Roll no. 65

Date: 22/4/23
Page:

EXPERIMENT – 6

AIM: Implementation of attribute closure algorithm.

THEORY: In database management systems, attribute closure is an important concept related to functional dependencies. The attribute closure of a set of attributes is the set of all attributes that can be determined by those attributes through a set of functional dependencies

A functional dependency is a constraint between two sets of attributes in a relation. It states that if two tuples have the same value for the first set of attributes, they must also have the same value for the second set of attributes. For example, if we have a relation R(A,B,C) with the functional dependency A→B, it means that if two tuples have the same value for A, they must also have the same value for B.

The attribute closure of a set of attributes is important because it can help us determine the superkeys and candidate keys of a relation. A superkey is a set of attributes that can uniquely identify a tupel in a relation, while a candidate key is minimal superkey (i.e., a superkey that contains no unnecessary attributes). The attribute closure of a set of attributes can help us determine if that set of attributes is a superkey or a candidate key.

To find the attribute closure of a set of attributes, we need to apply the functional dependencies to that set of attributes repeatedly until we can no longer add any new attributes. For example, if we have a relation R(A,B,C,D) with the functional dependencies $A \to B$ and $B \to C$, the attribute closure of A would be {A,B,C}.

In summary, the attribute closure of a set of attributes is the set of all attributes that can be determined by those attributes through a set of functional dependencies. It is an important concept in database management systems because it can help us determine the superkeys and candidate keys of a relation.

ALGORITHM :

Step1: Initialize the closure of A as A+ = A

Step2: Repeat until no more attributes can be added to A+;

a. For each functional dependency $X \to Y$ in F;

i. If X is a subset of A+, then add Y to A+;

Step3: Return A+

classmate

Date :

Page :

Example : We are given the relation $R(A,B,C,D,E)$. This means that the table R has five columns: A, B, C, D and E. We are also given the set of functional dependencies: $\{A \to B, B \to C, C \to D, D \to E\}$.

To find attribute of A or find A^+.

- First, we add A to $\{A\}^+$
- We have $A \to B$, so we can determine B. Therefore, A^+ is now $\{A,B\}$
- We have $B \to C$, so we can determine C. Therefore $\{A\}^+$ is now $\{A,B,C\}$.
- Now, we have A, B, C, D and $D \to E$, so we can add D to $\{A\}^+$
- Now, we have used all of the columns in R and we have all used all functional dependencies. $\{A\}^+ = \{A,B,C,D,E\}$.

Attribute Closure Code (Java) :

```
import java.util.*;
 class prac6aashish{
    public static void main(String[] args) {
        // char[] R={'A','B','C','D','E'};  //Relations
        String[][]
FD={{"A","BC"},{"CD","E"},{"B","D"},{"E","A"}};  //functional dependencies
        String[] closure={"A","B","E","C","D","BC","BD"};
        int i,j,k,l,count=0,flag=-1;
        for(i=0;i<closure.length;i++)
        {
            Set<Character> AC=new HashSet<Character>();
            for(char c: closure[i].toCharArray())
                AC.add(c);
            for(j=0;j<FD.length;j++)
            {
                ArrayList<Character> A=new ArrayList<Character>(AC);
                for(k=0;k<FD[j][0].length();k++)
                {
                    for(int m=0;m<A.size();m++)
                    {
                        if(FD[j][0].charAt(k)==A.get(m))
                        {
                            flag=0;
                            break;
                        }
                        else if(m==A.size()-1)
                        {
                            flag=1;
                            break;
                        }
                    }
                    if(flag==1)
                    break;
                }
                if(flag==0)
                {
                    int a=A.size();
                    for(l=0;l<FD[j][1].length();l++)
                    {
                        AC.add(FD[j][1].charAt(l));
                    }
                    int b=AC.size();
                    if(a==b)
                    {
                        count++;
                    }
                }
```

```
                else if(flag==1)
                {
                    count++;
                }
                if(count==FD.length && j==FD.length-1)
                {
                    count=0;
                    break;
                }
                else if(count!=FD.length && j==FD.length-1)
                {
                    j=-1;
                    count=0;
                }
            }
            System.out.println("Attribute closure of "+closure[i]+" is "+AC+"\n");
        }

    }
}
```

Output :

```
PS D:\GCOEN SEM 4\Database Management System L
Attribute closure of A is [A, B, C, D, E]

Attribute closure of B is [B, D]

Attribute closure of E is [A, B, C, D, E]

Attribute closure of C is [C]

Attribute closure of D is [D]

Attribute closure of BC is [A, B, C, D, E]

Attribute closure of BD is [B, D]
```

CONCLUSION: Here we have used a relation $R = \{A, B, C, D, E\}$ and 4 functional dependencies FD, where $FD = \{A \rightarrow BC,\ CD \rightarrow E,\ B \rightarrow D,\ E \rightarrow A\}$ and found out the attribute closure Ac of A, B, C, D, E, BC and BD and successfully implemented the algorithm of attribute closure.