

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | CSLR 51 : DBMS LAB-7 | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Roll no. : 106119100Name : Rajneesh PandeySection : CSE-B |  | | | |
|  | | |  | | |  | | |

CODE

-------------------------------------------------------------------------

/\*

CSLR-51 : DBMS LAB-7

Roll no. : 106119100

Name : Rajneesh Pandey

Section : CSE-B

\*/

#include <bits/stdc++.h>

using namespace std;

vector<string> ans;

vector<int> mapps;

vector<int> nodes;

string relationship\_alpha;

int relationship\_alpha\_len;

int dependency;

unordered\_map<int, int> depends;

unordered\_map<char, int> alpha\_to\_int;

unordered\_map<int, char> int\_to\_alpha;

string mask\_to\_string(const int &mask)

{

    string str = "";

    for (int i = 0; i < relationship\_alpha\_len; i++)

    {

        if ((mask >> i) & 1)

            str += int\_to\_alpha[i];

    }

    return str;

}

int string\_to\_mask(const string &s)

{

    int mask = 0;

    for (auto &ch : s)

    {

        mask |= (1 << alpha\_to\_int[ch]);

    }

    return mask;

}

void init()

{

    cout << "Enter attributes in the Relationship: \n";

    cin >> relationship\_alpha;

    relationship\_alpha\_len = relationship\_alpha.length();

    mapps.assign(1 << relationship\_alpha\_len, 0);

    for (int i = 0; i < (1 << relationship\_alpha\_len); i++)

        mapps[i] = i;

    for (int i = 0; i < relationship\_alpha\_len; i++)

    {

        alpha\_to\_int[relationship\_alpha[i]] = i;

        int\_to\_alpha[i] = relationship\_alpha[i];

    }

    cout << "Enter total number of dependencies :\n";

    cin >> dependency;

    cout << "Enter dependencies : \n";

    for (int i = 0; i < dependency; i++)

    {

        cout<< i + 1 << " : ";

        string lhs, rhs;

        cin >> lhs >> rhs;

        depends[string\_to\_mask(lhs)] = depends[string\_to\_mask(lhs)] | string\_to\_mask(rhs);

    }

    cout << "\n\nFinished taking inputs.........\n";

    cout << "Processing.........\n";

}

/\* C+ = C; while (there is changes to C+)

    { do (for each functional dependency X->Y in F) { if (X⊆C+) then C+= C+∪Y } }

\*/

int get\_closures(int mask)

{

    int c = mask;

    int prevc = 0;

    while (c != prevc)

    {

        prevc = c;

        for (auto &ele : depends)

        {

            if ((c & ele.first) == ele.first)

                c |= ele.second;

        }

    }

    return c;

}

void get\_closures\_all()

{

    for (int i = 0; i < (1 << relationship\_alpha\_len); i++)

    {

        mapps[i] = get\_closures(i);

    }

}

void get\_keys()

{

    bool found = false;

    int total = (1 << relationship\_alpha\_len) - 1;

    for (int len = 1; len <= relationship\_alpha\_len; len++)

    {

        vector<int> perm(relationship\_alpha\_len, 0);

        for (int t = relationship\_alpha\_len - len; t < relationship\_alpha\_len; perm[t++] = 1)

            ;

        do

        {

            int mask = 0;

            for (int i = 0; i < relationship\_alpha\_len; i++)

                if (perm[i])

                    mask |= (1 << i);

            if (mapps[mask] == total)

            {

                found = true;

                ans.push\_back(mask\_to\_string(mask));

            }

        } while (next\_permutation(perm.begin(), perm.end()));

        if (found)

            break;

    }

    cout << "\nCandicate Keys are: \n";

    for (auto &cands : ans)

        cout << cands << " ";

}

int main()

{

    init();

    get\_closures\_all();

    get\_keys();

    return 0;

}

---------------------------------------------------------------

Problem 1

1. 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}} on R. What is the key for R?

A computer screen capture

Description automatically generated with medium confidence

Problem 2

Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R? write a c program to find it.

**A computer screen capture

Description automatically generated with medium confidence**