**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.

**Example-**

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-

### ****Closure of attribute A-****

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,

**A+ = { A , B , C , D , E , F , G }**

### ****Closure of attribute D-****

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,

**D+ = { D , F }**

### ****Closure of attribute set {B, C}-****

{ 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,

**{ B , C }+ = { B , C , D , E , F , G }**

## ****Finding the Keys Using Closure-****

## ****Super Key-****

* If the closure result of an attribute set contains all the attributes of the relation, then that attribute set is called as a super key of that relation.
* Thus, we can say-

**“The closure of a super key is the entire relation schema.”**

### ****Example-****

In the above example,

* The closure of attribute A is the entire relation schema.
* Thus, attribute A is a super key for that relation.

## ****Candidate Key-****

* If there exists no subset of an attribute set whose closure contains all the attributes of the relation, then that attribute set is called as a candidate key of that relation.

### ****Example-****

In the above example,

* No subset of attribute A contains all the attributes of the relation.
* Thus, attribute A is also a candidate key for that relation.

## ****PRACTICE PROBLEM BASED ON FINDING CLOSURE OF AN ATTRIBUTE SET-****

## ****Problem-****

Consider the given functional dependencies-

AB → CD

AF → D

DE → F

C → G

F → E

G → A

Which of the following options is false?

(A) { CF }+ = { A , C , D , E , F , G }

(B) { BG }+ = { A , B , C , D , G }

(C) { AF }+ = { A , C , D , E , F , G }

(D) { AB }+ = { A , C , D , F ,G }

## ****Solution-****

Let us check each option one by one-

### ****Option-(A):****

{ CF }+ = { C , F }

= { C , F , G } ( Using C → G )

= { C , E , F , G } ( Using F → E )

= { A , C , E , E , F } ( Using G → A )

= { A , C , D , E , F , G } ( Using AF → D )

Since, our obtained result set is same as the given result set, so, it means it is correctly given.

### ****Option-(B):****

{ BG }+ = { B , G }

= { A , B , G } ( Using G → A )

= { A , B , C , D , G } ( Using AB → CD )

Since, our obtained result set is same as the given result set, so, it means it is correctly given.

### ****Option-(C):****

{ AF }+ = { A , F }

= { A , D , F } ( Using AF → D )

= { A , D , E , F } ( Using F → E )

Since, our obtained result set is different from the given result set, so,it means it is not correctly given.

### ****Option-(D):****

{ AB }+ = { A , B }

= { A , B , C , D } ( Using AB → CD )

= { A , B , C , D , G } ( Using C → G )

Since, our obtained result set is different from the given result set, so,it means it is not correctly given.

Thus,

**Option (C) and Option (D) are correct.**

# ****Closure Of Functional Dependency : Introduction****

* **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. These are:**

**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.**

**Example-2 : Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies.**

**FD1 : A 🡪 BC**

**FD2 : C 🡪 B**

**FD3 : D 🡪 E**

**FD4 : E  🡪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}** |

### ****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 🡪 B**

**FD2 : B  🡪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 🡪 BC**

**FD2 : C🡪  B**

**FD3 : D🡪 E**

**FD4 : E  🡪D**

**Now, calculating the closure of the attributes as :**

|  |
| --- |
| **{A}+ = {A, B, C}**  **{B}+ = {B}**  **{C}+ = {C, B}**  **{D}+ = {E, D}**  **{E}+ = {E, D}** |

**In this case, a single attribute is unable to determine all the attribute on its own like in previous example. Here, we need to combine two or more attributes to determine the candidate keys.**

|  |
| --- |
| **{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.**

|  |
| --- |
| **NOTE : Any relation “R” can have either single or multiple candidate keys.** |

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

**For example : Consider the relation R(A, B, C, D) with functional dependencies :**

**FD1 : A 🡪 BC**

**FD2 : B 🡪 C**

**FD3 : D 🡪 C**

**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 of the to the candidate key, it will remain unaffected). Those attributes, which if removed does not affect closure of that set.**

**Canonical Cover 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.

### ****Case-02: 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-02: 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.

## ****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