of a relation which can be used to identify a tuple uniquely.
- Consider student table:

student(sno, sname,sphone,age)

- we can take **sno** as candidate key. we can have more than 1 candidate key in a table. types of candidate keys:

1. simple(having only one attribute)
2. composite(having multiple attributes as candidate key)

- **Super Key:**
- Super Key is set of attributes of a relation which can be used to identify a tuple uniquely.
-
  - Adding zero or more attributes to candidate key generates super key.
  - A candidate key is a super key but vice versa is not true.
-
- Consider student table: student(<u>sno</u>, sname,sphone,age) we can take sno, (sno, sname) as super key

- **ATTRIBUTE CLOSURE/CLOSURE SET**
- **Attribute closure:** Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.
- NOTE:
- To find attribute closure of an attribute set-

1)add elements of attribute set to the result set.

2)recursively add elements to the result set which can be functionally determined from the elements of result set.

- **Attribute closure:**
- **To find all the candidate keys present in the relation we need to find attribute closure**
- **We can find the candidate keys using the Armstrong Axiom rules but that would be lengthy process and difficult process**

- **R(A,B,C,D,E)**
- **FD { A->B, B->C, C->D, D->E}**

A ->C

A-> A

A-> D

A-> E

A-> ABCDE

B -> D

B -> E

B -> B

B -> BCDE

C -> E

C -> C

C -> CDE

E -> E

- **ATTRIBUTE CLOSURE/CLOSURE SET**
- **X+ -> contains set of attributes determined by X**
- **A+ = {A, B, C, D, E}**
- **AD+ = {A, D, B, C, E}**
- **B+ = {B,C, D, E}**
- **CD+ = { C, D, E}**

FD { A->B, B->C, C->D, D->E}

A -> ABCDE

AD ->BD

AD -> B

AD -> D

- **ATTRIBUTE CLOSURE/CLOSURE SET**
- **X+ -> contains set of attributes determined by X**

FD { A->B, B->C, C->D, D->E}

SK
- **A+ = {A, B, C, D, E}**

SK
- **AD+ = {A, D, B, C, E}**
- **B+ = {B,C, D, E}**
- **CD+ = { C, D, E}**

Super Key: Set of attributes whose closure contains all attributes of a given relation

- **AB+ = {A,B,C,D,E}** SK

if we have 'N' attributes with one candidate key then the number of possible superkeys is $2^{(N - 1)}$

- **ATTRIBUTE CLOSURE/CLOSURE SET**
- **X+ -> contains set of attributes determined by X**
- SK **A+ = {A, B, C, D, E}** CK
- SK **AD+ = {A, D, B, C, E}**

FD { A->B, B->C, C->D, D->E}

Candidate Key: super key whose proper subset is not a super key

- **R(A,B,C,D,E)**
- **FD { A->B, D->E}**
- **ABCDE+ = {A,B,C,D,E} (ABCDE IS SUPER KEY)**
- **STEPS:**

1. **Discard one by one attribute from the super key**
2. **From prime attributes check your prime attribute are available on the right hand side of the functional dependency**

- **Only one candidate key is available: ACD**

- R(A,B,C,D,E)   FD { A->B, D->E}
- SK - $ABCDE^{+}$ = {A,B,C,D,E} (ABCDE IS SUPER KEY)

- SK – ACDE = {A,B,C,D,E} B is discarded
- SK – ACD = {A,B,C,D,E} C is discarded

Rule: 1 Discard one by one attribute from the super key

- Proper subsets are AC^{+}, CD^{+}, AD^{+}, A^{+}, C^{+}, D^{+}

- Prime Attributes: A,C,D

Candidate Key: super key whose proper subset is not a super key

- **R(A,B,C,D,E)   FD { A->B, D->E}**
- Prime Attributes: A,C,D

Rule: 2. From prime attributes check your prime attribute are available on the right hand side of the functional dependency

Only one candidate key is available: ACD

- **Example 2:**
- **R(A,B,C,D)   FD { A->B, B -> C, C->A}**
- **SK - ABCD$^+$ = {A,B,C,D}**
- **SK - ACD$^+$ = {A,C,D,B} B is discarded**
- **SK - AD$^+$ = {A,D,B,C} C is discarded**
- **Proper Subsets: A$^+$, D$^+$**
- **Prime Attributes: AD**

Candidate Key: super key whose proper subset is not a super key

A$^+$ = {A,B,C}

D$^+$ = {}

Proper subset is not a super key so AD is candidate key

- **Example 2:**
- **R(A,B,C,D)   FD { A->B, B -> C, C->A}**
- **Prime Attributes: AD**

Rule: 2. From prime attributes check your prime attribute are available on the right hand side of the functional dependency

- **Prime Attributes:**
- **AD**
- CD (A->C)

- **Now Prime Attributes are: ADC**

↓ replaces

C^+ = {C,A,B}

D^+ = {}

Proper subset is not a super key so CD is candidate key

- **Example 2:**
- **R(A,B,C,D) FD { A->B, B -> C, C->A}**
- **Now Prime Attributes are: ADC**
- **Check prime attributes are right side of Functional Dependency**
- **AD**
- **CD**
- **BD**
- **Now Prime Attributes are: ADCB**

↓ replaces

- **Check prime attributes are right side of Functional Dependency**

↓ replaces

$B^+ = \{B,C,A\}$

$D^+ = \{\}$

Proper subset is not a super key so BD is candidate key

- **Example 2:**
- **R(A,B,C,D)   FD { A->B, B -> C, C->A}**
- **Now Prime Attributes are: ADCB**
- **Check prime attributes are right side of Functional Dependency**
- **AD**
- **CD**
- **BD**
- **AD**
- **No. of candidate keys are 3 (AD, CD, BD)**
- **Prime Attributes are ABCD**

↓ replaces

↓ replaces

↓ replaces

- R(A,B,C,D)
- FD: {AB->CD, D->B, C->A
- SK ABCD+ = {A,B,C,D}
- SK AB+ = {A,B,C,D}
- A+ ={A}
- B+ ={B}
- AB, AD, CD, BC are candidate keys
- Prime Attributes : A,B,D, C
- AD+
- D+ ={D,B}
- CB+

- Candidate Keys:
- AB, AD, BC, CD

- **Example 3: find all candidate key**
- **R(A,B,C,D,E,F)**
- **FD { AB->C, C-> DE, E->F, D->A, C->B}**
- **SK - $ABCDEF^+$ = {A,B,C,D,E,F}**
- **SK - $ABDEF^+$ = {A,B,C,D,E,F} C is discarded**
- **SK - ABF^+ = {A,B,F,C,D,E} DE is discarded**
- **SK - AB^+ = {A,B,F,C,D,E} F is discarded**

C->DE by spliting C-> D, C->E

AB->C, C->E, E->F then AB->F

A^+ = {A}

B^+ = {B}

Proper subset is not a super key so AB is candidate key

- **Example 3: find all candidate key**
- **R(A,B,C,D,E,F)**
- **FD { AB->C, C-> DE, E->F, D->A, C->B}**
- **Prime Attributes are: A,B**

Rule: 2. From prime attributes check your prime attribute are available on the right hand side of the functional dependency

- **AB**
- **DB**
- **Now Prime Attributes are: A,B,D**

↓ replaces

D^+ = {D,A}

B^+ = {B}

Proper subset is not a super key so DB is candidate key

- **Example 3: find all candidate key**
- **R(A,B,C,D,E,F)**
- **FD { AB->C, C-> DE, E->F, D->A, C->B}**
- **Prime Attributes are: A,B,D**
- **AB**
- **DB**

replaces

replaces

- **AC**

A^+ = {A}

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

Now C is a super key so AC is not a candidate key

But C is candidate Key

- Example 3: find all candidate key
- R(A,B,C,D,E,F)
- FD { AB->C, C-> DE, E->F, D->A, C->B}
- Prime Attributes are: A,B,D,C
- AB
- DB
- CB

replaces

- AB->C

replaces

- We need to replace C with AB but we already done AB
- Candidate Keys: AB, BD, C

$B^+ = \{B\}$

$C^+ = \{C,D,E,F,A,B\}$

Now C is a super key so CB is not a candidate key

- **Prime and non-prime attributes**
- Attributes which are parts of any candidate key of relation are called as prime attribute, others are non-prime attributes.

Normalization

STUDENT

Sid	Sname	Credits	Dept. name	Building	Room No
1	Rahul	5	CSE	B1	101
2	Jiya	8	CSE	B1	101
3	Vara	9	ECE	B2	201
4	Priya	9	ECE	B2	201
5	Ankur	7	CE	B1	110
6	Aakash	7	EEE	B1	115
7	Vanshika	8	CE	B1	110
8	Rev	7	CSE	B1	101

Normalization

STUDENT

Sid	Sname	Credits	Dept. name
1	Rahul	5	CSE
2	Jiya	8	CSE
3	Vara	9	ECE
4	Priya	9	ECE
5	Ankur	7	CE
6	Aakash	7	EEE
7	Vanshika	8	CE
8	Rev	7	CSE

Dept. name	Building	Room No
CSE	B1	101
ECE	B2	201
CE	B1	110
EEE	B1	115

- **Normalization:**
- Normalization is a process of designing a consistent database with minimum redundancy which support data integrity by grating or decomposing given relation into smaller relations preserving constraints on the relation.
- Def: normalization is a process of making the table free from insertion, updation and deletion anolomalies and save space by reducing redundant data

First Normal Form

- Rule 1: Each column must have atomic values
- Rule 2: A Column should contain value from same domain or same type
- Rule 3: Each Column should have a unique name
- Rule 4: No ordering to rows & columns
- Rule 5: no duplicate rows

First Normal Form

- Student

Sid	Sname	S.Addr	P.No
1	Vara	AP, INDIA	P1, p2
2	Prasad	TN, INDIA	P3
3	Audi	Punjab, INDIA	P4, p5
4	Siva	Raj, INDIA	p8

- This is not in first normal form because the above table is having multivalued and composite attribute

First Normal Form

- Rule 1:
- The relation is in first normal form if and only if all the attributes are atomic domains
- Or
- Each column must have atomic values
- Atomic – which cannot be divided

Sid	Sname	S.Addr	P.No
1	Vara	AP, INDIA	P1, p2
2	Prasad	TN, INDIA	P3
3	Audi	Punjab, INDIA	P4, p1
4	Siva	Raj, INDIA	p8

Sid	Sname	State	Country	P.No
1	Vara	AP	INDIA	P1
1	Vara	AP	INDIA	P2
2	Prasad	TN	INDIA	P3
3	Audi	Punjab	INDIA	P4
3	Audi	Punjab	INDIA	p1

First Normal Form

- Rule 2: A Column should contain value from same domain or same type
- Rule 3: Each Column should have a unique name
- Rule 4: No ordering to rows & columns
- Rule 5: no duplicate rows

Sid	Sname	State	Country
1	Vara	AP	INDIA
1	Vara	AP	INDIA
2	Prasad	TN	INDIA
3	Audi	Punjab	INDIA
3	Audi	Punjab	INDIA

Sid	P.No
1	P1
1	P2
2	P3
3	P4
3	p1

Second Normal Form

- Rule 1: it is in First Normal Form
- Rule 2: No Partial Dependency in the relation
- Partial Dependency: Proper subset of candidate key -> non prime attributes
- Example: R(A,B,C,D,E,F)
- FD= {A->B, B->C, C->D, D->E}

Second Normal Form

- Example: R(A,B,C,D,E,F)
- FD= {A->B, B->C, C->D, D->E}
- $ABCDEF^+ = \{A,B,C,D,E,F\}$
- $AF^+ = \{A,F,B,C,D,E\}$
- AF is candidate key
- Prime Attributes: A,F
- Non Prime Attributes: B,C,D,E

$A^+ = \{A,B,C,D,E\}$

$F^+ = \{F\}$

Proper subset is not a super key so AF is candidate key

Second Normal Form

- Example: R(A,B,C,D,E,F)
- FD= {A->B, B->C, C->D, D->E}
- Prime Attributes: A,F
- Non Prime Attributes: B,C,D,E

Partial Dependency: Proper subset of candidate key -> non prime attributes

Proper subset of AF

A-> B Partial Dependency

F ->

So the given relation is not in second normal form

Second Normal Form

- Example:R(A,B,C,D)
- FD={AB->CD, C->A, D->B}
- $ABCD^+ = \{A,B,C,D\}$
- $AB^+ = \{A,B,C,D\}$
- Prime Attributes: A,B
- CK: AB
- CB and AD
- CK: AB, CB, AD
- Prime Attributes: A,B,C,D
- Note:all the attributes of the relation are prime attributes so the given relation is in 2NF

$A^+ = \{A\}$

$B^+ = \{B\}$

Proper subset is not a super key so AB is candidate key

Second Normal Form

- Example:R(A,B,C,D)
- FD={A->B,B->C,C->D}
- $ABCD^+ = \{A,B,C,D\}$
- $A^+ = \{A,B,C,D\}$
- CK: A
- Prime Attribute: A
- If the candidate key is having only one attribute then we can say the relation is in 2NF

Third Normal Form

Sid	Sname	DoB	State	Country	PIN Code	Tot. Credits
1	A	–	HR	IND	122001	–
2	B	–	HR	IND	122001	–
3	C	–	HR	IND	122001	–
4	D	–	punjab	IND	1223450	–

- If the candidate key is having only one attribute then we can say the relation is in second normal form
- Pincode -> State, Country
- Redundancy is there, update anomaly is there

Third Normal Form

Sid	Sname	DoB	State	Country	PIN Code	Tot. Credits
1	A	–	HR	IND	122001	–
2	B	–	HR	IND	122001	–
3	C	–	HR	IND	122001	–
4	D	–	punjab	IND	1223450	–

State	Country	PIN Code
HR	IND	122001
punjab	IND	1223450

untry

- Non prime attribute –> non prime attribute (transitive dependency)

Third Normal Form

- Rules:
- It is in 2NF
- No transitive dependency for nor prime attributes
- Example: R(A,B,C,D)
- FD: (A->B, B->C, C->D)
- $ABCD^{+} = \{A,B,C,D\}$
- $A^{+} = \{A,B,C,D\}$
- CK: A
- PA: A

Third Normal Form

- B->C and C->D these two are transitive dependencies so it is not 3NF
- Example: R(A,B,C,D,E,F)
- FD: {AB->CDEF, BD->F}
- $ABCDEF^+ = \{A,B,C,D,E,F\}$
- $AB^+ = \{A,B,C,D,E,F\}$
- $A^+ = \{A\}$
- $B^+ = \{B\}$
- **CK: AB and PA: A,B and NPA: C,D,E,F**
- BD->F, D & F are non prime attributes so it in not in 3NF

Third Normal Form

- R(A,B,C,D,E)
- FD: (A->B, B->C, C->D, D->A)

Boyce-Codd Normal Form

- R(A,B,C)
- FD: (A->B, B->C, C->A)
- Rules:
- It is in 3NF
- For each non trivial FD x->y x must be super key
- Ck: A,B,C

Boyce-Codd Normal Form

- R(A,B,C,D,E)
- FD: (A->BCDE, BC>ACE, D->E)

- CK: A, BC,AC,BA
- R(A,B,C,D,E)
- FD: {AB->CDE, D->A}
- CK: AB, DB,

Boyce-Codd Normal Form

- R(A,B,C,D,E,F,G,H)
- FD:{ABC->DE, E->GH, H->G, G->H, ABCD->EF}

Dependency Preserving Decomposition

- Decomposition should fallow the following properties
- 1. dependency preserving
- 2. lossless decomposition

- R(A,B,C)
- 1 1 1
- 2 1 2
- 3 2 1
- 4 2 2

FD: A->B, A->C
A->BC, BC->A