Relational
Database Design

- First characterize fully the data requirements of the prospective database users, which usually involves in textual descriptions.

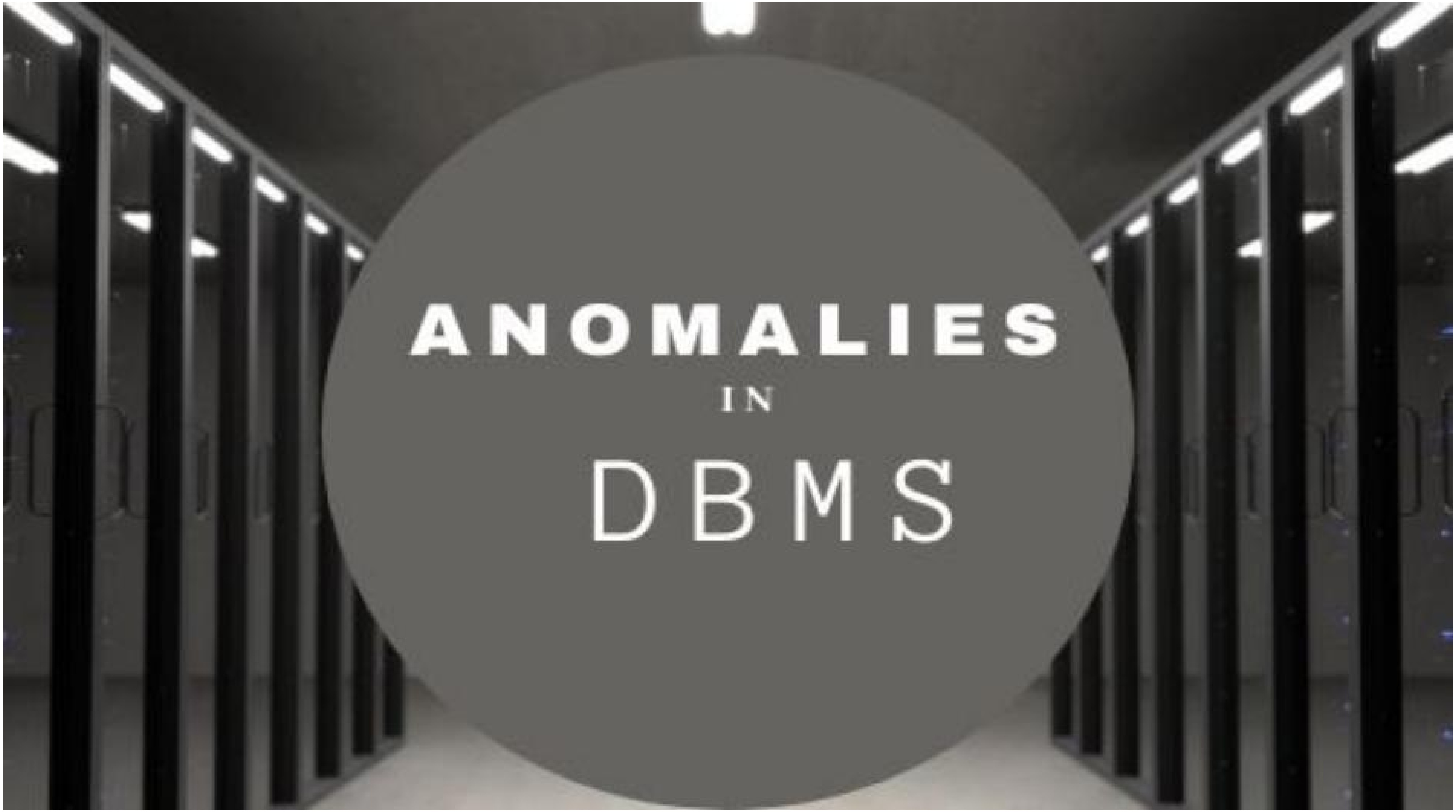
- Next, choose ER model to translate these requirements into a conceptual schema of the database.

- In the logical design phase, map the high level conceptual schema onto the implementation data model of the database system that will be used. The implementation data model is typically the Relational data model.

- Finally, use the resulting system specific database schema in the subsequent physical design phase, in which the physical features of the database are specified.

In designing a database schema, the major pitfalls which should be avoided are:

- Redundancy: it means repetition of the information
- Incompleteness: it means certain aspects of the enterprise may not be modeled due to difficulty or complexity


ANOMALIES
IN
DBMS

Bad Database Design/Concept of Anomalies

- **Database anomalies are the problems in relations that occur due to redundancy in the relations.**
- **Anomalies can often be caused when the tables that make up the database suffer from poor construction.**
- **These anomalies affect the process of inserting, deleting and updating data in the relations**
- **The intension of relational database theory is to eliminate anomalies from occurring in a database**


ANOMALIES
IN
DBMS
- Insertion
- Deletion
- Updation

Example:

Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, e_name for storing employee's name, e_address for storing employee's address, and e_dept for storing the department details in which the employee works.

e_id	e_name	e_address	e_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

Insert Anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if the e_dept field doesn't allow nulls.

Update Anomalies: In the above table, we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

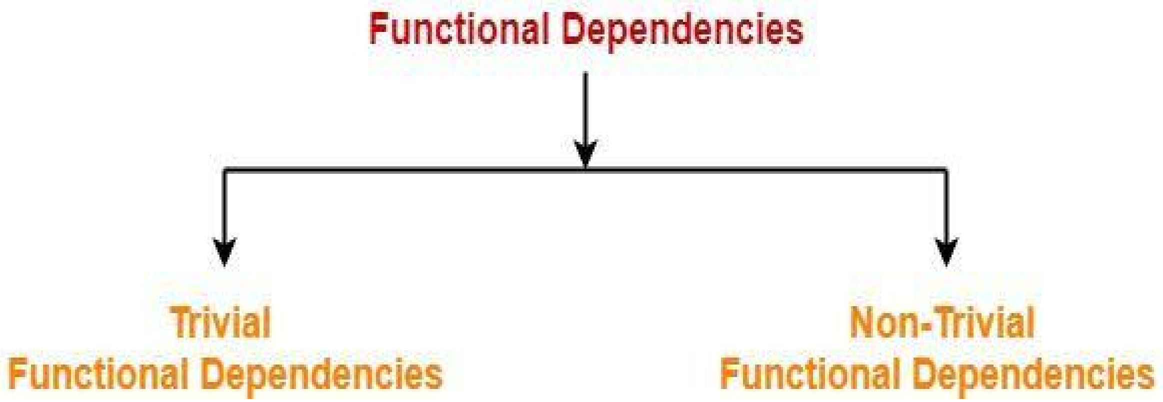
Delete Anomalies: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having e_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

So, we need to avoid these types of anomalies from the tables and maintain the integrity, accuracy of the database table. Therefore, we use the normalization concept in the database management system.

Functional Dependency

- Functional Dependency is the building block of normalization.

- 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. $X \rightarrow Y$



- In other words, if X functionally determines Y in relation R, then it is invalid to have two or more tuples that have the same X value, but different Y values in R

- The left side of FD is known as a **determinant**, the right side of the production is known as a **dependent**.

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

$$\text{Emp\_Id} \rightarrow \text{Emp\_Name}$$

We can say that Emp_Name is functionally dependent on Emp_Id.

Trivial Functional Dependencies-

A functional dependency X → Y is said to be trivial if and only if Y ⊆ X.
Thus, if RHS of a functional dependency is a subset of LHS, then it is called as a trivial functional dependency.

The examples of trivial functional dependencies are-

AB → A
AB → B
AB → AB

For example,
{Name, Course}→Course

Non-Trivial Functional Dependencies-

A functional dependency X → Y is said to be non-trivial if and only if Y∩X =Φ**.**
Thus, if there exists at least one attribute in the RHS of a functional dependency that is not a part of LHS, then it is called as a non-trivial functional dependency.

The examples of non-trivial functional dependencies are-

AB → BC
AB → CD,

For example, Prof→Grade

Inference Rules (IR)

The Armstrong's axioms are the basic inference rule which are used to conclude functional dependencies on a relational database.

The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.

Using the inference rule, we can derive additional functional dependency from the initial set.

- Reflexive Rule
- Augmentation Rule
- Transitive Rule
- Union Rule
- Decomposition Rule
- Pseudo transitive Rule

Reflexive Rule:

In the reflexive rule, if Y is a subset of X, then X determines Y.

If $X \supseteq Y$ then $X \rightarrow Y$

Example: X = {a, b, c, d, e} Y = {a, b, c}

Ex: {Name, Course}→Course

Augmentation Rule:

In augmentation, if X determines Y, then XZ determines YZ for any Z.

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

Example:

For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

Ex: *as Prof→Grade, therefore {Prof, Major}→{Grade, Major}*

Transitive Rule:

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

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

Ex: *as Course→Name and Name→Phone_no functional dependencies are present, therefore Course→Phone_no*

Union/Additive Rule:

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

If X → Y and X → Z then X → YZ

Ex:as Prof→Grade and Prof→Course FDs are present; therefore, Prof→{Grade, Course}

Decomposition Rule:

Decomposition rule is the reverse of union rule. If X determines Y and Z, then X determines Y and X determines Z separately.

If X → YZ then X → Y and X → Z

Ex: if Prof→{Grade, Course}, then this FD can be decomposed as Prof→Grade and Prof→Course

Composition Rule:

If X→Y and Z→W, then {X, Z}→{Y, W}.

Ex: if Prof→Grade and Name→Phone_no, then the FDs can be composed as {Prof, Name}→{Grade, Phone_no}

Pseudo transitive Rule:

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If X → Y and YZ → W then XZ → W

Ex: if Prof→Grade and {Grade, Major}→Course, then the FD {Prof, Major}→Course is valid

Closure of an Attribute Set-

- The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.
- Closure of attribute set {X} is denoted as {X}+.

Steps to Find Closure of an Attribute Set

Step-01: Add the attributes contained in the attribute set for which closure is being calculated to the result set.
Step-02: Recursively add the attributes to the result set which can be functionally determined from the attributes already contained in the result set.

Consider a relation:
R (A , B , C , D , E , F , G) with the functional dependencies-
A → BC
BC → DE
D → F
CF → G

Now, let us find the closure of some attributes and attribute sets-

$A^+ = \{A\}$
$= \{A, B, C\}$ (Using A → BC)
$= \{A, B, C, D, E\}$ (Using BC → DE)
$= \{A, B, C, D, E, F\}$ (Using D → F)
$= \{A, B, C, D, E, F, G\}$ (Using CF → G)

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

$D^+ = \{D\}$
$= \{D, F\}$ (Using D → F)

We can not determine any other attribute using attributes D and F contained in the result set.

Thus, $\mathbf{D^+ = \{D, F\}}$

$\{B, C\}^+ = \{B, C\}$
$= \{B, C, D, E\}$ (Using BC → DE)
$= \{B, C, D, E, F\}$ (Using D → F)
$= \{B, C, D, E, F, G\}$ (Using CF → G)

Thus, $\mathbf{\{B, C\}^+ = \{B, C, D, E, F, G\}}$

Consider the given functional dependencies-

$AB \rightarrow CD$

$AF \rightarrow D$

$DE \rightarrow F$

$C \rightarrow G$

$F \rightarrow E$

$G \rightarrow A$

Determine:

1. $\{ CF \}^+$
2. $\{ BG \}^+$
3. $\{ AF \}^+$
4. $\{ AB \}^+$

$\{ CF \}^+ = \{ C , F \}$

$= \{ C , F , G \}$ (Using $C \rightarrow G$)

$= \{ C , E , F , G \}$ (Using $F \rightarrow E$)

$= \{ A , C , E , F , G \}$ (Using $G \rightarrow A$)

$= \{ A , C , D , E , F , G \}$ (Using $AF \rightarrow D$)

$\{ BG \}^+ = \{ B , G \}$

$= \{ A , B , G \}$ (Using $G \rightarrow A$)

$= \{ A , B , C , D , G \}$ (Using $AB \rightarrow CD$)

$\{ AF \}^+ = \{ A , F \}$

$= \{ A , D , F \}$ (Using $AF \rightarrow D$)

$= \{ A , D , E , F \}$ (Using $F \rightarrow E$)

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

$= \{ A , B , C , D \}$ (Using $AB \rightarrow CD$)

$= \{ A , B , C , D , G \}$ (Using $C \rightarrow G$)

Class Activity

Given relational schema R(P Q R S T U V).
Functional dependency set is denoted by FD = { $P \rightarrow Q$, $QR \rightarrow ST$, $PTV \rightarrow V$ }.

Determine Closure of (QR)+ and (PR)+

Given relational schema R(P Q R S T).
There is a set of functional dependency denoted by FD = { $P \rightarrow QR$, $RS \rightarrow T$, $Q \rightarrow S$, $T \rightarrow P$ }.

Determine Closure of (T)+

Find Candidate Keys from Closure

- A set of minimal attribute(s) that can identify each tuple uniquely in the given relation is called as a candidate key.
- For any given relation, It is possible to have multiple candidate keys.

We can determine the candidate keys of a given relation using the following steps-

Step 1:

- Determine all **essential attributes** of the given relation.
- Essential attributes are those attributes which **are not present on RHS** of any functional dependency.
- Essential attributes are always a part of every candidate key.

Step 2:

- If all essential attributes together can determine all remaining non-essential attributes, then-

The combination of essential attributes is the candidate key.

It is the only possible candidate key.

Step 3:

- If all essential attributes together can not determine all remaining non-essential attributes, then-

The set of essential attributes and some non-essential attributes will be the candidate key(s).

In this case, multiple candidate keys are possible. To find the candidate keys, we check different combinations of essential and non-essential attributes.

Example

Let R(A, B, C, D, E, F) be a relation scheme with the following functional dependencies-

$$A \rightarrow B$$

$$C \rightarrow D$$

$$D \rightarrow E$$

Here, the attributes which are not present on RHS of any functional dependency are A, C and F.

So, essential attributes are- **A, C and F**.

Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies-

$$C \rightarrow F$$
$$E \rightarrow A$$
$$EC \rightarrow D$$
$$A \rightarrow B$$

Find the candidate keys?

Step-1:

- *Determine all essential attributes of the given relation.*
- *Here, essential attributes of the relation are, C and E.*
- *So, attributes C and E will definitely be a part of every candidate key.*

Step-2:

Now,

- *We will check if the essential attributes togeather can determine all remaining non-essential attributes.*
- *To check, we find the closure of CE, i.e {CE}+*

{CE}+	**= {C, E} = {C, E, F}**	**(Using C →F)**
	= {A, C, E, F}	**(Using E →A)**
	= {A, C, D, E, F}	**(Using EC →D)**
	= {A, B, C, D, E, F}	**(Using A → B)**

We, conclude that CE can determine all the attributes of the given relation.

So, CE is the only possible candidate key of the relation.

Let R = (A, B, C, D, E) be a relation scheme with the following dependencies-

$$AB \rightarrow C$$
$$C \rightarrow D$$
$$B \rightarrow E$$

Find the candidate keys?

Step-1:

- *Determine all essential attributes of the given relation.*
- *Here, essential attributes of the relation are, A and B.*
- *So, attributes A and B will definitely be a part of every candidate key.*

Step-2:

Now,

- *We will check if the essential attributes togeather can determine all remaining non-essential attributes.*
- *To check, we find the closure of AB, i.e {AB}+*

{AB}+	**= {A, B} = {A, B, C}**	**(Using AB →C)**
	= {A, B, C, D}	**(Using C →D)**
	= {A, B, C, D, E}	**(Using B →E)**

We, conclude that AB can determine all the attributes of the given relation.

So, AB is the only possible candidate key of the relation.

Consider the relation scheme R(E, F, G, H, I, J, K, L, M, N) and the set of functional dependencies-

$\{E, F\} \rightarrow \{G\}$

$\{F\} \rightarrow \{I, J\}$

$\{E, H\} \rightarrow \{K, L\}$

$\{K\} \rightarrow \{M\}$

$\{L\} \rightarrow \{N\}$

Find the candidate keys:

Step-01:

- Determine all essential attributes of the given relation.
- Essential attributes of the relation are- E, F and H.
- So, attributes E, F and H will definitely be a part of every candidate key.

Step-02:

- We will check if the essential attributes together can determine all remaining non-essential attributes.
- To check, we find the closure of EFH.

$$\begin{aligned}
\{EFH\}^+ &= \{E, F, H\} \\
&= \{E, F, G, H\} \text{ (Using } EF \rightarrow G \text{)} \\
&= \{E, F, G, H, I, J\} \text{ (Using } F \rightarrow IJ \text{)} \\
&= \{E, F, G, H, I, J, K, L\} \text{ (Using } EH \rightarrow KL \text{)} \\
&= \{E, F, G, H, I, J, K, L, M\} \text{ (Using } K \rightarrow M \text{)} \\
&= \{E, F, G, H, I, J, K, L, M, N\} \text{ (Using } L \rightarrow N \text{)}
\end{aligned}$$

We conclude that EFH can determine all the attributes of the given relation.

So, EFH is the only possible candidate key of the relation.

R = ABCD, F={AB -> C, BC -> D, CD -> A}

Essential attributes is B.

B+ = {B}

Since closure of B cannot determine all attributes in the relation, thus the essential attribute (B) must be combined with non-essential attributes to find the candidate key.

(AB)+ = {ABCD}
(BC)+ = {BCDA}

Hence, **AB & BC** are the candidate keys.

Find the candidate keys for these:

1. R = (ABCDE), F = {A -> C, E -> D, B -> C}
2. R = ABCDE, F = {A -> BE, C -> BE, B -> D}
3. R = ABCDEF, F = {A -> B, B -> D, C -> D, E -> F}
4. R = ABCD, F={AB -> C, C -> D, CD -> A}
5. R = ABCD, F={A -> BCD, C -> A}

Canonical Cover of Functional Dependencies in DBMS

- Whenever a user updates the database, the system must check whether any of the functional dependencies are getting violated in this process. If there is a violation of dependencies in the new database state, the system must roll back.

- Working with a huge set of functional dependencies can cause unnecessary added computational time. **This is where the canonical cover comes into play.**

- A canonical cover of a set of functional dependencies F is a simplified set of functional dependencies that has the same closure as the original set F.

- A canonical cover is a simplified and reduced version of the given set of functional dependencies. Since it is a reduced version, it is also called as Irreducible set.

Canonical Cover: A canonical cover F_c of a set of functional dependencies **F** such that ALL the following properties are satisfied:

- **F** logically implies all dependencies in F_c
- F_c logically implies all dependencies in **F**.
- No functional dependency in F_c contains as **extraneous** attribute.
- Each left side of a functional dependency in F_c is unique. That is, there are no two dependencies $\alpha_1 \rightarrow \beta_1$ and $\alpha_2 \rightarrow \beta_2$ in such that $\alpha_1 \rightarrow \alpha_2$.

*An attribute of a functional dependency is said to be **extraneous** if we can remove it without changing the closure of the set of functional dependencies.*

Algorithm to compute canonical cover for set F.

```
Repeat
1. Use the union rule to replace any dependencies in α₁ → β1 and α₂ →
β₂ with α₁ → β₁β₂
2. Find a functional dependency α → β with an extraneous attribute
either in α or in β.
3. If an extraneous attribute is found, delete it from α → β.
until F does not change
```

Given a relational Schema R(A, B, C, D) and set of Function Dependency FD = { B → A, AD → BC, C → ABD }. Find the canonical cover?

Solution: Given FD = { B → A, AD → BC, C → ABD }, now decompose the FD using decomposition rule(Armstrong Axiom).

1. B → A
2. AD → B (using decomposition inference rule on AD → BC)
3. AD → C (using decomposition inference rule on AD → BC)
4. C → A (using decomposition inference rule on C → ABD)
5. C → B (using decomposition inference rule on C → ABD)
6. C → D (using decomposition inference rule on C → ABD)

Now set of FD = { B → A, AD → B, AD → C, C → A, C → B, C → D }

The next step is to find closure of the left side of each of the given FD by including that FD and excluding that FD, if closure in both cases are same then that FD is redundant and we remove that FD from the given set, otherwise if both the closures are different then we do not exclude that FD.

Calculating closure of all FD { B → A, AD → B, AD → C, C → A, C → B, C → D }

1a. Closure B+ = BA using FD = { **B → A**, AD → B, AD → C, C → A, C → B, C → D }

1b. Closure B+ = B using FD = { AD → B, AD → C, C → A, C → B, C → D }

From 1 a and 1 b, we found that both the Closure(by including **B → A** and excluding **B → A**) are not equivalent, hence FD B → A is important and cannot be removed from the set of FD.

2 a. Closure AD+ = ADBC using FD = { B →A, **AD → B**, AD → C, C → A, C → B, C → D }

2 b. Closure AD+ = ADCB using FD = { B → A, AD → C, C → A, C → B, C → D }

From 2 a and 2 b, we found that both the Closure (by including **AD → B** and excluding **AD → B**) are equivalent, hence FD **AD → B** is not important and can be removed from the set of FD.

Hence resultant FD = { B → A, AD → C, C → A, C → B, C → D }

3 a. Closure AD+ = ADCB using FD = { B →A, **AD → C**, C → A, C → B, C → D }

3 b. Closure AD+ = AD using FD = { B → A, C → A, C → B, C → D }

From 3 a and 3 b, we found that both the Closure (by including **AD → C** and excluding **AD → C**) are not equivalent, hence FD AD → C is important and cannot be removed from the set of FD.

Hence resultant FD = { B → A, AD → C, C → A, C → B, C → D }

4 a. Closure C+ = CABD using FD = { B →A, AD → C, **C → A**, C → B, C → D }

4 b. Closure C+ = CBDA using FD = { B → A, AD → C, C → B, C → D }

From 4 a and 4 b, we found that both the Closure (by including **C → A** and excluding **C → A**) are equivalent, hence FD **C → A** is not important and can be removed from the set of FD.

Hence resultant FD = { B → A, AD → C, C → B, C → D }

5 a. Closure C+ = CBDA using FD = { B →A, AD → C, **C → B**, C → D }

5 b. Closure C+ = CD using FD = { B → A, AD → C, C → D }

From 5 a and 5 b, we found that both the Closure (by including **C → B** and excluding **C → B**) are not equivalent, hence FD **C → B** is important and cannot be removed from the set of FD.

Hence resultant FD = { B → A, AD → C, C → B, C → D }

6 a. Closure C+ = CDBA using FD = { B →A, AD → C, C → B, **C → D** }

6 b. Closure C+ = CBA using FD = { B → A, AD → C, C → B }

From 6 a and 6 b, we found that both the Closure(by including **C → D** and excluding **C → D**) are not equivalent, hence FD **C → D** is important and cannot be removed from the set of FD.

Hence resultant FD = { B → A, AD → C, C → B, C → D }

- Since FD = { B → A, AD → C, C → B, C → D } is resultant FD, now we have checked the redundancy of attribute, since the left side of FD AD → C has two attributes, let's check their importance, i.e. whether they both are important or only one.

Closure AD+ = ADCB using FD = { B →A, **AD → C**, C → B, C → D }

Closure A+ = A using FD = { B →A, **AD → C**, C → B, C → D }

Closure D+ = D using FD = { B →A, **AD → C**, C → B, C → D }

Since the closure of AD+, A+, D+ that we found are not all equivalent, hence in FD AD → C, both A and D are important attributes and cannot be removed.

Hence resultant FD = { B → A, AD → C, C → B, C → D } and we can rewrite as

FD = { B → A, AD → C, C → BD } is Canonical Cover of FD = { B → A, AD → BC, C → ABD }.

Assignments:

Given a relational Schema R(W, X, Y, Z) and set of Function Dependency FD = { W → X, Y → X, Z → WXY, WY → Z }.
Find the canonical cover?

Given a relational Schema R(V, W, X, Y, Z) and set of Function Dependency FD = { V → W, VW → X, Y → VXZ }.
Find the canonical cover?

The End