CSLR51 – Database Management Systems Laboratory #Session: 08 || Date: 03/10/2024
Viva Due: Q.No. 2 (03/10/2024) Moodle Due: 09/10/2024 at 11 PM

Question 2:
Develop an implementation package that would contribute to a normalization setup by generating the Candidate key(s) and Super key(s) in a Relation given the Functional Dependencies.
Your code should work for any given FD's, not just for the given sample below.
Example:
Given R(X Y Z W) and FD = { XYZ → W, XY → ZW and X → YZW}
 Candidate key: {X};
Super keys: {X, XY, XZ, XW, XYZ, XYW, XZW, XYZW}

Given R(X Y Z W) and FD = {X→Y, Y→Z, Z→X}
Candidate keys: {WX, WY, WZ};
Super keys: {WXY, WXZ, WYZ, WXYZ}

Improved solution according to the algorithm given in the standard book:

```
#include<stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define max_attr 4
#define max_fd 3
#define max_len 3

char attributes[max_attr][max_len+1];
char left[max_fd][max_len+1];
char right[max_fd][max_len+1];
int nattr, nfd;
int found[26]={0};

void readInput() {
    printf("Enter number of attributes: ");
    scanf("%d", &nattr);
    printf("Enter attributes (space separated): ");
    for (int i = 0; i < nattr; i++) {
        scanf("%s", attributes[i]);
        if(isalpha(attributes[i][0]))
        found[tolower(attributes[i][0])-'a']=1;
    }
    printf("Enter number of functional dependencies: ");
    scanf("%d", &nfd);
    getchar();  // Consume the newline character left by scanf
    for (int i = 0; i < nfd; i++) {
        printf("Enter the functional dependencies LHS for %d :", i);
        scanf("%s",left[i]);
    }
    for(int i = 0; i < nfd; i++){
        printf("Enter the functional dependencies RHS for %d :", i);
        scanf("%s",right[i]);
    }
}

int isSubset(const char *set, const char *subset) {
    for (int i = 0; subset[i] != '\0'; i++) {
        if (!strchr(set, subset[i])) {
            return 0; // Not a subset
        }
    }
    return 1; // Is a subset
}

char *closure(char *result) {
    int changed;
    do {
        changed = 0; // Reset changed flag
        for (int i = 0; i < nfd; i++) {
            if (isSubset(result, left[i])) {
                for (int j = 0; right[i][j] != '\0'; j++) {
                    if (!strchr(result, right[i][j])) {
                        strncat(result, &right[i][j], 1);
                        changed = 1;
                    }
                }
            }
        }
    } while (changed);

    return result;
}

int Set(char * result)
{
    for(int i=0;i<nattr;i++)
    {
        if(strchr(result,attributes[i][0])==NULL)
        return 0;
    }
    return 1;
}

void generateSuperKeys(char * candidate){
    int totalKeys=1<<nattr;
    printf("Super keys:\n");
    for(int i=1;i<totalKeys;i++){
        char key[max_attr+1]="";
        for(int j=0;j<nattr;j++){
            if(i & (1<<j)){
                strncat(key,attributes[j],1);
            }
        }
        if(isSubset(key,candidate))
        {
            printf("%s\n",key);
        }
    }
    return;
}

int check_candidate_key(char * base,char * both,int add){
    char temp[5];
    strcpy(temp,base);
    int found=0;
    if(add==1){
    for(int i=0;both[i]!='\0';i++){
            char a[2]={both[i],'\0'};
            char temp1[5];
            strcat(base,a);
            strcpy(temp1,base);
            if(Set(closure(base))){
                printf("Candidate key is %s\n", temp1);
                found=1;
                generateSuperKeys(temp1);
                strcpy(base,temp);
            }
    }
    }
    else if(add==2){
    for(int i=0;both[i]!='\0';i++){
        for(int j=i+1;both[j]!='\0';j++){
            char a[3]={both[i],both[j],'\0'};
            char temp1[5];
            strcat(base,a);
            strcpy(temp1,base);
            if(Set(closure(base))){
                printf("Candidate key is %s\n", temp1);
                found=1;
                generateSuperKeys(temp1);
                strcpy(base,temp);
            }
        }
    }
    }
    else if(add==3){
    for(int i=0;both[i]!='\0';i++){
        for(int j=i+1;both[j]!='\0';j++){
            for(int k=j+1;both[k]!='\0';k++){
                char a[4]={both[i],both[j],both[k],'\0'};
                char temp1[5];
                strcat(base,a);
                strcpy(temp1,base);
                if(Set(closure(base))){
                    printf("Candidate key is %s\n", temp1);
                    found=1;
                    generateSuperKeys(temp1);
                    strcpy(base,temp);
                }
            }
        }
    }
    }
    return found;
}

void find() {
    int found_left[26] = {0};
    int found_right[26] = {0};

    for (int i = 0; i < nfd; i++) {
        for (int j = 0; j < strlen(left[i]); j++) {
            if (isalpha(left[i][j])) {
```

```
            found_left[tolower(left[i][j]) - 'a'] = 1;
        }
    }
    for (int j = 0; j < strlen(right[i]); j++) {
        if (isalpha(right[i][j])) {
            found_right[tolower(right[i][j]) - 'a'] = 1;
        }
    }
}

char left_attrs[max_len + 1] = "";
char right_attrs[max_len + 1] = "";
char both[max_len + 1] = "";
char not[max_len + 1] = "";

for (int i = 0; i < 26; i++) {
    char s[2] = {i + 'a', '\0'};
    if (found_left[i] && !found_right[i]) {
        strcat(left_attrs, s);
    } else if (!found_left[i] && found_right[i]) {
        strcat(right_attrs, s);
    } else if (found_left[i] && found_right[i]) {
        strcat(both, s);
    } else if(found[i]){
        strcat(not, s);
    }
}

printf("Attributes only on the left: %s\n", left_attrs);
printf("Attributes only on the right: %s\n", right_attrs);
printf("Attributes on both sides: %s\n", both);
printf("Attributes on neither side: %s\n", not);

strcat(left_attrs,not);
char left_attr[5];
strcpy(left_attr,left_attrs);
if(Set(closure(left_attrs))){
    printf("Candidate key is %s\n",left_attr);
    generateSuperKeys(left_attr);
}
else
{
    for(int i=1;i<=sizeof(both);i++){
        if (!check_candidate_key(left_attrs, both, i)) {
            printf("No candidate key found with %d characters added.\n",i);
            }
            else
            return;
        }
    }
}

int main() {
    readInput();
    find();
    return 0;
}
```

OUTPUT:

```
reethi@DESKTOP-8744EFO:~/dir1/dbms$ gcc l8_q2_new.c
reethi@DESKTOP-8744EFO:~/dir1/dbms$ ./a.out
Enter number of attributes: 4
Enter attributes (space separated): w x y z
Enter number of functional dependencies: 3
Enter the functional dependencies LHS for 0 :xyz
Enter the functional dependencies LHS for 1 :xy
Enter the functional dependencies LHS for 2 :x
Enter the functional dependencies RHS for 0 :w
Enter the functional dependencies RHS for 1 :wz
Enter the functional dependencies RHS for 2 :wyz
Attributes only on the left: x
Attributes only on the right: w
Attributes on both sides: yz
Attributes on neither side:
Candidate key is x
Super keys:
x
wx
xy
wxy
xz
wxz
xyz
Wxyz

reethi@DESKTOP-8744EFO:~/dir1/dbms$ gcc l8_q2_new.c
reethi@DESKTOP-8744EFO:~/dir1/dbms$ ./a.out
Enter number of attributes: 4
Enter attributes (space separated): w x y z
Enter number of functional dependencies: 3
Enter the functional dependencies LHS for 0 :x
Enter the functional dependencies LHS for 1 :y
Enter the functional dependencies LHS for 2 :z
Enter the functional dependencies RHS for 0 :y
Enter the functional dependencies RHS for 1 :z
Enter the functional dependencies RHS for 2 :x
Attributes only on the left:
Attributes only on the right:
Attributes on both sides: xyz
Attributes on neither side: w
Candidate key is wx
Super keys:
wx
wxy
wxz
wxyz
Candidate key is wy
Super keys:
wy
wxy
wyz
wxyz
Candidate key is wz
Super keys:
wz
wxz
wyz
wxyz
```