**QUESTIONS**

Ques 1.

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

C → F

E → A

EC → D

A → B

Which of the following is a key for R?

1. CD
2. EC
3. AE
4. AC

Also, determine the total number of candidate keys and super keys.

**Solution-**

We will find candidate keys of the given relation in the following steps-

**Step-01:**

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

**Step-02:**

Now,

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

So, we have-

{ 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.

***Thus, Option (B) is correct.***

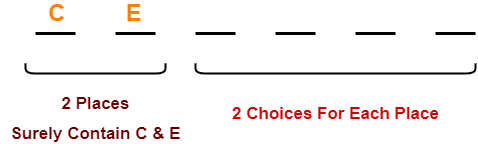
**Total Number of Candidate Keys-**

Only one candidate key CE is possible.

**Total Number of Super Keys-**

There are total 6 attributes in the given relation of which-

* There are 2 essential attributes- C and E.
* Remaining 4 attributes are non-essential attributes.
* Essential attributes will be definitely present in every key.
* Non-essential attributes may or may not be taken in every super key.



So, number of super keys possible = 2 x 2 x 2 x 2 = 16.

Thus, total number of super keys possible = 16.

**Problem-02:**

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

AB → C

C → D

B → E

Determine the total number of candidate keys and super keys.

**Solution-**

We will find candidate keys of the given relation in the following steps-

**Step-01:**

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

**Step-02:**

Now,

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

So, we have-

{ 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.

***Thus, AB is the only possible candidate key of the relation.***

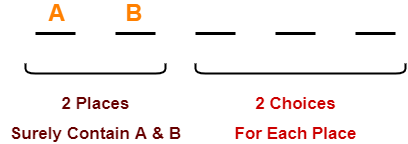
**Total Number of Candidate Keys-**

Only one candidate key AB is possible.

**Total Number of Super Keys-**

There are total 5 attributes in the given relation of which-

* There are 2 essential attributes- A and B.
* Remaining 3 attributes are non-essential attributes.
* Essential attributes will be definitely present in every key.
* Non-essential attributes may or may not be taken in every super key.



So, number of super keys possible = 2 x 2 x 2 = 8.

Thus, total number of super keys possible = 8.

**Problem-03:**

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 }

What is the key for R?

1. { E, F }
2. { E, F, H }
3. { E, F, H, K, L }
4. { E }

Also, determine the total number of candidate keys and super keys.

**Solution-**

We will find candidate keys of the given relation in the following steps-

**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:**

Now,

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

So, we have-

{ EFH }+

= { E , F , H }

= { E , F , G , H } ( Using EF → G )

= { E , F , G , H , I , J } ( Using F → IJ )

= { E , F , G , H , I , J , K , L } ( Using EH → KL )

= { E , F , G , H , I , J , K , L , M } ( Using K → M )

= { E , F , G , H , I , J , K , L , M , N } ( Using L → N )

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

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

***Thus, Option (B) is correct.***

**Total Number of Candidate Keys-**

Only one candidate key EFH is possible.

**Total Number of Super Keys-**

There are total 10 attributes in the given relation of which-

* There are 3 essential attributes- E, F and H.
* Remaining 7 attributes are non-essential attributes.
* Essential attributes will be definitely present in every key.
* Non-essential attributes may or may not be taken in every super key.

A picture containing text

Description automatically generated

So, number of super keys possible = 2 x 2 x 2 x 2 x 2 x 2 x 2 = 128.

Thus, total number of super keys possible = 128.

**Problem-04:**

Consider the relation scheme R(A, B, C, D, E, H) and the set of functional dependencies-

A → B

BC → D

E → C

D → A

What are the candidate keys of R?

1. AE, BE
2. AE, BE, DE
3. AEH, BEH, BCH
4. AEH, BEH, DEH

**Solution-**

We will find candidate keys of the given relation in the following steps-

**Step-01:**

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

The only possible option is (D).

***Thus, Option (D) is correct.***

**Functional Dependency**

**Functional Dependency-**

|  |
| --- |
| In any relation, a functional dependency α → β holds if-  Two tuples having same value of attribute α also have same value for attribute β. |

**Mathematically,**

If α and β are the two sets of attributes in a relational table R where-

* α ⊆ R
* β ⊆ R

Then, for a functional dependency to exist from α to β,

If t1[α] = t2[α], then t1[β] = t2[β]

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

|  |
| --- |
| **fd : α** → **β** |

**Types Of Functional Dependencies-**

There are two types of functional dependencies-

Diagram

Description automatically generated

1. Trivial Functional Dependencies
2. Non-trivial Functional Dependencies

**1. 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.

**Examples-**

The examples of trivial functional dependencies are-

* AB → A
* AB → B
* AB → AB

**2. Non-Trivial Functional Dependencies-**

* A functional dependency X → Y is said to be non-trivial if and only if Y ⊄ X.
* 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.

**Examples-**

The examples of non-trivial functional dependencies are-

* AB → BC
* AB → CD

**Inference Rules-**

**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.

**Decomposition-**

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

**Composition-**

If A → B and C → D, then AC → BD always holds.

**Additive-**

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

**Rules for Functional Dependency-**

**Rule-01:**

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

**Example-**

Consider the following table-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 ‘A’ are unique-

* A → B
* A → BC
* A → CD
* A → BCD
* A → DE
* A → BCDE

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

|  |
| --- |
| **A** → **Any combination of attributes A, B, C, D, E** |

Similar will be the case for attributes B and E.

**Rule-02:**

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

**Example-**

Consider the following table-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 → C
* AB → C
* ABDE → C
* DE → C
* AE → C

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

|  |
| --- |
| **Any combination of attributes A, B, C, D, E**→ **C** |

Combining Rule-01 and Rule-02 we can say-

|  |
| --- |
| In general, a functional dependency α → β always holds-  If either all values of α are unique or if all values of β are same or both. |

**Rule-03:**

For a functional dependency X → Y to hold, if two tuples in the table agree on the value of attribute X, then they must also agree on the value of attribute Y.

**Rule-04:**

For a functional dependency X → Y, violation will occur only when for two or more same values of X, the corresponding Y values are different.

**Equivalence of Two Sets of Functional Dependencies-**

Before you go through this article, make sure that you have gone through the previous article on [**Functional Dependency**](https://www.gatevidyalay.com/functional-dependency-in-dbms/).

In DBMS,

* 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 ⊇ G)

**Case-02:** G covers F (G ⊇ F)

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

**Case-01: Determining Whether F Covers G-**

Following steps are followed to determine whether F covers G or not-

**Step-01:**

* Take the functional dependencies of set G into consideration.
* For each functional dependency X → Y, find the closure of X using the functional dependencies of set G.

**Step-02:**

* Take the functional dependencies of set G into consideration.
* For each functional dependency X → Y, find the closure of X using the functional dependencies of set F.

**Step-03:**

* Compare the results of Step-01 and Step-02.
* If the functional dependencies of set F has determined all those attributes that were determined by the functional dependencies of set G, then it means F covers G.
* Thus, we conclude F covers G (F ⊇ G) otherwise not.

**Case-02: Determining Whether G Covers F-**

Following steps are followed to determine whether G covers F or not-

**Step-01:**

* Take the functional dependencies of set F into consideration.
* For each functional dependency X → Y, find the closure of X using the functional dependencies of set F.

**Step-02:**

* Take the functional dependencies of set F into consideration.
* For each functional dependency X → Y, find the closure of X using the functional dependencies of set G.

**Step-03:**

* Compare the results of Step-01 and Step-02.
* If the functional dependencies of set G has determined all those attributes that were determined by the functional dependencies of set F, then it means G covers F.
* Thus, we conclude G covers F (G ⊇ F) otherwise not.

**Case-03: Determining Whether Both F and G Cover Each Other-**

* If F covers G and G covers F, then both F and G cover each other.
* Thus, if both the above cases hold true, we conclude both F and G cover each other (F = G).

**PRACTICE PROBLEM BASED ON EQUIVALENCE OF FUNCTIONAL DEPENDENCIES-**

**Problem-**

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

**Set F-**

A → C

AC → D

E → AD

E → H

**Set G-**

A → CD

E → AH

Which of the following holds true?

(A) G ⊇ F

(B) F ⊇ G

(C) F = G

(D) All of the above

**Solution-**

**Determining whether F covers G-**

**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 ⊇ 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 ⊇ 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-**

Before you go through this article, make sure that you have gone through the previous article on [**Functional Dependency in DBMS**](https://www.gatevidyalay.com/functional-dependency-in-dbms/).

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**.

**Characteristics-**

* Canonical cover is free from all the extraneous functional dependencies.
* The closure of canonical cover is same as that of the given set of functional dependencies.
* Canonical cover is not unique and may be more than one for a given set of functional dependencies.

**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.

**Example-**

The functional dependency X → YZ will be written as-

X → Y

X → 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.

Then following two cases are possible-

**Case-01: No-**

* There exists no functional dependency containing more than one attribute on its left side.
* In this case, the set obtained in Step-02 is the canonical cover.

**Case-01: Yes-**

* There exists at least one functional dependency containing more than one attribute on its left side.
* In this case, consider all such functional dependencies one by one.
* Check if their left side can be reduced.

Use the following steps to perform a check-

* Consider a functional dependency.
* Compute the closure of all the possible subsets of the left side of that functional dependency.
* If any of the subsets produce the same closure result as produced by the entire left side, then replace the left side with that subset.

After this step is complete, the set obtained is the canonical cover.

**PRACTICE PROBLEM BASED ON FINDING CANONICAL COVER-**

**Problem-**

The following functional dependencies hold true for the relational scheme R ( W , X , Y , Z ) –

X → W

WZ → XY

Y → WXZ

Write the irreducible equivalent for this set of functional dependencies.

**Solution-**

**Step-01:**

Write all the functional dependencies such that each contains exactly one attribute on its right side-

X → W

WZ → X

WZ → Y

Y → W

Y → X

Y → Z

**Step-02:**

Check the essentiality of each functional dependency one by one.

**For X → W:**

* Considering X → W, (X)+ = { X , W }
* Ignoring X → W, (X)+ = { X }

Now,

* Clearly, the two results are different.
* Thus, we conclude that X → W is essential and can not be eliminated.

**For WZ → X:**

* Considering WZ → X, (WZ)+ = { W , X , Y , Z }
* Ignoring WZ → X, (WZ)+ = { W , X , Y , Z }

Now,

* Clearly, the two results are same.
* Thus, we conclude that WZ → X is non-essential and can be eliminated.

Eliminating WZ → X, our set of functional dependencies reduces to-

X → W

WZ → Y

Y → W

Y → X

Y → Z

Now, we will consider this reduced set in further checks.

**For WZ → Y:**

* Considering WZ → Y, (WZ)+ = { W , X , Y , Z }
* Ignoring WZ → Y, (WZ)+ = { W , Z }

Now,

* Clearly, the two results are different.
* Thus, we conclude that WZ → Y is essential and can not be eliminated.

**For Y → W:**

* Considering Y → W, (Y)+ = { W , X , Y , Z }
* Ignoring Y → W, (Y)+ = { W , X , Y , Z }

Now,

* Clearly, the two results are same.
* Thus, we conclude that Y → W is non-essential and can be eliminated.

Eliminating Y → W, our set of functional dependencies reduces to-

X → W

WZ → Y

Y → X

Y → Z

**For Y → X:**

* Considering Y → X, (Y)+ = { W , X , Y , Z }
* Ignoring Y → X, (Y)+ = { Y , Z }

Now,

* Clearly, the two results are different.
* Thus, we conclude that Y → X is essential and can not be eliminated.

**For Y → Z:**

* Considering Y → Z, (Y)+ = { W , X , Y , Z }
* Ignoring Y → Z, (Y)+ = { W , X , Y }

Now,

* Clearly, the two results are different.
* Thus, we conclude that Y → Z is essential and can not be eliminated.

From here, our essential functional dependencies are-

X → W

WZ → Y

Y → X

Y → Z

**Step-03:**

* Consider the functional dependencies having more than one attribute on their left side.
* Check if their left side can be reduced.

In our set,

* Only WZ → Y contains more than one attribute on its left side.
* Considering WZ → Y, (WZ)+ = { W , X , Y , Z }

Now,

* Consider all the possible subsets of WZ.
* Check if the closure result of any subset matches to the closure result of WZ.

(W)+ = { W }

(Z)+ = { Z }

Clearly,

* None of the subsets have the same closure result same as that of the entire left side.
* Thus, we conclude that we can not write WZ → Y as W → Y or Z → Y.
* Thus, set of functional dependencies obtained in step-02 is the canonical cover.

Finally, the canonical cover is-

X → W

WZ → Y

Y → X

Y → Z

**Canonical Cover**