Finding Attribute Closure and Candidate Keys using Functional Dependencies

Here will find the attribute closure and also we will find the candidate keys using the functional dependency. We will look into this topic in detail. But before proceeding to this topic, we will first learn about what is functional dependency.

A functional dependency **X->Y** in a relation holds if two tuples having the same value for X also have the same value for Y i.e. X uniquely determines Y. Consider the table given below.

In the EMPLOYEE relation given in Table,

- Functional Dependency **E-ID->E-NAME** holds because, for each E-ID, there is a unique value of E-NAME.
- Functional Dependency **E-ID->E-CITY** and **E-CITY->E-STATE** also holds.
- Functional Dependency **E-NAME->E-ID does not hold** because E-NAME 'John' is not uniquely determining E-ID. There are 2 E-IDs corresponding to John (E001 and E003).

Table EMPLOYEE

E-ID	E-NAME	E-CITY	E-STATE
E001	John	Delhi	Delhi
E002	Mary	Delhi	Delhi
E003	John	Noida	U.P.

Table 1: The FD set for EMPLOYEE relation given in Table 1 are:

{E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY->E-STATE}

Trivial and Non-Trivial Functional Dependency

Trivial Functional Dependency: A trivial functional dependency is one which will always hold in a relation. In the example given above, **E-ID, E-NAME->E-ID** is a trivial functional dependency and will always hold because {E-ID, E-NAME} ⊃ {E-ID}. You can also see from the table that for each value of {E-ID, E-NAME}, the value of E-ID is unique, so {E-ID, E-NAME} functionally determines E-ID.

Non-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 that holds in the above relation.

Properties of Functional Dependencies

Let *X*, *Y*, and *Z* be 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 $X \rightarrow Y$. 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 $X \rightarrow Y$, then $XZ \rightarrow YZ$. 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 $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$. 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.
- **Attribute Closure:** The set of attributes that are functionally dependent on the attribute A is called **Attribute Closure** of A and it can be represented as **A+.**

Steps to Find the Attribute Closure

Q.1. Given the FD set of a Relation R, The attribute closure set S is the set of Attribute Closure A.

- Add A to S.
- Recursively add attributes that can be functionally determined from attributes of the set S until done.

E-ID	E-NAME	E-CITY	E-STATE
E001	John	Delhi	Delhi
E002	Mary	Delhi	Delhi
E003	John	Noida	U.P.

- From Table 1, FDs are

Given R (E-ID, E-NAME, E-CITY, E-STATE)
FDs = { E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY->E-STATE }

The attribute closure of E-ID can be calculated as:

- Add E-ID to the set {E-ID}
- Add Attributes that can be derived from any attribute of the set. In this case, E-NAME and E-CITY, E-STATE can be derived from E-ID. So these are also a part of the closure.
- As there is one other attribute remaining in relation to be derived from E-ID. So the result is:

(E-ID)+ = {E-ID, E-NAME, E-CITY, E-STATE }

Similarly,

(E-NAME)+ = {E-NAME}
(E-CITY)+ = {E-CITY, E_STATE}

Q.2 Find the attribute closures of given FDs R(ABCDE) = {AB->C, B->D, C->E, D->A}.

To find (B)+, we will add an attribute in the set using various FDs which have been shown in the table below.

Attributes Added in Closure	FD used
{B}	Triviality
{B, D}	B->D
{B, D, A}	D->A
{B, D, A, C}	AB->C
{B, D, A, C, E}	C->E

- We can find (C, D)+ by adding C and D into the set (triviality) and then E using(C->E) and then A using (D->A) and the set becomes.

(C,D)+ = {C,D,E,A}

- Similarly, we can find (B, C)+ by adding B and C into the set (triviality) and then D using (B->D) and then E using (C->E), and then A using (D->A) and set becomes

(B,C)+ ={B,C,D,E,A}

Candidate Key

Candidate Key is a **minimal set of attributes** of a relationship that can be used to identify a **tuple** uniquely. For Example, each tuple of EMPLOYEE relation given in Table 1 can be uniquely identified by **E-ID** and it is minimal as well. So it will be the Candidate key of the relationship.

A candidate key may or may not be a primary key. Super Key is a **set of attributes** of a relationship that can be used to identify a tuple uniquely. For Example, each tuple of EMPLOYEE relation given in Table 1 can be uniquely identified by **E-ID or (E-ID, E-NAME) or (E-ID, E-CITY) or (E-ID, E-STATE) or (E_ID, E-NAME, E-STATE),** etc. So all of these are super keys of EMPLOYEE relation.

Note: A candidate key is always a super key but vice versa is not true.

Q.3 Finding Candidate Keys and Super Keys of a Relation using FD set.

The **set of attributes** whose attribute closure is a set of all attributes of the relation is called the super key of the relation. For Example, the EMPLOYEE relation shown in Table 1 has the following FD set. **{E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY->E-STATE}.** Let us calculate the attribute closure of different sets of attributes:

(E-ID)+ = {E-ID, E-NAME,E-CITY,E-STATE}
(E-ID,E-NAME)+ = {E-ID, E-NAME,E-CITY,E-STATE}
(E-ID,E-CITY)+ = {E-ID, E-NAME,E-CITY,E-STATE}
(E-ID,E-STATE)+ = {E-ID, E-NAME,E-CITY,E-STATE}
(E-ID,E-CITY,E-STATE)+ = {E-ID, E-NAME,E-CITY,E-STATE}
(E-NAME)+ = {E-NAME}
(E-CITY)+ = {E-CITY,E-STATE}

As **(E-ID)+**, **(E-ID, E-NAME)+, (E-ID, E-CITY)+**, **(E-ID, E-STATE)+**, **(E-ID, E-CITY, E-STATE)+** give set of all attributes of relation EMPLOYEE. So all of these are super keys of relation.

The minimal set of attributes whose attribute closure is a set of all attributes of relation is called the candidate key of the relation. As shown above, (E-ID)+ is a set of all attributes of relation and it is minimal. So E-ID will be the candidate key. On the other hand (E-ID, E-NAME)+ also is a set of all attributes but it is not minimal because its subset (E-ID)+ is equal to the set of all attributes. So (E-ID, E-NAME) is not a candidate key.