Module 3

KEYS, FUNCTIONAL DEPENDENCY AND DEPENDENCY DIAGRAM

Dr. L.M. Jenila Livingston

VIT Chennai

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. $X \to Y$.

How to locate the given value ‘X’

A	B	C
1	a	X
2	b	Y
3	b	X
4	c	y

Select A where C=‘X’;
Column name is not sufficient to identify the value ‘X’ as the row/tuple name is not given

Functional Dependency

A	B	C
1	a	X
2	b	Y
3	b	X
4	c	y

A → (B,C)

Select A where B=‘b’ and C=‘X’;

(B,C) → A

SuperKey

- **SuperKey**: A **key** that can be uniquely used to identify a database record, that **may contain extra attributes** that are not necessary to uniquely identify records.

In this table

{A} → {B,C}

{B,C} → {A}

{A,B} → {C}

{A,C} → {B}

Is used to identify the records.

A	B	C
1	a	X
2	b	Y
3	b	X
4	c	y

Candidate Key

- A **candidate key** is a set of attributes (or attribute) which uniquely identify the tuples in relation or table.
- **Minimal super key** form a candidate key (Subset of key is not a Super key)
- **R{A,B,C,D}**

A→ B, C, D (Super key since all attributes involved)
A,B→C, D (Super key)
A,B,C→D (Super key)
B,D → A, C (Super key)
C→ A, D (Not a Super key)

	Super Key	Candidate Key
A→ B, C, D	✓	✓
A,B→C, D	✓	x
A,B,C→D	✓	x
B,D -> A, C	✓	✓
C→ A, D	x	x

- Student{ID, First_name, Last_name, DOB}
- Here we can see the two candidate keys {**ID**} and **{First_name, Last_name, DOB}**. So here, there are present more than one candidate keys, which can uniquely identify a tuple in a relation.

Primary Key

- Primary Key is a set of attributes (or attribute) which uniquely identify the tuples in relation or table.
- The primary key is a minimal super key (candidate key), so **there is only one primary key in any relation**.
- Primary key will not accept duplicate values
- **Student{<u>ID</u>, First_name, Last_name, DOB}**
- ID is a primary key

Primary Key

	Super Key	Candidate Key	Primary Key
A→ B, C, D	✓	✓	✓
A,B→C, D	✓	x	x
A,B,C→D	✓	x	x
B,D →A,B	✓	✓	x
C→ A, D	x	x	x

Unique Key

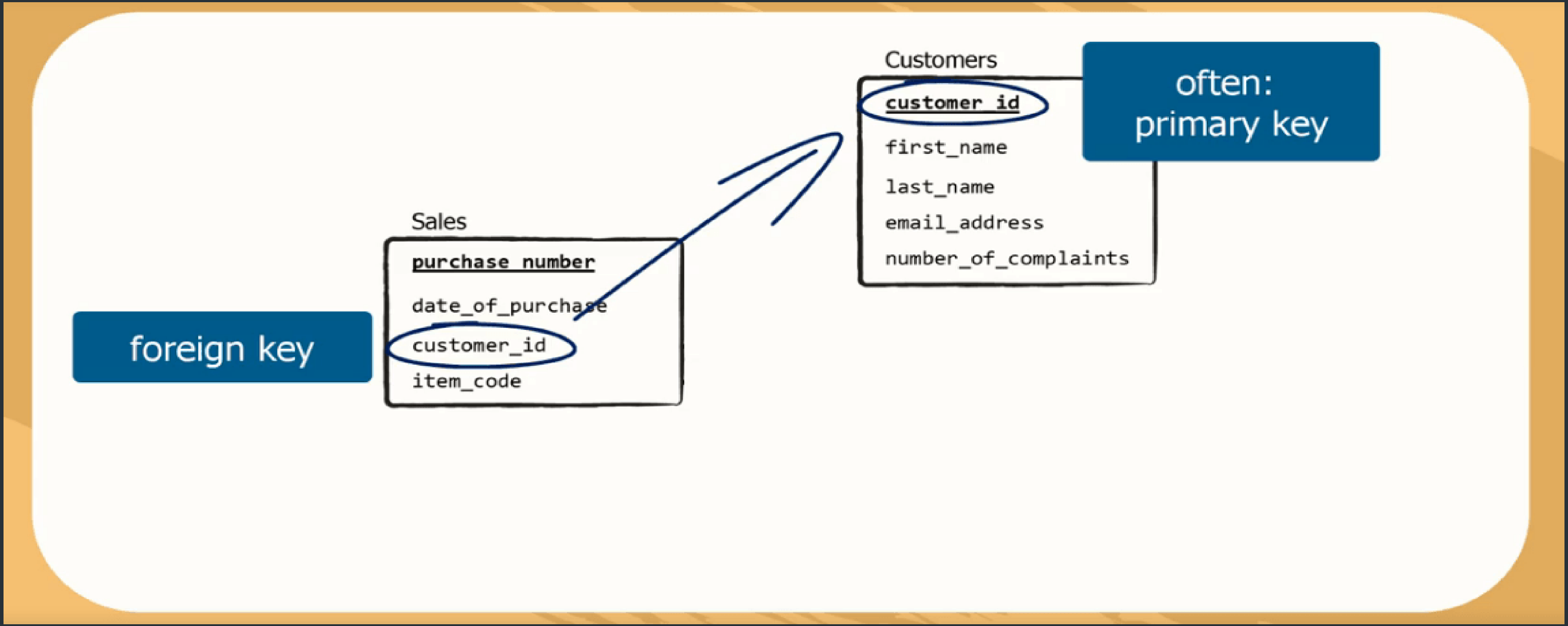
- Same concept as Primary Key
- But it will accept null values

You can say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.

Foreign Key

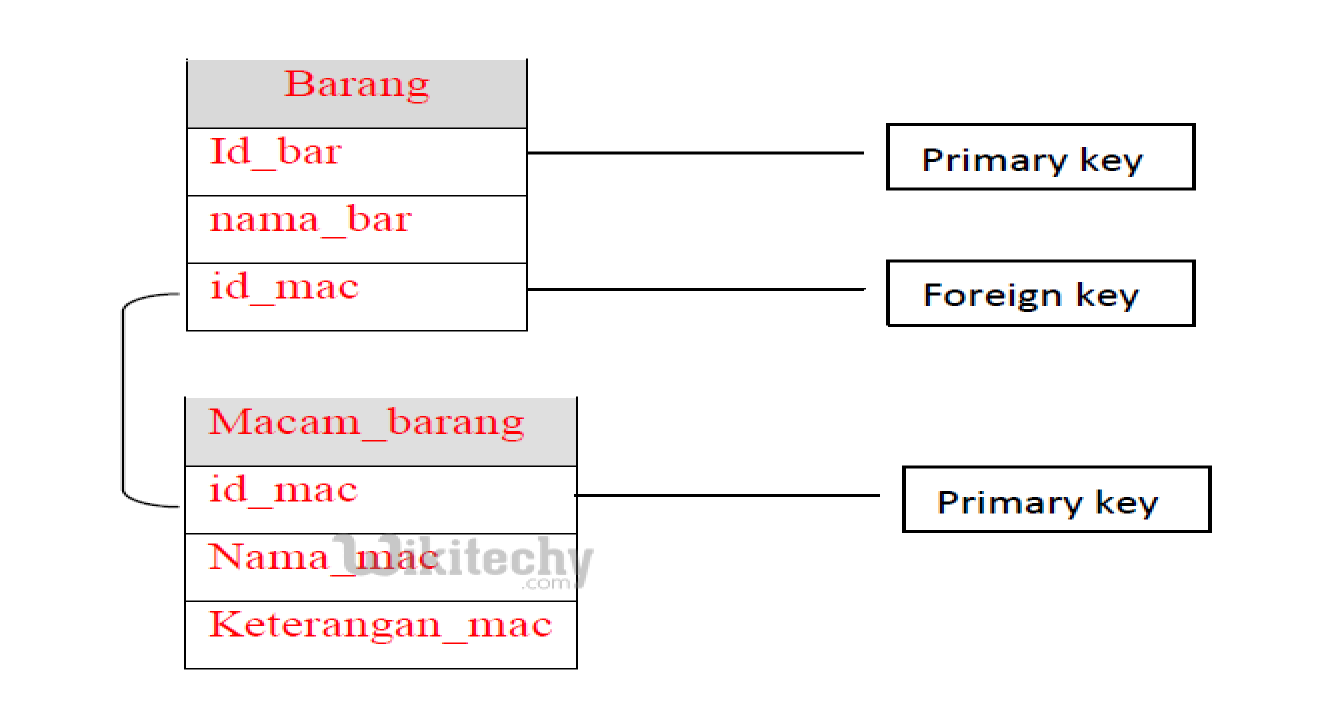
- Otherwise called as reference key
- A **foreign key** is a column or group of columns in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary **key** of another table, thereby establishing a link between them.

Foreign Key



Customers
customer_id
first_name
last_name
email_address
number_of_complaints
often: primary key
Sales
purchase_number
date_of_purchase
customer_id
item_code
foreign key

Foreign Key


Barang
Id_bar
nama_bar
id_mac
Primary key
Foreign key
Macam_barang
id_mac
Nama_mac
Keterangan_mac
Primary key
Wikitechy.com

Functional Dependency (FD)

- The attributes of a table is said to be dependent on each other when an attribute/ attributes of a table uniquely identifies another attribute of the same table.

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	C2	d2	e1
a3	b2	C1	d1	e1
a4	b2	C2	d2	e1
a5	b3	C3	d1	e1

Table R

$A \to B$, $A \to C$, $A \to D$, $A \to E$

Summarized as

$A \to BCDE$

From our understanding of primary keys, A is a primary key.

Functional Dependency

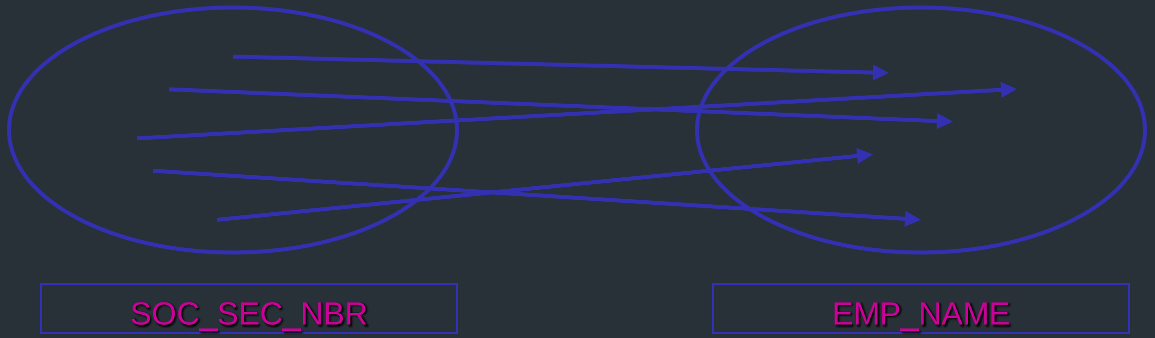
- Relationship between columns X and Y such that, given the value of X, one can *determine* the value of Y. Written as X $\rightarrow$ Y

i.e., for a given value of X we can obtain (or look up) a specific value of X

- X is called the *determinant* of Y
- Y is said to be *functionally dependent* on X

Functional Dependency

- Example
  - SOC_SEC_NBR → EMP_NAME



-One and only one EMP_NAME for a specific SOC_SEC_NBR

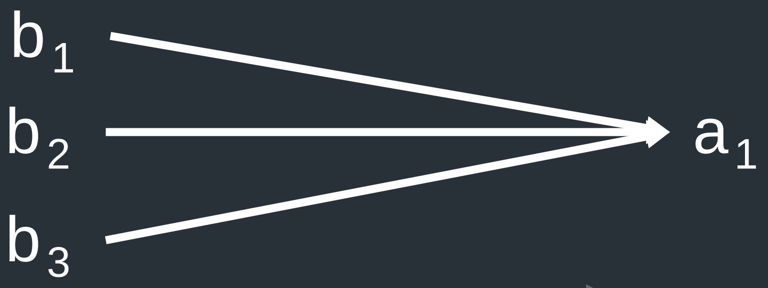
- SOC_SEC_NBR is the *determinant* of EMP_NAME

- EMP_NAME is functionally *dependent* on SOC_SEC_NBR

Functional Dependence

An attribute A is <u>functionally dependent</u> on attribute(s) B if:

*given a value b for B there is **one and only one corresponding value** a for A (at a time).*



$$B \rightarrow A$$

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

- STUD_NO → STUD_NAME, **FD hold** because for each STUD_NAME, there is a unique value of STUD_NO.
- STUD_NO → STUD_PHONE, **FD hold**
- STUD_NAME → STUD_NO, **FD does not hold,** because STUD-NAME 'Ram' is not uniquely determining STUD-ID. There are STUD-NO corresponding to Ram (1 and 2).
- STUD_NAME → STUD_STATE, **FD does not hold**

{ STUD_NO → STUD_NAME,
STUD_NO → STUD_PHONE,
STUD_NO → STUD_STATE,
STUD_NO → STUD_COUNTRY,
STUD_NO → STUD_AGE,
STUD_STATE → STUD_COUNTRY }

Properties of FD

Let *X*, *Y*, and *Z* are sets of attributes in a relation *R*. There are several properties of functional dependencies which always hold in R also known as Armstrong Axioms.

- **Reflexivity:** If *Y* is a subset of *X*, then *X* → *Y*. If $Y \subseteq X$, then $X \to Y$
- e.g.; Let X represents {E-ID, E-NAME} and Y represents {E-ID}.
  {E-ID, E-NAME} → E-ID is true for the relation.
- **Augmentation:** If *X* → *Y*, then *XZ* → *YZ*.

e.g.; Let X represents {E-ID}, Y represents {E-NAME} and Z represents {E-CITY}.
As {E-ID} →{E-NAME} is true for the relation,
so { E-ID,E-CITY} →{E-NAME,E-CITY} will also be true.

- **Transitivity:** If *X* → *Y* and *Y* → *Z*, then *X* → *Z*.
- e.g.; Let X represents {E-ID}, Y represents {E-CITY} and Z represents {E-STATE}.
  As {E-ID} →{E-CITY} and {E-CITY} →{E-STATE} is true for the relation,
  so { E-ID } →{E-STATE} will also be true.

Properties of FD

- **Union:** It states that if X determines Y and X determines Z then X must also determine Y and Z

$$\text{If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

- **Decomposition:** This rule states that if X determines Y and Z, then X determines Y and X determines Z separately

$$\text{If } X \rightarrow YZ \text{ then } X \rightarrow Y \text{ and } X \rightarrow Z$$

Trivial Dependency

A trivial functional dependency is the one which will always hold in a relation.

- **X → Y will always hold if X ⊇ Y** If $Y \subseteq X$, then $X \rightarrow Y$
- In the example given above, **E-ID, E-NAME → E-ID** is a trivial functional dependency
- If a functional dependency is not trivial, it is called **Non-Trivial Functional Dependency**. Non-Trivial functional dependency may or may not hold in a relation.
- e.g; **E-ID → E-NAME** is a non-trivial functional dependency which holds in the above relation.

Trivial Dependency

- Which of the following is the trivial functional dependency?
- (a) {P, R} → {S, T}
  (b) {P, R} → {R, T}
  (c) {P, S} → {S}
  (d) {P, S, U} → {Q}
- Gate 2015

Trivial Dependency

- Which of the following is the trivial functional dependency?
- (a) {P, R} → {S, T}
  (b) {P, R} → {R, T}
  (c) {P, S} → {S}
  (d) {P, S, U} → {Q}
- Ans: option (c)
  Explanation:
  A functional dependency X → Y is trivial, if Y is a subset of X.
  In the above question , {S}is a subset of {P,S}. Hence option (c) is the answer.

Transitive dependency

- Let A, B, and C designate three distinct (but not necessarily disjoint) sets of attributes of a relation. Suppose all three of the following conditions hold:
- If X → Y and Y → Z is true, then X → Y is a transitive dependency.
- X → Y
- Y does not → X
- Y → Z

{Book} → {Author}

{Author} does not → {Book}

{Author} → {Author Nationality}

Exercise

- Given the following relation instance.

```
-------
X  Y  Z
-------
1  4  2
1  5  3
1  6  3
3  2  2
-------
```

Which of the following functional dependencies are satisfied by the instance?

(a) $XY \rightarrow Z$ and $Z \rightarrow Y$
(b) $YZ \rightarrow X$ and $Y \rightarrow Z$
(c) $YZ \rightarrow X$ and $X \rightarrow Z$
(d) $XZ \rightarrow Y$ and $Y \rightarrow X$

- (Gate 2000)

Exercise

- Given the following relation instance.

```
-------
X  Y  Z
-------
1  4  2
1  5  3
1  6  3
3  2  2
-------
```

Which of the following functional dependencies are satisfied by the instance?

(a) XY → Z and Z → Y
(b) YZ → X and Y → Z
(c) YZ → X and X → Z
(d) XZ → Y and Y → X

- (Gate 2000)

Attribute Closure (F)+

- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.

How to find attribute closure of an attribute set?

To find attribute closure of an attribute set:

- Add elements of attribute set to the result set. (Add A to S)
- Recursively add elements to the result set which can be functionally determined from the elements of the result set.

Attribute Closure

{ STUD_NO → STUD_NAME,
STUD_NO → STUD_PHONE,
STUD_NO → STUD_STATE,
STUD_NO → STUD_COUNTRY,
STUD_NO → STUD_AGE,
STUD_STATE → STUD_COUNTRY }

- **attribute closure** can be determined as:
- (STUD_NO)+ = {STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY, STUD_AGE}
- (STUD_STATE)+ = {STUD_STATE, STUD_COUNTRY}

Exercise

- 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 false?

(a) $CF^+ = \{ACDEFG\}$

(b) $BG^+ = \{ABCDG\}$

(c) $AF^+ = \{ACDEFG\}$

(d) $AB^+ = \{ABCDFG\}$

(Gate 2006, 2014)

Exercise - Solution

- Ans: option(c) and Option (d)
- Explanation:
- $AF^+ = \{AFDE\}$
- $AB^+ = \{ABCDG\}$.

How to check whether an FD can be derived from a given FD set?

- To check whether an FD A → B can be derived from an FD set F,
- Find (A)+ using FD set F.
- If B is subset of (A)+,
  - then A → B is true
  - else not true.

FD from FD Set

- **In a schema with attributes A, B, C, D and E following set of functional dependencies are given**
  {A → B, A → C, CD → E, B → D, E → A}
  Which of the following functional dependencies is NOT implied by the above set? (GATE IT 2005)
  A. CD → AC
  B. BD → CD
  C. BC → CD
  D. AC → BC

FD from FD Set

- Using FD set given in question,
  (CD)+ = {CDEAB} which means CD → AC also holds true.
  (BD)+ = {BD} which means BD → CD can't hold true. So this FD is no implied in FD set.
- Others can be checked in the same way.

- So (B) is the required option.

How to find Candidate Keys and Super Keys using Attribute Closure?

- If **attribute closure** of an attribute set contains **all attributes** of relation, the attribute set will be **super key** of the relation.

- If **no subset** of this attribute set can **functionally determine all attributes** of the relation, the set will be **candidate key** as well

Finding a key – Exercise1

- **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}

Finding a key

- **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).

Finding a key – Exercise 2

- Compute the closure of the following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

  $A \rightarrow BC$

  $CD \rightarrow E$

  $B \rightarrow D$

  $E \rightarrow A$

- List the candidate keys for R.

- Compute the closure of the following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

  A → BC
  CD → E
  B → D
  E → A
- List the candidate keys for R.

Answer:

- A → BC, B → D so A → D so A → DC → E therefore A → ABCDE
- E → A, A → ABCDE, so E → ABCDE
- **Attribute closure:**

A+={ABCDE}
B+={BD}
C+={C}
D+={D}
E+={ABCDE}

AB+={ABCDE}
AC+={ABCDE}
AD+={ABCDE}
AE+={ABCDE}
BC+={ABCDE}
BD+={BD}
BE+={ABCDE}
CD+={ABCDE}
CE+={ABCDE}
DE+={ABCDE}

ABC+={ABCDE}
ABD+={ABCDE}
ABE+={ABCDE}
ACD+={ABCDE}
ACE+={ABCDE}
ADE+={ABCDE}
BCD+={ABCDE}
BDE+={ABCDE}
CDE+={ABCDE}

ABCD+={ABCDE}
ABCE+={ABCDE}
ABDE+={ABCDE}
ACDE+={ABCDE}
BCDE+={ABCDE}

Finding a key – Exercise 3

- Consider a relation R(A,B,C,D,E) with the following dependencies:
- {AB → C, CD → E, DE → B}
- Is AB a candidate key of this relation? If not, is ABD? Explain your answer.

- A → A
- B → B
- C → C
- D → D
- E → E
- AB → ABC
- AC → AC
- AD → AD
- AE → AE
- BC → BC
- BD → BD
- BE → BE
- CD → BCDE
- CE → CE
- DE → BDE
- ABD → ABCDE
- No. The closure of AB does not give you all of the attributes of the relation.
  Yes, ABD is a candidate key. No subset of its attributes is a key.

Extraneous Attribute

- If we are able to **remove an attribute** from a functional dependency **without changing the closure** of the set of functional dependencies, that attribute is called as Extraneous Attribute.
- *Dictionary meaning of 'Extraneous' is 'irrelevant', 'inappropriate', or 'unconnected'*

Extraneous Attribute

- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.

a.) Attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$

b.) Attribute A is extraneous in β if $A \in \beta$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$

Extraneous Attribute

- Assume a set of functional dependencies F, and the **closure** of set of functional dependencies **F⁺**.
- Also, assume that we **remove an attribute (Extraneous Attribute) from** any of the FDs under **F** and find the closure of new set of functional dependencies.
- Let us mention the **new closure** of set of functional dependencies as **F1⁺**.
- If **F⁺ equals** the newly constituted closure (Minimal Cover) **F1⁺**, then the attribute which has been removed is called as Extraneous Attribute.
- In other words, that attribute does not violate any of the functional dependencies.

Extraneous Attribute

- Let us consider a relation R with schema R(A, B, C) and set of functional dependencies
- F = { AB → C, A → C }.
- The closure F^+ = {A → C, AB → 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).
- $F1^+$ = {A → C}.

Minimal cover

Definition

A set of FDs F is minimum if F has as few FDs as any equivalent set of FDs.

Simple properties/steps of minimal cover:

Steps to find the minimal cover;

1. Ensure **singleton attribute on the right hand side** of each functional dependency (**apply decomposition rule**).
2. Remove **extraneous (redundant) attribute from the left hand side** of each functional dependency.
3. Remove redundant functional dependency if any.

Canonical Cover

- Both concepts are same
- A **canonical cover** is "allowed" to have more than one attribute on the right hand side. A **minimal cover** cannot allow more than one attribute at RHS.
- As an example, the **canonical cover** is A → BC where the **minimal cover** would be A → B, A → C .

Exercise 1 – Minimal Cover

- **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**

- 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

- **Step-2 : Remove extraneous attributes from LHS of functional dependencies.**
- **Here, only one FD has two or more attributes of LHS i.e. AD $\rightarrow$ C.**
- **In this case, attribute "C" is determined by AD only.**
- **Hence, no extraneous attributes are present and the FD will remain the same and will not be removed.**

- Step-3 : Remove FD's having transitivity.
- FD1 : B → A
- FD2 : C → A
- FD3 : C → B
- FD4 : AD → C
- FD5 : C → D

Repetition, So check B->C or C->B in the FD set. C->B exists. Use transitivity rule and remove transitivity

- Above FD1, FD2 and FD3 are forming transitive pair. Hence, using Armstrong's law of transitivity

  i.e. if X → Y, Y → X then X → Z

  should be removed.

C →B, B →A then remove C →A

Therefore we will have the following FD's left :

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

Minimal Cover – Exercise 2

Let R(A, B, C) be a relation with the following set F of functional dependencies;
F = { A → B, B → A, A → C, C → A, B → C }
Find the minimal cover of F.

According to rule 1, if we have any FDs with more than one attribute on the Right Hand Side, that FD should be decomposed using decomposition rule. We don't have such FDs.

According to rule 2, if we have any FDs that have more than one attribute on the Left Hand Side (determiner), that FD must be checked for partial dependency. We don't have such FDs.
Hence, the given set satisfies both rules.

Minimal Cover

After Step 3 – Removing repetitions

F = { A → B, B → A, A → C, C → A, B → C }

- Apply transitivity rule:
- **A → C, B → C** (check A → B or B → A exists in the concrete list; **A → B** exists)

  A → B, B → C so remove A → C
- **So F = { A → B, B → A, C → A, B → C }**
- **B → A, C → A** (check B → C or C → A exists in the concrete list; **B → C** exists)

  B → C, C → A so remove B → A
- **So F_c = { A → B, C → A, B → C }**

Minimal Cover – Exercise 3

Find the minimal cover of the set of functional dependencies given;

{A → C, AB → C, C → DI, CD → I, EC → AB, EI → C}

Find the minimal cover of the set of functional dependencies given;

{A → C, AB → C, C → DI, CD → I, EC → AB, EI → C}

Solution

1). Right Hand Side (RHS) of all FDs should be single attribute. So we write F as F1, as follows;

**F1 = {A → C, AB → C, C → D, C → I, CD → I, EC → A, EC → B,
EI → C}**

2. Remove extraneous attributes.

Extraneous attribute is a redundant attribute on the LHS of the functional dependency. In the set of FDs, $AB \rightarrow C$, $CD \rightarrow I$, $EC \rightarrow A$, $EC \rightarrow B$, and $EI \rightarrow C$ have more than one attribute in the LHS. Hence, we check one of these LHS attributes are extraneous or not.

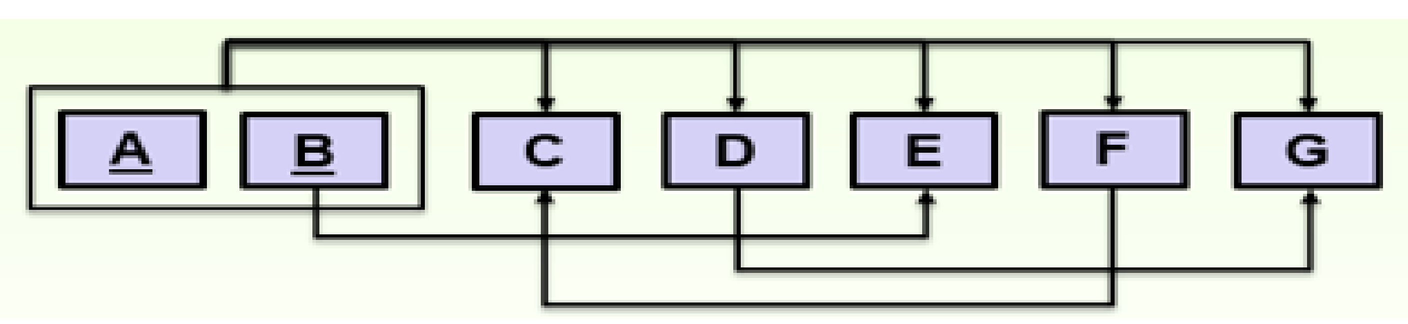
F2 = {A → C, C → D, C → I, EC → A, EC → B}

3. Eliminate redundant functional dependency.

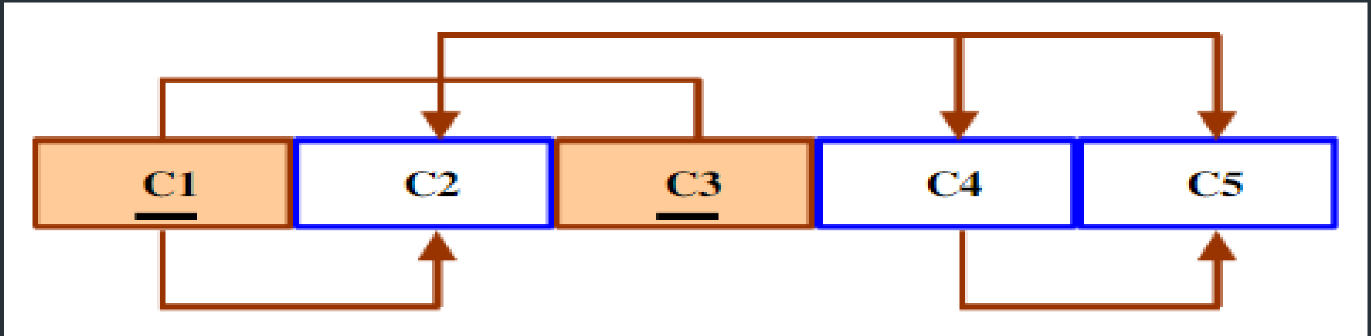
- None of the FDs in F2 is redundant. Hence, F2 is minimal cover.
- Hence, **set of functional dependencies F2 is the minimal cover for the set F.**

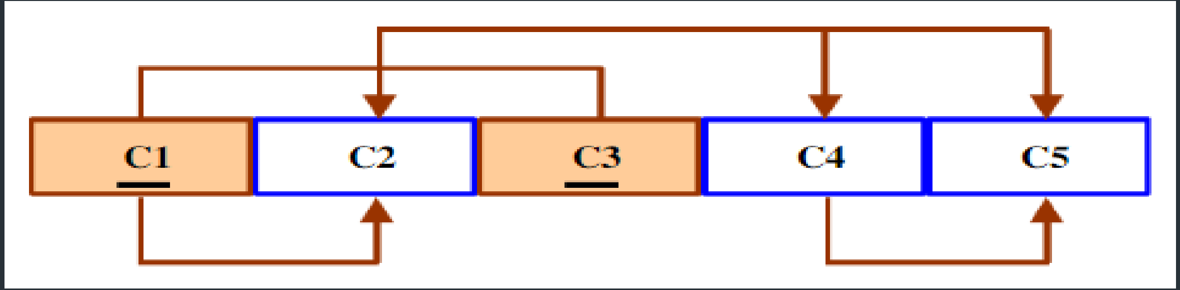
F2 = {A → C, C → D, C → I, EC → A, EC → B}

Dependency diagram



Dependency diagram





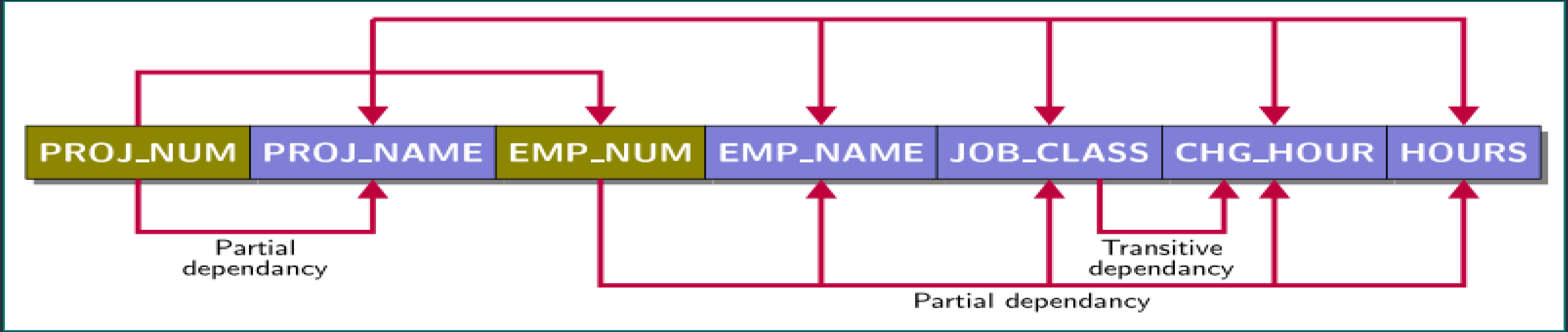
- The following dependencies are identified:
- C1 and C3, are the Primary Key.
- Partial Dependencies:

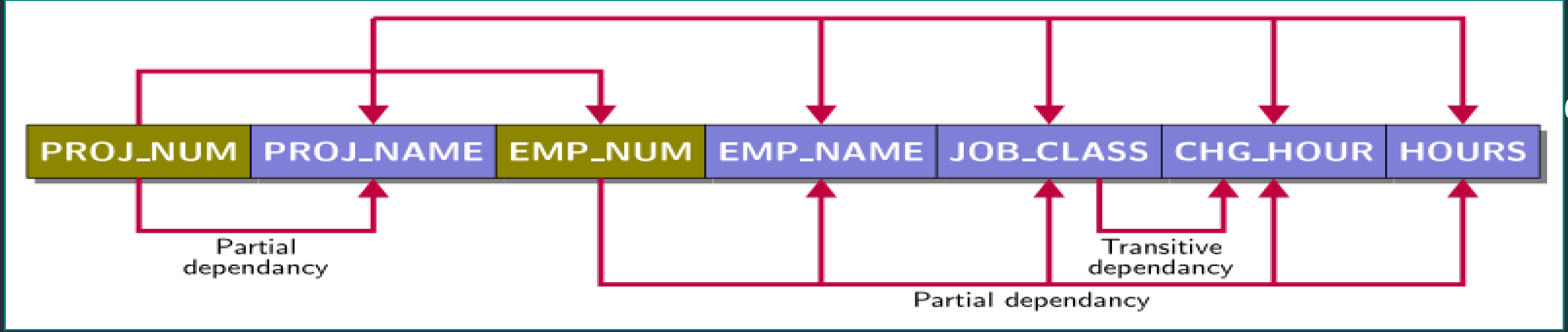
  C1—> C2
- Transitive Dependency:

  C4 —> C5

Dependency diagram

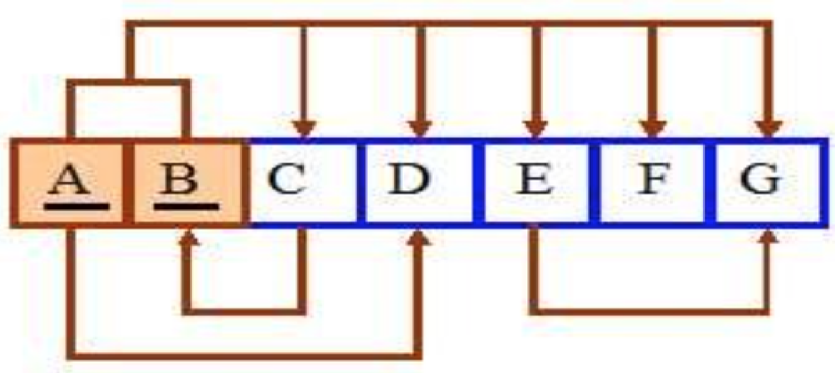
- A dependency diagram, shown in Figure, illustrates the various dependencies that might exist in a *non-normalized table*. A non-normalized table is one that has data redundancy in it.





- The following dependencies are identified:
- Proj_Num and Emp_Num, combined, are the PK.
- Partial Dependencies:
  - Proj_Num —> Proj_Name
  - Emp_Num —> Emp_Name, Job_Class, Chg_Hour, Hours
- Transitive Dependency:
  - Job_Class —> Chg_Hour

a) Given the following dependency diagram, label all the dependencies.



b) Redesign the database to 2NF. Show all the steps.
c) Redesign the database to 3NF. Show all the steps.


Thank You!