International Journal Of Advance Research, Ideas And Innovations In Technology

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 4)
Available online at: www.ijariit.com

A method to discover constant conditional functional dependencies for a given relation

Sneha Sadalagi
sadalagisneha@gmail.com
Acharya Institute of Technology, Department of CSE, Bengaluru, Karnataka

Dr. P V Kumar
veereswarakumar@acharya.ac.in
Acharya Institute of Technology, Department of CSE, Bengaluru, Karnataka

ABSTRACT

This CCFD paper explores the idea of discovering the constant conditional functional dependencies (CCFDs). Constant CFDs are particularly important for object identification, which is essential to data cleaning and data integration. The further algorithm provides a set of cleaning rule discovery tools for users to choose it for different applications. These CCFDs are derived from conditional functional dependencies (CFDs). The algorithm is implemented for finding the constant conditional functional dependencies from a given relation.

Keywords— *Functional dependency (FD), Conditional functional dependency (CFD), Constant conditional functional dependency (CCFD), Data cleaning*

1. INTRODUCTION

Nowadays finding poor data quality has become a major problem in many organizations. Finding inconsistency of the data is the problem and it also leads to a problem in poor business decisions resulting from the poor data quality. Therefore, recently CFDs are helpful for detecting and repairing inconsistent data in a relational database. Conditional functional dependencies are the recent extension of functional dependencies. As conditional functional dependencies are used for data cleaning, CFDs require manual effort for finding the quality data. Hence to solve the CFD problem it is necessary to discover the problem of CFDs by finding constant patterns only. Conditional functional dependencies are divided into constant conditional functional dependencies and variable conditional functional dependencies.

.

In the paper, we discover constant CFDs which are useful for object identification and for data integration. We are implementing an algorithm which provides the set of constant conditional functional dependencies. In the paper initial step is to create the tables in SQL and finding their respective counting the values and accordingly creating different tables with respect to the distinct values. Finally finding the constant conditional functional dependencies from the given relation is the functionality of the paper.

2. LITERATURE SURVEY

In the paper [1] the authors explained the integrity class constraints which are used for relational databases, and that is referred to as conditional functional dependencies (CFDs) and it explains the study of their applications in data cleaning. It also explains the functional dependencies (FDs) which are mainly used for developing schema design.

In the paper [2] the authors have been proposed a new type of semantic rules extended from the traditional functional dependencies. In this paper, it shows great potential for detecting and repairing inconsistent data. This paper uses constant CDFs for making the 100% confidence association rules.

In the paper [3] the authors presented the data mining techniques in the area of the data cleaning to discover the constant conditional functional dependencies (CCFDs) from the relational databases. These are used as business rules for context-dependent data validations.

In the paper [4] the authors presented the techniques for developing CFDs from relations. Here they provided three methods for discovering CFDs. The first method is CFDMiner, it is the technique which is based on mining closed item sets and is used for constant patterns only. The second algorithm is CTANE which is referred to as a levelwise algorithm that extends TANE and this algorithm is used for mining FDs. The third algorithm is FastCFD which is based on a depth-first approach for discovering FastFDs.

Example 1: The following relational schema **customer** data is taken from [1]. This specifies the customer relation. In this, the attributes are CC-Country Code, AC-Area Code, PN-Phone Number, and CT-City.

Traditional FDs shown in r0 (Table 1) includes the following:

$$f: [CC, AC] \rightarrow CT$$

Here f is a Functional dependency (FD) with the same country and area code which leads to the same city.

Table 1: An instance r0 of the cust relation

CC	AC	PN	CT
01	908	111	mh
01	212	222	ny
01	908	222	mh
44	131	333	ed
44	131	444	ed
44	908	444	mh
01	131	222	un

Φ0: ([CC, AC]→CT, (44, 131||ed))
Φ1: ([CC, AC]→CT, (01,908||mh))
Φ2: ([CC, AC] →CT, (44, 131||ed))
Φ3: ([CC, AC]→CT, (01,131||un))
Φ4: ([CC, AC]→CT, (01, 212||ny))

In ϕ0 & Φ2, (44, _||_) is the pattern tuple that enforces a semantically related constant which binds (CC, AC, CT) in a tuple. It states that for customers in the UK, AC uniquely determines CT. It is FD that only holds on the subset of tuples with pattern "CC=44," instead of entire relation in r0. In case of ϕ1 explains that for any customer in the US CC=01 with AC=908, the CT of the customer must be mh only, which enforces its pattern tuple as (01, 908||mh).

Definition of CFDs: A CFD which is denoted as φ over R is the pair (A→B, tp) where,

(i) A is the set of attributes in attr(R), and B is the single attribute in attr(R).
(ii) X→A is a standard functional dependency, referred to as the functional dependency embedded in φ.
(iii) tp is a pattern tuple with attributes A and B, where for each C in A∪{B}, tp[C] is either a constant "a" in dom(C) or an unnamed variable "_" that draws values from dom(C).

3. PROPOSED WORK

Problem Statement: "Discovering Constant Conditional Functional dependencies"

Definition of CCDF: A CFD (X→A, tp) is said to be constant CFD if and only if its pattern tuple tp consist of constants only. i.e., tp[A] is a constant.

Consider the following codes for the representation of attributes as CC=1, AC=2, PN=3, CT=4.

Table 2: An example for CCFDs

1	2	3	4
1	1	1	1
1	2	2	2
1	1	2	1
2	3	3	3
2	3	4	3
2	1	4	1
1	3	2	4

Certain examples for constant conditional functional dependencies from Table 2.

CCFD1= (1, 1||1)
1, 2→4 (t1, t3)
CCFD2= (1, 2||2)
1, 2→4 (t2)
CCFD3= (2, 3||3)
1, 2→4 (t4, t5)
CCFD4= (2, 1||1)
1, 2→4 (t6)
CCFD5= (1, 3||4)
1, 2→4 (t7)

In CCFD1-CCFD5 all the pattern tuple is related with constants which binds (1, 2 and 4) attributes. It explains that attribute 4=1 occurs as constant when the attributes 1=1 and 2=1 occurs as constants.

4. METHODOLOGY

In this section, the CCFD algorithm is implemented in SQL query language and also C programming, which uses Oracle 11G and Turbo C as the software. SQL is the standard language for querying a relational system database. SQL allows the user to access the database and execute the given query. C programming language is a compiled language which is also called a high-level assembly language.

Algorithm for discovering Constant CFDs:

1. Create the structure for the table (T1).
2. Insert the raw data into a table (T1).
3. Generate the Power Set for the attributes in the table (PST1).
4. (i) For every element (X) in Power Set, find Count(X), which is a number of distinct values of (X).
   (ii) Generate table (T2) for the distinct count values, along with considered attribute list.
5. Generate four tables basing on the number of attributes in each element of power set. (T21, T22, T23, T24).
6. Transfer all the tuples from T21, T22, T23, and T24 into T3.
7. For any two rows from table T3, select a row (R1) and select another row (R2) from table T3 such that:
   (i) The count values are same for both R1 & R2.
   (ii) The attribute list (P1) in (R1) is a subset of the attribute list (P2) in another row (R2).
   Here it can be said that P1→ P2-P1 is a Functional dependency (FD). [If X⊆Y where X, Y ϵ PS, and Count (X) = Count (Y), then X→ Y-X is called a Functional dependency (FD)]
8. Following the same logic, generate the set of all possible functional dependencies using the table (T3).
9. Consider only the FDs having at least two attributes in the left-hand side of the FDs.
10. Find and display all the Constant conditional functional dependencies for each FD.

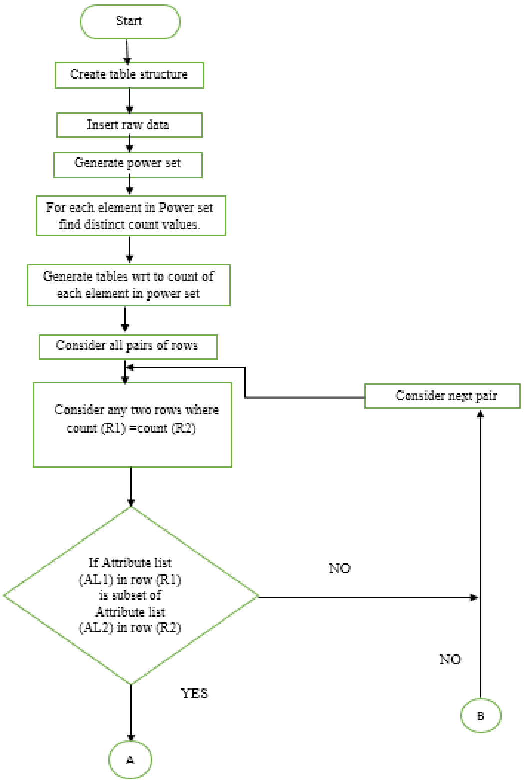


Fig. 1: Flowchart for CCFD

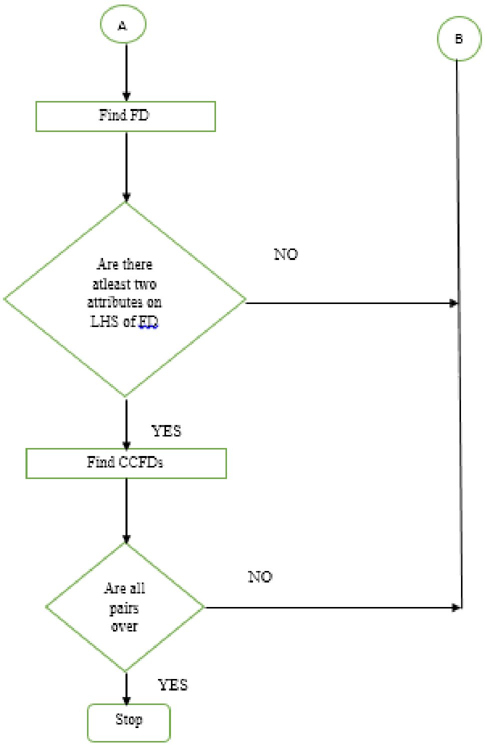
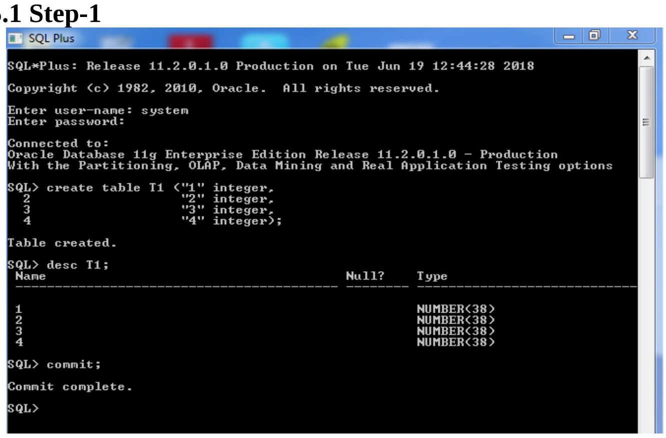


Fig. 2: Shows the flowchart for constant conditional functional dependencies for the given relation

5. CASE STUDY

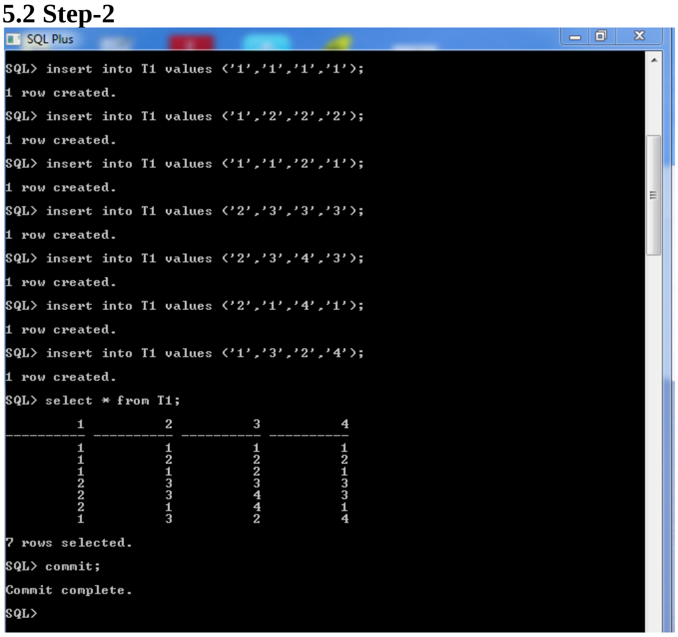
5.1 Step-1



```
SQL*Plus: Release 11.2.0.1.0 Production on Tue Jun 19 12:44:28 2018

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> create table T1 ("1" integer,
  2                   "2" integer,
  3                   "3" integer,
  4                   "4" integer);

Table created.

SQL> desc T1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 1                                                  NUMBER(38)
 2                                                  NUMBER(38)
 3                                                  NUMBER(38)
 4                                                  NUMBER(38)

SQL> commit;

Commit complete.

SQL>
```

Snapshot 1: Structure of table

5.2 Step-2



```
SQL> insert into T1 values ('1','1','1','1');

1 row created.

SQL> insert into T1 values ('1','2','2','2');

1 row created.

SQL> insert into T1 values ('1','1','2','1');

1 row created.

SQL> insert into T1 values ('2','3','3','3');

1 row created.

SQL> insert into T1 values ('2','3','4','3');

1 row created.

SQL> insert into T1 values ('2','1','4','1');

1 row created.

SQL> insert into T1 values ('1','3','2','4');

1 row created.

SQL> select * from T1;

         1          2          3          4
---------- ---------- ---------- ----------
         1          1          1          1
         1          2          2          2
         1          1          2          1
         2          3          3          3
         2          3          4          3
         2          1          4          1
         1          3          2          4

7 rows selected.

SQL> commit;

Commit complete.

SQL>
```

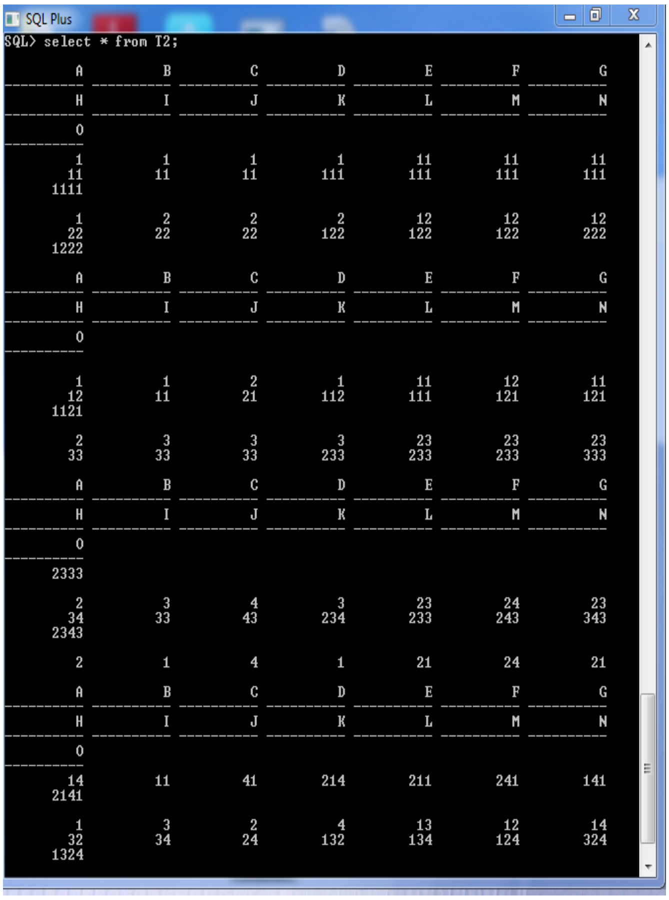
Snapshot 2: Input data

5.3 Step-3

```
SQL> select substr(sys_connect_by_path(col, ','), 2) as power_set from test1
  2  connect by prior col < col;

POWER_SET
--------------------------------------------------------------------------------

1
1,2
1,2,3
1,2,3,4
1,2,4
1,3
1,3,4
1,4
2
2,3
2,3,4

POWER_SET
--------------------------------------------------------------------------------

2,4
3
3,4
4

15 rows selected.

SQL> commit;

Commit complete.

SQL>
```

Snapshot 3: Power set

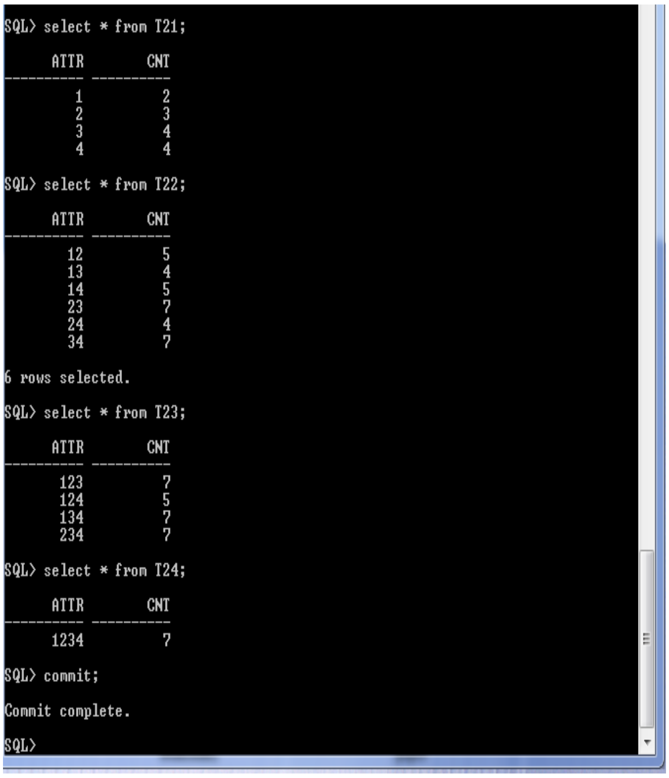
5.4 Step-4



```
SQL> select * from T2;

         A          B          C          D          E          F          G
---------- ---------- ---------- ---------- ---------- ---------- ----------
         H          I          J          K          L          M          N
---------- ---------- ---------- ---------- ---------- ---------- ----------
         O
----------
         1          1          1          1         11         11         11
        11         11         11        111        111        111        111
      1111

         1          2          2          2         12         12         12
        22         22         22        122        122        122        222
      1222

         A          B          C          D          E          F          G
---------- ---------- ---------- ---------- ---------- ---------- ----------
         H          I          J          K          L          M          N
---------- ---------- ---------- ---------- ---------- ---------- ----------
         O
----------

         1          1          2          1         11         12         11
        12         11         21        112        111        121        121
      1121

         2          3          3          3         23         23         23
        33         33         33        233        233        233        333
         A          B          C          D          E          F          G
---------- ---------- ---------- ---------- ---------- ---------- ----------
         H          I          J          K          L          M          N
---------- ---------- ---------- ---------- ---------- ---------- ----------
         O
----------
      2333

         2          3          4          3         23         24         23
        34         33         43        234        233        243        343
      2343

         2          1          4          1         21         24         21
         A          B          C          D          E          F          G
---------- ---------- ---------- ---------- ---------- ---------- ----------
         H          I          J          K          L          M          N
---------- ---------- ---------- ---------- ---------- ---------- ----------
         O
----------
        14         11         41        214        211        241        141
      2141

         1          3          2          4         13         12         14
        32         34         24        132        134        124        324
      1324
```

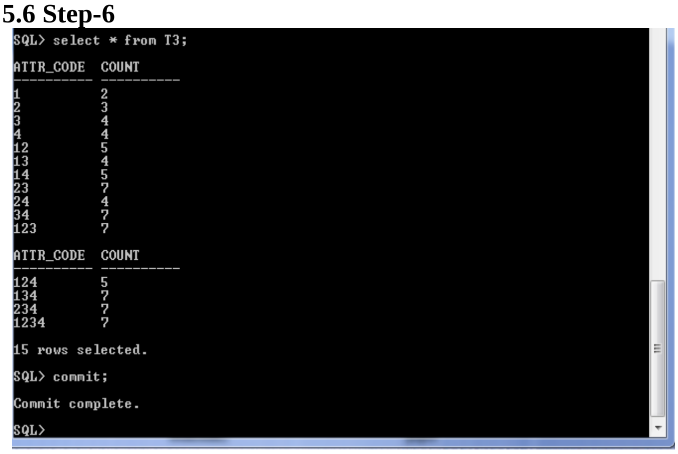
Snapshot 4: Inserting update values for counting distinct values

5.5 Step-5



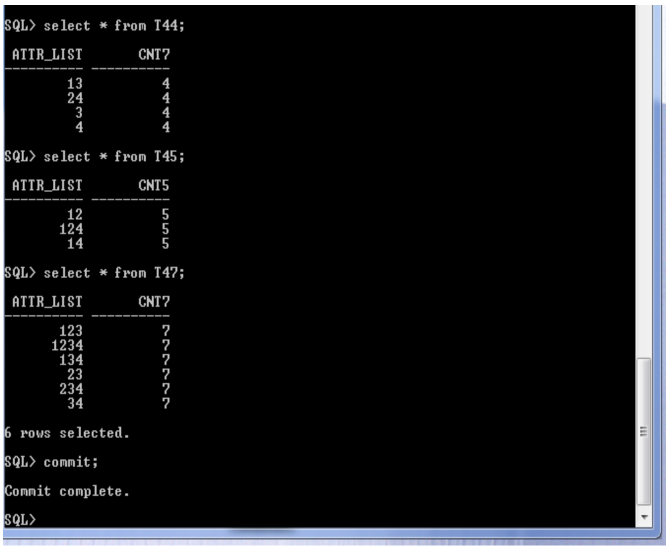
Snapshot 5: Update distinct values with different tables

5.6 Step-6



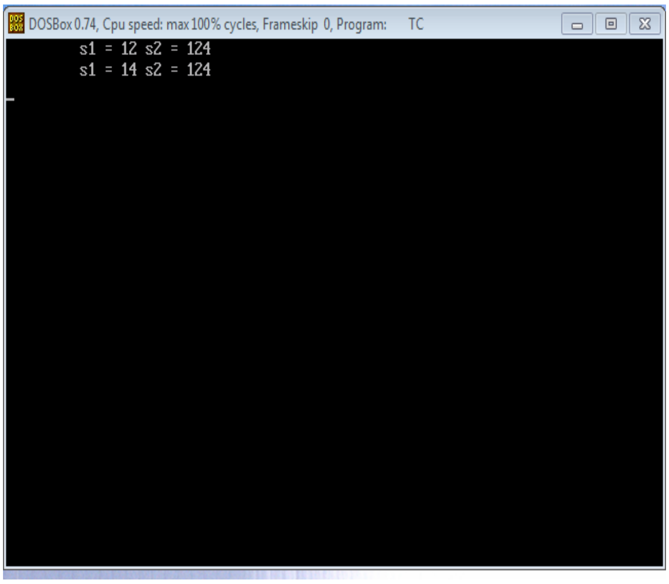
Snapshot 6: Distinct count values with attribute names

5.7 Step-7



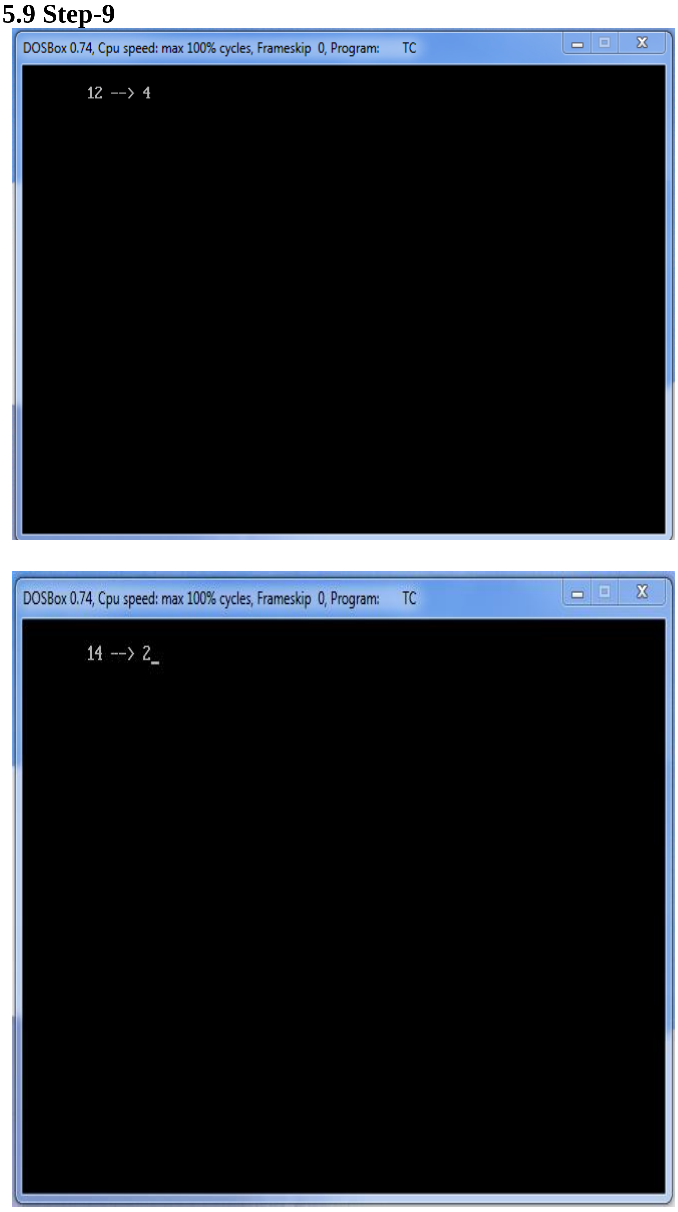
Snapshot 7: Tables for count=4, count=5 and count=7

5.8 Step-8



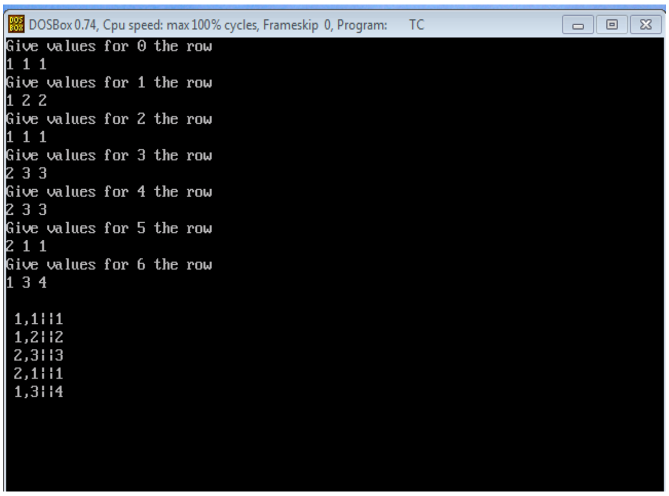
Snapshot 8: Comparing subset with respect to count=5, for finding functional dependencies

5.9 Step-9



Snapshot 9: Output for FDs

5.10 Step-10



Snapshot 10: Output for CCFDs for FD 1 2→4

5.11 Step 11: 1, 2→4

Find and display all the Constant conditional functional dependencies for each FD.

1, 1 || 1 [T1,T3]
2, 3 || 3 [T4,T5]
1, 2 || 2 [T2]
2, 1 || 1 [T6]
1, 3 || 4 [T7]

6. CONCLUSION

We have developed and implemented an algorithm for discovering constant conditional functional dependencies, which is important for both finding inconsistencies in the data and also for data cleaning. This implementation helps to avoid the complexity of the data in industries and business issues. It also provides a set of tools for users to choose from different applications. Further, we plan to discover the variable conditional functional dependencies for fixing rules for data cleaning.

7. REFERENCES

[1] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, “Conditional Functional Dependencies for Capturing Data Inconsistencies”, ACM Trans. Database Systems (TODS), vol. 33, no. 2, pp. 1-48, 2008.

[2] Jiuyong Li, Jixue Liu, Hannu Toivonen and Jianming Yong, “Effective Pruning for the Discovery of Conditional Functional Dependencies”, The Computer Journal, Vol. 56 No. 3, 2013.

[3] D Devi Kalyani, “Mining Constant Conditional Functional Dependencies for Improving Data Quality”, International Journal of Computer Applications (0975-8887), Volume 74-No.15, July 2013.

[4] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong, “Discovering Conditional Functional Dependencies”, IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 5, May 2011.

[5] Rashed Salem, Asmaa Abdo, “Fixing rules for data cleaning based on conditional functional dependency”, Future Computing and Informatics Journal 1 (2016) 10-26 Menoufia University, Egypt.

