## Functional Dependency-

In any relation, a functional dependency $\alpha \rightarrow \beta$ holds if-
Two tuples having same value of attribute $\alpha$ also have same value for attribute $\beta$.

If $\alpha$ and $\beta$ are the two sets of attributes in a relational table R where-

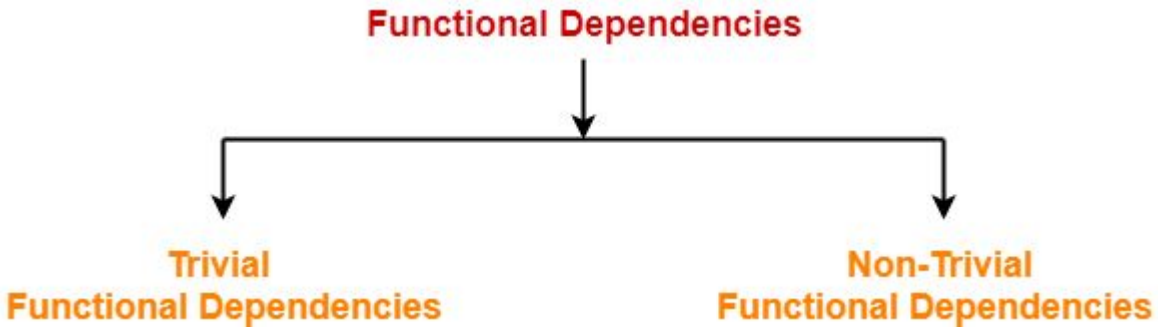
- $\alpha \subseteq R$
- $\beta \subseteq R$

Then, for a functional dependency to exist from $\alpha$ to $\beta$,

If t1[α] = t2[α], then t1[β] = t2[β]

| α | β |
| --- | --- |
| t1[α] | t1[β] |
| t2[α] | t2[β] |
| ……. | ……. |

# Types Of Functional Dependencies-



The examples of trivial functional dependencies are-

- $AB \rightarrow A$
- $AB \rightarrow B$
- $AB \rightarrow AB$

# Armstrong's Axioms RULE

**Reflexivity-**

If B is a subset of A, then A → B always holds.

**Transitivity-**

If A → B and B → C, then A → C always holds.

**Augmentation-**

If A → B, then AC → BC always holds.

# Secondary Rule of Armstrong's Axioms

**Union :** If A → B and A → C, then A → BC always holds.

**Decomposition-** If A → BC, then A → B and A → C always holds.

**Pseudo Transitivity :** if A -->B and BC-->D then AC-->D

**Composition** : if A--->B and C-->D then AC--->BD

A functional dependency X → Y will always hold if all the values of X are unique (different) irrespective of the values of Y.

The following functional dependencies will always hold since all the values of attribute ‘A’ are unique-

| A | B | C | D | E |
|---|---|---|---|---|
| 5 | 4 | 3 | 2 | 2 |
| 8 | 5 | 3 | 2 | 1 |
| 1 | 9 | 3 | 3 | 5 |
| 4 | 7 | 3 | 3 | 8 |

- $A \rightarrow B$
- $A \rightarrow BC$
- $A \rightarrow CD$
- $A \rightarrow BCD$
- $A \rightarrow DE$
- $A \rightarrow BCDE$

In general, we can say following functional dependency will always hold-

A functional dependency X → Y will always hold if all the values of Y are same irrespective of the values of X.

| A | B | C | D | E |
|---|---|---|---|---|
| 5 | 4 | 3 | 2 | 2 |
| 8 | 5 | 3 | 2 | 1 |
| 1 | 9 | 3 | 3 | 5 |
| 4 | 7 | 3 | 3 | 8 |

The following functional dependencies will always hold since all the values of attribute ‘C’ are same-

- $A \rightarrow C$
- $AB \rightarrow C$
- $ABDE \rightarrow C$
- $DE \rightarrow C$
- $AE \rightarrow C$

In general, we can say following functional dependency will always hold true-

## 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-

Add the attributes contained in the attribute set for which closure is being calculated to the result set.

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-

$$A \rightarrow BC$$

$$BC \rightarrow DE$$

$$D \rightarrow F$$

$$CF \rightarrow G$$

## 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,

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

## Equivalence of Two Sets of Functional Dependencies-

- Two different sets of functional dependencies for a given relation may or may not be equivalent.
- If F and G are the two sets of functional dependencies, then following 3 cases are possible-

**Case-01**: F covers G ($F \supseteq G$)

**Case-02:** G covers F ($G \supseteq F$)

**Case-03:** Both F and G cover each other ($F = G$)

A relation R (A , C , D , E , H) is having two functional dependencies sets F and G as shown-

**Set F-**

$$A \rightarrow C$$

$$AC \rightarrow D$$

$$E \rightarrow AD$$

$$E \rightarrow H$$

**Set G-**

$$A \rightarrow CD$$

$$E \rightarrow AH$$

Which of the following holds true?

(A) $G \supseteq F$

(B) $F \supseteq G$

(C) $F = G$

(D) All of the above

**Step-01:**

- $(A)^+$ = { A , C , D } // closure of left side of A → CD using set G
- $(E)^+$ = { A , C , D , E , H } // closure of left side of E → AH using set G

**Step-02:**

- $(A)^+$ = { A , C , D } // closure of left side of A → CD using set F
- $(E)^+$ = { A , C , D , E , H } // closure of left side of E → AH using set F

## Step-03:

Comparing the results of Step-01 and Step-02, we find-

- Functional dependencies of set F can determine all the attributes which have been determined by the functional dependencies of set G.
- Thus, we conclude F covers G i.e. $F \supseteq G$.

# Determining whether G covers F-

## Step-01:

- $(A)^+$ = { A , C , D } // closure of left side of A → C using set F
- $(AC)^+$ = { A , C , D } // closure of left side of AC → D using set F
- $(E)^+$ = { A , C , D , E , H } // closure of left side of E → AD and E → H using set F

**Step-02:**

- $(A)^+$ = { A , C , D } // closure of left side of A → C using set G
- $(AC)^+$ = { A , C , D } // closure of left side of AC → D using set G
- $(E)^+$ = { A , C , D , E , H } // closure of left side of E → AD and E → H using set G

**Step-03:**

Comparing the results of Step-01 and Step-02, we find-

- Functional dependencies of set G can determine all the attributes which have been determined by the functional dependencies of set F.
- Thus, we conclude G covers F i.e. $G \supseteq F$.

## Determining whether both F and G cover each other-

- From Step-01, we conclude F covers G.
- From Step-02, we conclude G covers F.
- Thus, we conclude both F and G cover each other i.e. F = G.

Thus, Option (D) is correct.

## Canonical Cover in DBMS-

- 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**.

**Need-**

- Working with the set containing extraneous functional dependencies increases the computation time.
- Therefore, the given set is reduced by eliminating the useless functional dependencies.
- This reduces the computation time and working with the irreducible set becomes easier.

**Steps To Find Canonical Cover-**

**Step-01:**

Write the given set of functional dependencies in such a way that each functional dependency contains exactly one attribute on its right side.

The functional dependency $X \rightarrow YZ$ will be written as-

$$X \rightarrow Y$$

$$X \rightarrow Z$$

**Step-02:**

- Consider each functional dependency one by one from the set obtained in Step-01.
- Determine whether it is essential or non-essential.

To determine whether a functional dependency is essential or not, compute the closure of its left side-

- Once by considering that the particular functional dependency is present in the set
- Once by considering that the particular functional dependency is not present in the set

Then following two cases are possible-

## Case-01: Results Come Out to be Same-

If results come out to be same,

- It means that the presence or absence of that functional dependency does not create any difference.
- Thus, it is non-essential.
- Eliminate that functional dependency from the set.

## NOTE-

- Eliminate the non-essential functional dependency from the set as soon as it is discovered.
- Do not consider it while checking the  essentiality of other functional dependencies.

**Case-01: Results Come Out to be Different-**

If results come out to be different,

- It means that the presence or absence of that functional dependency creates a difference.
- Thus, it is essential.
- Do not eliminate that functional dependency from the set.
- Mark that functional dependency as essential.

**Step-03:**

- Consider the newly obtained set of functional dependencies after performing Step-02.
- Check if there is any functional dependency that contains more than one attribute on its left side.

# Key

A key is a set of attributes that can identify each tuple uniquely in the given relation.

Super key

Candidate key

Primary key

**Super Key-**

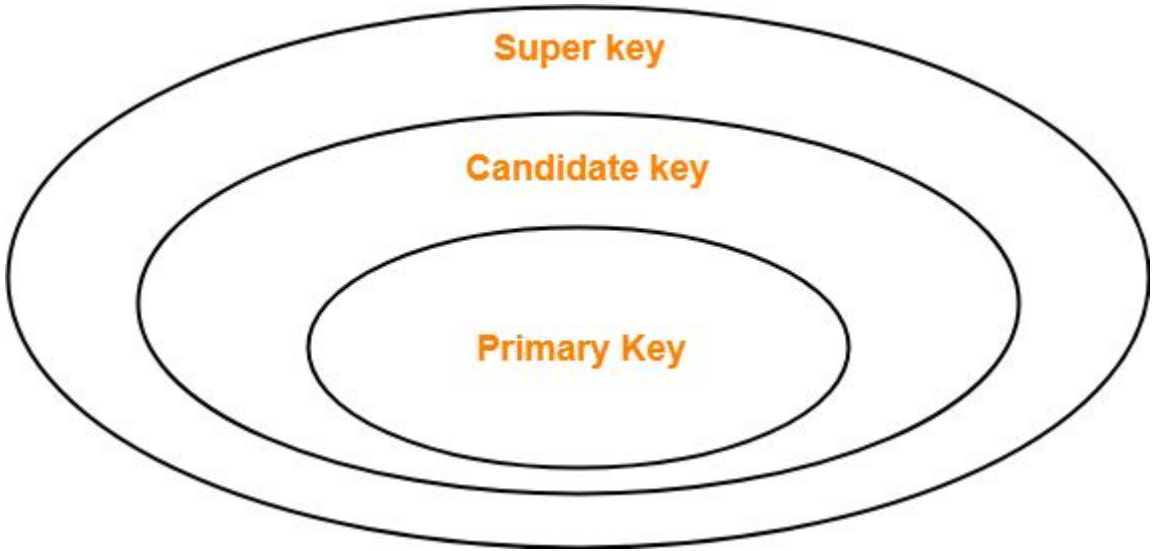
- A super key is a set of attributes that can identify each tuple uniquely in the given relation.
- A super key is not restricted to have any specific number of attributes.
- Thus, a super key may consist of any number of attributes.

**2. Candidate Key-**

A minimal super key is called as a candidate key.

**. Primary Key-**

Candidate key that the database designer implements is called as a primary key.



| *StudentID* | *StudentName* | *StudentEmail* | *StudentAge* | *CPI* |
|---|---|---|---|---|
| 2345 | Shankar | shankar@math | X | 9.4 |
| 1287 | Swati | swati@ee | 19 | 9.5 |
| 7853 | Shankar | shankar@cse | 19 | 9.4 |
| 9876 | Swati | swati@mech | 18 | 9.3 |
| 8765 | Ganesh | ganesh@civil | 19 | 8.7 |

For ***(StudentName, StudentAge)*** to be a key for this instance, the value X should NOT be equal to _____.