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.

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.
Consider the following codes for \( \Phi_0 \) & \( \Phi_2 \), (44, _||_) is the pattern tuple that enforces \( \Phi_4 \): \([CC, AC]\).

\[ \Phi_0: (CC, AC) \rightarrow CT, (44, 01)||mh) \]

In \( \Phi_0 \) & \( \Phi_2 \), (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 \( r_0 \). In case of \( \Phi_1 \) 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 \( \phi \) over R is the pair (A \( \rightarrow \) B, \( tp \)) where, 

(i) A is the set of attributes in \( \text{attr(R)} \), and B is the single attribute in \( \text{attr(R)} \).

(ii) \( X \rightarrow A \) is a standard functional dependency, referred to as the functional dependency embedded in \( \phi \).

(iii) \( tp \) is a pattern tuple with attributes A and B, where for each C in \( \text{Au(B)} \), \( tp[C] \) is either a constant “a” in \( \text{dom(C)} \) or an unnamed variable “_” that draws values from \( \text{dom(C)} \).

**3. PROPOSED WORK**

Problem Statement: “Discovering Constant Conditional Functional dependencies”

**Definition of CCDF:** A CFD \((X \rightarrow A, tp)\) is said to be constant CFD if and only if its pattern tuple \( tp \) consists 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>
<thead>
<tr>
<th>Table 2: An example for CCFDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Certain examples for constant conditional functional dependencies from Table 2.

CCFD1= \((1, 1)||1\)

1, 2 \( \rightarrow \)4 (t1, t3)

CCFD2= \((1, 2)||2\)

1, 2 \( \rightarrow \)4 (t2)

CCFD3= \((2, 3)||3\)

1, 2 \( \rightarrow \)4 (t4, t5)

CCFD4= \((2, 1)||1\)

1, 2 \( \rightarrow \)4 (t6)

CCFD5= \((1, 3)||4\)

1, 2 \( \rightarrow \)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 \(P1 \rightarrow P2-P1\) is a Functional dependency (FD). \([\text{If } X \subseteq Y \text{ where } X, Y \epsilon PS, \text{and Count(X)} = \text{Count(Y)}\) then \(X \rightarrow Y-X\) 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.

---

© 2018, www.IJARIIT.com All Rights Reserved
5. CASE STUDY

5.1 Step-1

5.2 Step-2

5.3 Step-3

5.4 Step-4

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
5.11 Step 11: 1 \rightarrow 4
Find and display all the Constant conditional functional dependencies for each FD.

\begin{align*}
1, 1 & \rightarrow [T1,T3] \\
2, 3 & \rightarrow [T4,T5] \\
1, 2 & \rightarrow [T2] \\
2, 1 & \rightarrow [T6] \\
1, 3 & \rightarrow [T7]
\end{align*}

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