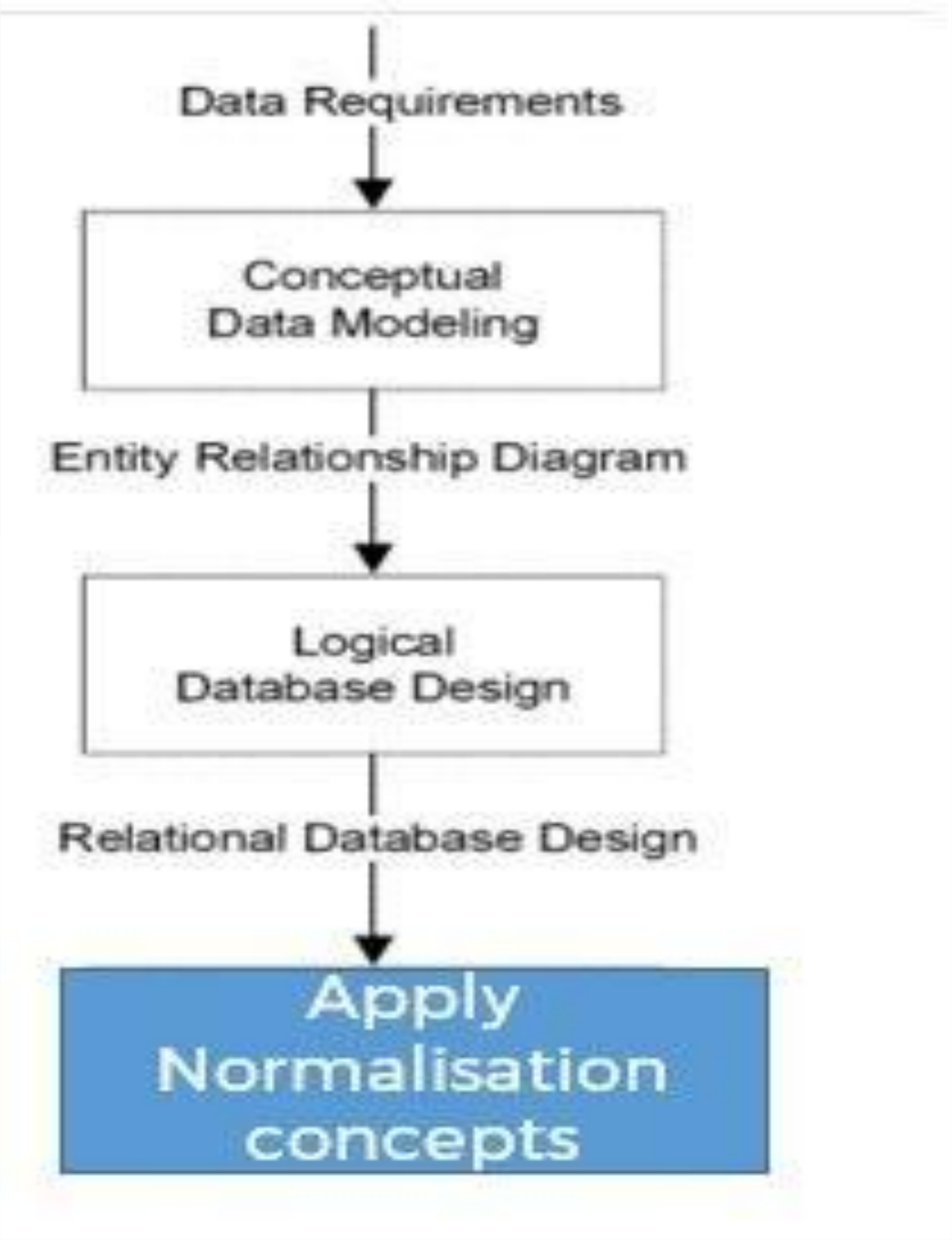
Functional Dependency

Presented by
Dr.Siddique Ibrahim
VIT-AP University

Topics to be covered on

- Functional Dependency
- Types of Functional Dependency
- Armstrong’s Axioms
- Closure of Functional Dependency
- Canonical cover of Functional Dependency


Data Requirements
Conceptual
Data Modeling
Entity Relationship Diagram
Logical
Database Design
Relational Database Design
Apply
Normalisation
concepts

Normalization Introduction

- Bottom up approach
- Divides the Larger table into smaller table and links them using relationship
- Reduce the data redundancy
- <u>It overcomes Anomalies</u>
  - Insert
  - Delete
  - Update

Roll No	Name	Dept	HoD	HoD phone
1	X	Cse	CH	123
2	Y	Cse	CH	123
3	Z	IT	IH	321
4	A	IT	IH	321

Functional dependency

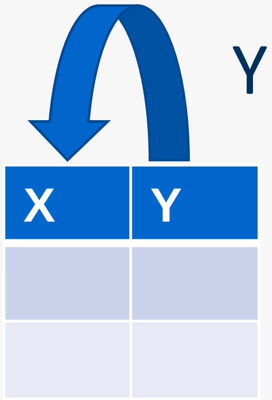
- The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table

- Functional Dependency (FD) determines the relation of one attribute to another attribute in a database management system (DBMS) system.
- Functional dependency helps you to maintain the quality of data in the database. A functional dependency is denoted by an arrow →.
- The functional dependency of X on Y is represented by
  - $X \to Y$.
  - Functional Dependency plays a vital role to find the difference between good and bad database design.

What is functional Dependency

Y is depends upon X

Relation R



Then, Y is functionally dependent on X

Determine

Determinant

Dependent

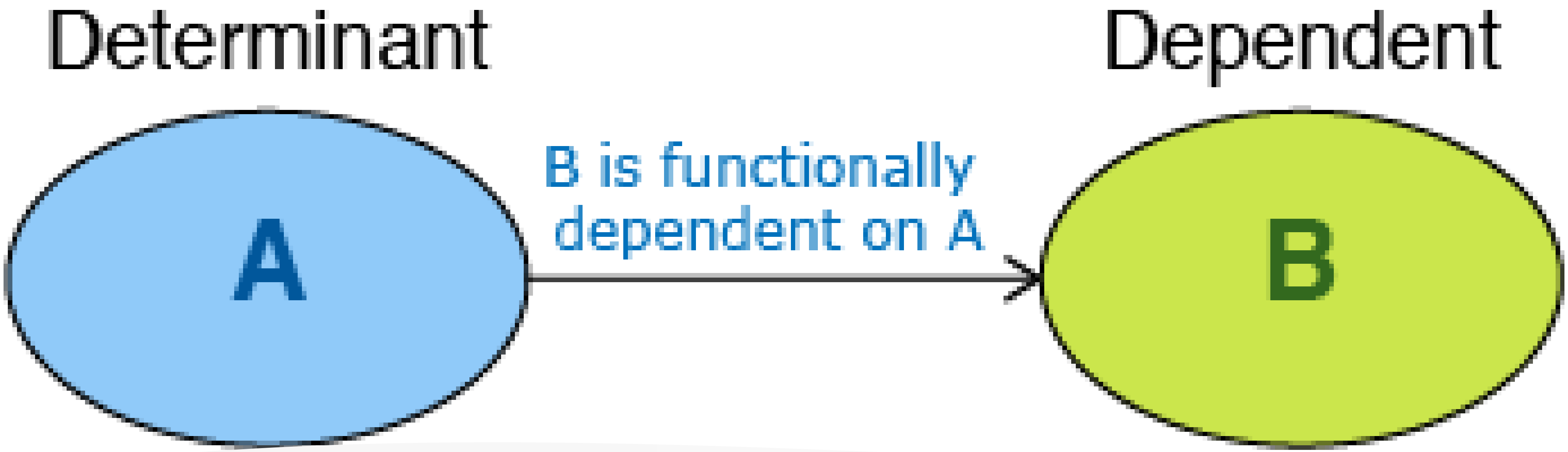
Primary Key

S.ID	Name	Surname
S1	Prabhu	N
S2	Ram	M
S3	Prabhu	N
S4	Siddharth	G

S. ID -> Name (Functionally dependent on ID)
S.ID-> Surname

FD. Table

Hence, Non key attributes that will depends upon key attributes



Functional Dependency

- Just like relationship between entities, attributes within an entity can be dependent on each other. These dependencies are expressed in terms of **functional dependency**.
- An attribute A is said to functionally determine attribute B if each value of A is associated with only one value of B.
- A is called the **Determinant** while B is called the **Dependent**.

Functional Dependency In DBMS : Examples

- **Example-1 : Consider a table student_details containing details of some students.**

Roll_No	Name	Marks
1.	Anoop	20
2.	Anurag	30
3.	Saurav	40
4.	Rakesh	30
5.	Pritesh	10
6.	Anoop	40

Example : student_details Table

- **FD1 : Roll_No $\rightarrow$ Name**
- **FD2 : Roll_No $\rightarrow$ Marks**

Example2

Employee_number	Employee Name	Salary	City
1	ANU	10000	BANGALORE
2	AJAY	75000	MYSORE
3	RAHUL	95000	MANGALORE

In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

Key terms

Roll No	Name	Dept_Name	Dept_building
42	Ajith	CSE	AB1
43	Pranesh	IT	AB1
44	Arun	CSE	AB1
45	Arun	MECH	AB2
46	Mano	ECE	AB2
47	Singh	MECH	AB2
48	Rahul	IT	AB1
49	Mano	IT	AB1

Valid FD

- roll_no → { name, dept_name, dept_building },→ Here, roll_no can determine values of fields name, dept_name and dept_building, hence a valid Functional dependency
- roll_no → dept_name , Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.
- dept_name → dept_building , Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building
- More valid functional dependencies: roll_no → name, {roll_no, name} ⇢ {dept_name, dept_building}, etc.

Invalid FD

- name → dept_name   Students with the same name can have different dept_name, hence this is not a valid functional dependency.
- dept_building → dept_name   There can be multiple departments in the same building, For example, in the above table departments MECH and ECE are in the same building AB2, hence dept_building → dept_name is an invalid functional dependency.
- More invalid functional dependencies: name → roll_no, {name, dept_name} → roll_no, dept_building → roll_no, etc.

Functional Dependency Type	Description
Full Functional Dependency	If A and B are attributes of a relation, B is fully functionally dependent on A if it is functionally dependent on A, but not on any subset of A.
Partial Functional Dependency	If A and B are attributes of a relation, B is partially dependent on A if it is dependent on subset of A.
Transitive Functional Dependency	If A, B, and C are attributes of a relation such that if A -> B and B -> C, then C is transitively dependent on A via B.

Types of Functional Dependencies

There are three types of functional dependencies

Fully Functional Dependency

- Example:
- ABC→D
- {D is fully functional dependency on ABC}

- D cannot depends on any subset of ABC
- BC→D ✖
- C→D ✖
- A→D ✖

The combination of P.ID, Name, and Order ID will determine the price of the product then the below table is called fully FD

P.ID	Name	Order date	Price
P1	Headphone	01/01/2024	400
P2	Speaker	15/02/2024	500
P1	Speaker	17/03/2023	550
P4	Headphone	01/01/2025	400
P5	Headphone	02/02/2022	450

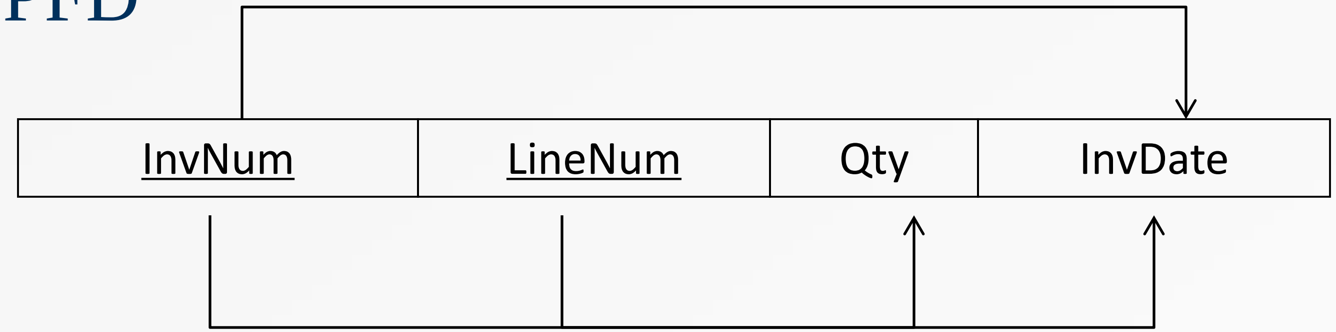
Example:2

- {Emp_num,Proj_num} -> Hour
- Is a full functional dependency. Here, Hour is the working time by an employee in a project

Partial dependency

A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.

A Attribute can be uniquely identified by subset of an attribute is called PFD



Candidate keys: {InvNum, LineNum}

InvDate is *partially dependent* on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

Example:2

- If {Emp_num,Proj_num} $\rightarrow$ Emp_name
- but also determine Emp_num $\rightarrow$ Emp_name then Emp_name is partially functionally dependent on {Empl_num,Proj_num}.

Transitive dependency

Transitive dependency

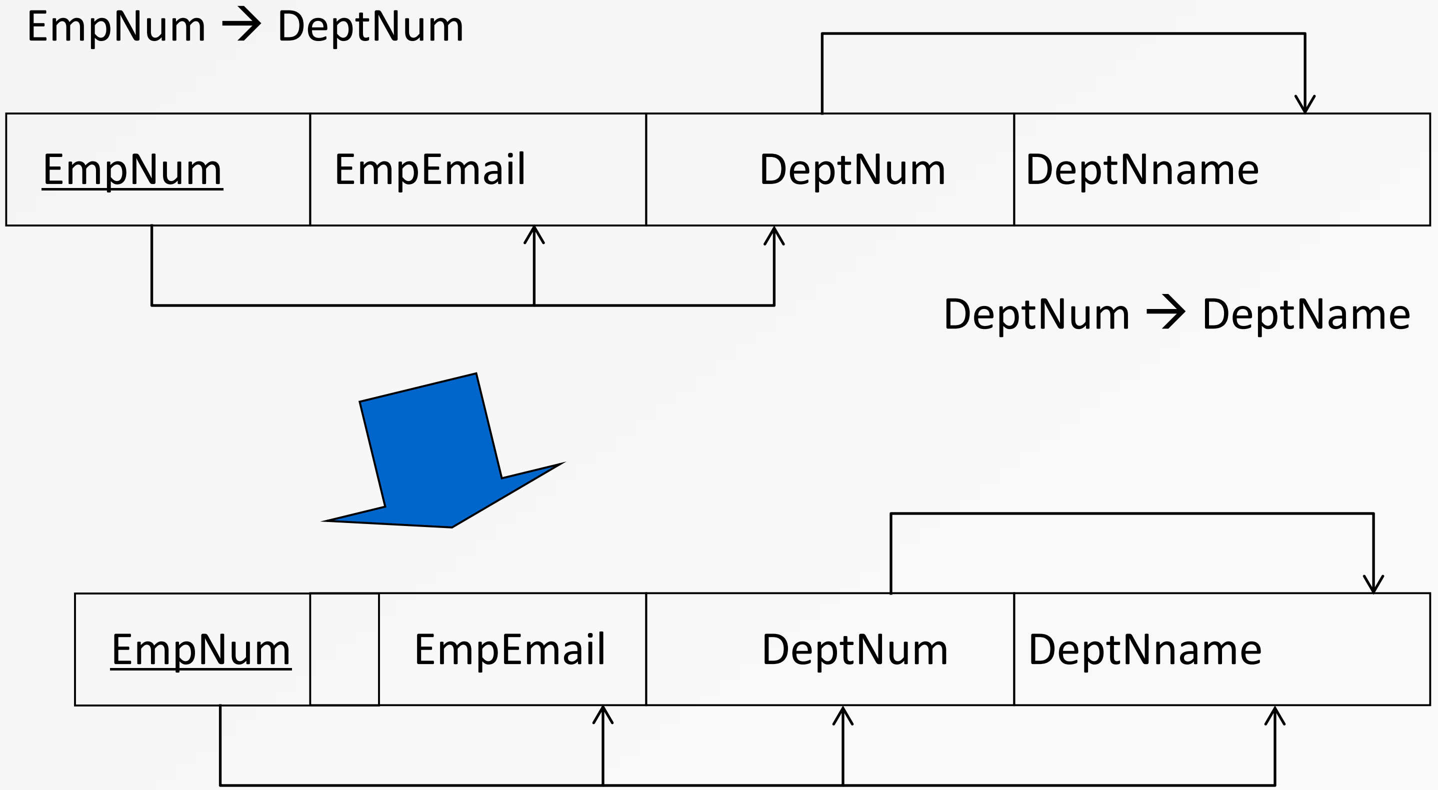
Consider attributes A, B, and C, and where

A $\rightarrow$ B and B $\rightarrow$ C.

Functional dependencies are transitive, which means that we also have the functional dependency A $\rightarrow$ C

We say that C is transitively dependent on A through B.

Transitive dependency



DeptName is *transitively dependent* on EmpNum via DeptNum

EmpNum → DeptName

Example:2 Codetantra Evaluation

- If {student reg no} -> mark
- {Dummy no}-> Mark
- {Reg no}-> {Dummy no}
- Dummy no -> FAT mark

Example:

Consider a relation which is used to store marks scored by students in various courses.
Student(StudentId, Name, ContactNo, Course, Marks, Grade)

STUDENTID	NAME	CONTACT NO	COURSE	MARKS	GRADE
1	James Potter	111-111-1111	OOP	80	B+
1	James Potter	111-111-1111	DBMS	95	A+
2	Ethan McCarty	222-222-3222	OOP	75	B
3	Emily Rayner	333-333-3333	PF	75	B

StudentId	→	Name	Full functional dependency
StudentId, Name	→	ContactNo	Partial functional dependency (StudentId)
StudentId		Course	No functional dependency
StudentId, Course	→	Marks	Full functional dependency
Marks	→	Grade	Full functional dependency
StudentId, Course	→	Grade	Transitive functional dependency
StudentId	→	ContactNo	Full functional dependency

Example

- Which functional dependency types is/are present in the following dependencies?

1. Empno -> EName, Salary, Deptno, DName

2DeptNo -> Dname

DName-> E.Name

3. Ename-> Salary

- Full functional dependency
- Transitive functional dependency
- Partial functional dependency

Functional Dependency In DBMS : Armstrong’s Axioms

Axioms in database management systems was introduced by William W. Armstrong in late 90’s and these axioms play a vital role while implementing the concept of functional dependency in DBMS for database normalization. There exists six inferences known a s “Armstrong’s Axioms” which are discussed below.

1. **Reflexive : It means, if set “B” is a subset of “A”, then A → B.**
2. **Augmentation : It means, if A → B, then AC → BC.**
3. **Transitive : It means, if A → B & B → C, then A → C.**
4. **Decomposition : It means, if A → BC, then A → B & A → C.**
5. **Union : It means, if A → B & A → C, then A → BC.**
6. **Pseudo-Transitivity : It means, if A → B and DB → C, then DA→ C.**

Closure Of Functional Dependency

- The Closure Of Functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency using the inference rules known as **Armstrong's Rules.**
- **If "F" is a functional dependency then closure of functional dependency can be denoted using "$\{F\}^+$".**

There are three steps to calculate closure of functional dependency

- **Step-1 :** Add the attributes which are present on Left Hand Side in the original functional dependency.
- **Step-2 :** Now, add the attributes present on the Right Hand Side of the functional dependency.
- **Step-3 :** With the help of attributes present on Right Hand Side, check the other attributes that can be derived from the other given functional dependencies. Repeat this process until all the possible attributes which can be derived are added in the closure.

Closure Of Functional Dependency : Examples

- **Example-1 : Consider the table student_details having (Roll_No, Name,Marks, Location) as the attributes and having two functional dependencies.**
- **FD1 : Roll_No → Name, Marks**
- **FD2 : Name → Marks, Location**

Find the clousure of the given functional dependency:!!

- **Step-1 : Add attributes present on the LHS of the first functional dependency to the closure.**
- **$\{Roll\_no\}^+ = \{Roll\_No\}$**
- **Step-2 : Add attributes present on the RHS of the original functional dependency to the closure.**
- **$\{Roll\_no\}^+ = \{Roll\_No, Name, Marks\}$**

Example-1 :

- FD1 : Roll_No → Name, Marks
- FD2 : Name → Marks, Location
- Step-3 : Add the other possible attributes which can be derived using attributes present on the RHS of the closure.

  Therefore, complete closure of Roll_No will be :

- $\{Roll\_no\}^+ = \{Roll\_No, Marks, Name, Location\}$

- Similarly, we can calculate closure for other attributes too i.e “Name”.
- $\{Name\}^+ = \{Name\}$
- $\{Name\}^+ = \{Name, Marks, Location\}$
- $\{Marks\}^+ = \{Marks\}$

  and

- $\{Location\}^+ = \{ Location\}$

Example-2 :

- Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies.
- FD1 : $A \rightarrow BC$
- FD2 : $C \rightarrow B$
- FD3 : $D \rightarrow E$
- FD4 : $E \rightarrow D$
- Now, we need to calculate the closure of attributes of the relation R. The closures will be:
- $\{A\}^+ = \{A, B, C\}$
- $\{B\}^+ = \{B\}$
- $\{C\}^+ = \{B, C\}$
- $\{D\}^+ = \{D, E\}$
- $\{E\}^+ = \{E,D\}$

- Consider a relation **R(A,B,C,D,E,F)**

- F:
- E->A, E->D, A->C, A->D, AE->F, AG->K.

- Find the closure of E or E+

The closure of E or E+ is as follows −

- E+ = E
- =EA {for E->A add A}
- =EAD {for E->D add D}
- =EADC {for A->C add C}
- =EADC {for A->D D already added}
- =EADCF {for AE->F add F}
- =EADCF {for AG->K don’t add k AG $\not\subset$ E+)

- We are given the relation R(A, B, C, D, E). This means that the table R has five columns: A, B, C, D, and E. We are also given the set of functional dependencies: {A->B, B->C, C->D, D->E}.

- What is {A}+?

- Let’s look at another example. 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}+?

- Let the relation R(A,B,C,D,E,F)

- F: B->C, BC->AD, D->E, CF->B. Find the closure of B.

- Consider a relation R (A , B , C , D , E , F , G) with the functional dependencies-
- Find the closure of A^+ , BC^+
- $A \to BC$
- $BC \to DE$
- $D \to F$
- $CF \to G$

- Closure of attribute A-
- A+ = { A }
- = { A , B , C } (Using $A \rightarrow BC$)
- = { A , B , C , D , E } (Using $BC \rightarrow DE$)
- = { A , B , C , D , E , F } (Using $D \rightarrow F$)
- = { A , B , C , D , E , F , G } (Using $CF \rightarrow G$)
- Thus,
- A+ = { A , B , C , D , E , F , G }

- The following functional dependencies are given:
- $\{AB \rightarrow CD, AF \rightarrow D, DE \rightarrow F, C \rightarrow G, F \rightarrow E, G \rightarrow A\}$
- Which one of the following options is/are false? (GATE 2006)

- $CF^+ = \{ACDEFG\}$
- $BG^+ = \{ABCDG\}$
- $AF^+ = \{ACDEFG\}$
- $AB^+ = \{ABCDFG\}$

GATE Question:(GATE-CS-2014)

- Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies
- {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. Find the closure for

- E
- F
- EF
- EFH

- GATE Question: Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. What is the key for R? (GATE-CS-2014)
- A. {E, F}
- B. {E, F, H}
- C. {E, F, H, K, L}
- D. {E}

Answer:

- Finding attribute closure of all given options, we get:
- {E,F}+ = {EFGIJ}
- {E,F,H}+ = {EFHGIJKLMN}
- {E,F,H,K,L}+ = {{EFHGIJKLMN}
- {E}+ = {E}
- {EFH}+ and {EFHKL}+ results in set of all attributes, but EFH is minimal. So it will be candidate key. So correct option is (B)

- Given the relation R with attributes A,B,C,D,E and the following functional dependencies:

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

Question: What is the closure of B+?

Given the relation R with attributes A, B, C, D, E and the following functional dependencies:

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

Question: Can A determine D? In other words, is $A \rightarrow D$ true?

- Since D is included in A+, we conclude that A→D is true.

- Given the relation R with attributes A,B,C,D,E and the following functional dependencies:
- A→B
- B→C
- C→D
- A→E
- Question: What is the closure of the set {A,B}, i.e., {A,B} + ?

Given the relation R with attributes A, B, C, D, E and the following functional dependencies:

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

Question: Does $\{A, C\}^+$ contain E? In other words, is $\{A, C\}^+$ such that $E \in \{A, C\}^+$?

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

Closure Of Functional Dependency : Calculating Candidate Key

- **“A Candidate Key of a relation is an attribute or set of attributes that can determine the whole relation or contains all the attributes in its closure.“**

- **Let’s try to understand how to calculate candidate keys.**

Example-1 :

- Consider the relation R(A,B,C) with given functional dependencies :
- FD1 : $A \rightarrow B$
- FD2 : $B \rightarrow C$
- Now, calculating the closure of the attributes as :
- $\{A\}^+ = \{A, B, C\}$
- $\{B\}^+ = \{B, C\}$
- $\{C\}^+ = \{C\}$
- Clearly, "A" is the candidate key as, its closure contains all the attributes present in the relation "R".

Example-2

- Consider another relation R(A, B, C, D, E) having the Functional dependencies :
- FD1 : $A \rightarrow BC$
- FD2 : $C \rightarrow B$
- FD3 : $D \rightarrow E$
- FD4 : $E \rightarrow D$
- Now, calculating the closure of the attributes as :
- $\{A\}^+ = \{A, B, C\}$
- $\{B\}^+ = \{B\}$
- $\{C\}^+ = \{C, B\}$
- $\{D\}^+ = \{E, D\}$
- $\{E\}^+ = \{E, D\}$
- $\{A, D\}^+ = \{A, B, C, D, E\}$
- $\{A, E\}^+ = \{A, B, C, D, E\}$
- Hence, "AD" and "AE" are the two possible keys of the given relation “R”. Any other combination other than these two would have acted as extraneous attributes.

Closure Of Functional Dependency : Key Definitions

1. **Prime Attributes : Attributes which are indispensable part of candidate keys. For example : "A, D, E" attributes are prime attributes in above example-2.**

2. **Non-Prime Attributes : Attributes other than prime attributes which does not take part in formation of candidate keys.**

3. **Extraneous Attributes : Attributes which does not make any effect on removal from candidate key.**

Closure Of Functional Dependency : Key Definitions

1. **Extraneous Attributes : Attributes which does not make any effect on removal from candidate key.**

- Example 1:

- **Let us consider a relation R with schema R(A, B, C) and set of functional dependencies F = { AB → C, A → C }. The closure for F is F+ = { AB → C, A → C }.**

- **In AB → C, B is extraneous attribute. The reason is, there is another FD A → C, which means when A alone can determine C, the use of B is unnecessary (redundant).**

- **Now, we can find the closure for the new set of functional dependencies, which is same as F+. Hence, we can declare that B is extraneous.**

- **AB → CD**
- In the set of functional dependencies F = {A → C, AB → CD}, C is extraneous in AB → CD because AB → C can be inferred even after deleting C

- **A → BC**
- In the set of functional dependencies F = {A → BC, B → C}, C is extraneous in A → BC because A → C can be inferred even after deleting C.

Example

- **Consider another relation R(A, B, C, D) having the Functional dependencies :**

- **FD1 : A → BC**
- **FD2 : B → C**
- **FD3 : D → AD**
- **Here, Candidate key can be "AD" only. Hence,**
- **Prime Attributes : A, D.**
- **Non-Prime Attributes : B, C**
- **Extraneous Attributes : B, C(As if we add any to the candidate key, it will remain unaffected). Those attributes, which if removed does not affect closure of that set.**

Canonical Cover Of Functional Dependency

- In any relational model, there exists a set of functional dependencies. These functional dependencies when closely observed might contain redundant attributes. The ability of removing these redundant attributes without affecting the capabilities of the functional dependency is known as **"canonical cover of functional dependency".**

- Canonical cover of functional dependency is sometimes also referred to as **"minimal cover".**

- Canonical cover of functional dependency is denoted using "M_c".

Canonical Cover Of Functional Dependency : Example

- Consider a relation R(A,B,C,D) having some attributes and below are mentioned functional dependencies.
- FD1 : B → A
- FD2 : AD → C
- **FD3 : C → ABD**
-

1. **Reflexive** : It means, if set "**B**" is a subset of "**A**", then **A** → **B**.
2. **Augmentation** : It means, if **A** → **B**, then **AC** → **BC**.
3. **Transitive** : It means, if **A** → **B** & **B** → **C**, then **A** → **C**.
4. **Decomposition** : It means, if **A** → **BC**, then **A** → **B** & **A** → **C**.
5. **Union** : It means, if **A** → **B** & **A** → **C**, then **A** → **BC**.
6. **Pseudo-Transitivity** : It means, if **A** → **B** and **DB** → **C**, then **DA** → **C**.

Step-1 : Decompose the functional dependencies using Decomposition rule(Armstrong's Axiom) i.e. single attribute on right hand side.

FD1 : B → A
FD2 : AD → C
FD3 : C → A
FD4 : C → B
FD5 : C → D

1.**Transitive** : It means, if **A** → **B** & **B** → **C**, then **A** → **C**.
2.**Decomposition** : It means, if **A** → **BC**, then **A** → **B** & **A** → **C**.

Example Contd.,

Step-2 : Remove extraneous attributes from LHS of functional dependencies by calculating the closure of FD's having two or more attributes on LHS.

Here, only one FD has two or more attributes of LHS i.e. AD → C.

$\{A\}^+ = \{A\}$

$\{D\}^+ = \{D\}$

Step-3 : Remove FD's having transitivity.

FD1 : B → A

FD2 : C → A

FD3 : C → B

FD4 : AD → C

FD5 : C → D

Above FD1, FD2 and FD3 are forming transitive pair. Hence, using Armstrong's law of transitivity i.e. if X → Y, Y → **Z** then X → Z should be removed. Therefore we will have the following FD's left :

Example Contd.,

FD1 : B → A
FD2 : C → B
FD3 : AD → C
FD4 : C → D

FD1 : B → A
FD2 : C → A
FD3 : C → B
FD4 : AD → C
FD5 : C → D

Also, FD2 & FD4 can be clubbed together now. Hence, the canonical cover of the relation R(A,B,C,D) will be:

M_c {R(ABCD)} = {B → A , C →BD, AD → C}