A Result Paper on Investigation of Incremental Detection Problems in Distributed Data

Mr.Dattatray Raghunath Kale, Mr. Satish R.Todmal

Abstract—**In current years we found huge complexities and data cleaning problems in the distributed data. In centralized or distributed, we cannot decrease the data consignment in small period. Interruption has occurred when using constraints in large amount of database. Data in real-life is often dirty. Errors, conflicts and inconsistencies are occurred in distributed data. Previous work cannot be expressed standard methods on data cleaning. The Conditional functional dependencies (CFD's) that are used for capturing the inconsistencies that traditional dependencies fail to catch. We show that the incremental detection problem is NP-complete for the distributed database that is partitioned either vertically or horizontally. We show some incremental algorithms for vertically partitioned and horizontally portioned data.**

Index Terms— **Distributed data, Conditional Functional Dependencies, error detection.**

I. Introduction

In real life, data is always found as a dirty, Errors, conflicts and inconsistencies are always originating in the real life data. In Australia 500,000 dead people maintain active Medicare cards. In US Pentagon asked 275 dead/wounded officers to reenlist?UK: there are 81 million National Insurance numbers but only 60 million eligible citizens. It is estimated that in a500,000 customer database120,000 customer records become invalid within a year, due to deaths, divorces, marriages, moves. Typical data error rate in industry: 1% - 5%, up to 30%..

One of the vital technical difficulties is how to tell whether the data is unclean or clean. Specify consistency using integrity constraints. Inconsistencies emerge as violations of constraints. Constraints considered so far: traditional they are, functional dependencies, inclusion dependencies, denial constraints (a special case of full dependencies). Dirty data is serious problem that influences many enterprises across all portions of their business ranging from process competence to revenue security. Even though, the underlying ideas are not new, but this is the first challenge to develop a real algorithm for discovering both regulations from a large datasets. We used two new pruning techniques in our algorithm to decrease the number CFDs to be checked hence improving its performance.

Manuscript received Dec, 2015.

Dattatray Raghunath Kale, *Department of computer engineering, imperial college of engineering & research, Pune, India, 9604386625.*

Satish R. Todmal*, Department of computer engineering, imperial college of engineering & research, Pune, India, 9960488094,*

The first technique merges the similar CFDs for discovering a few and more accurate rules, while the second technique finds the minimal set of CFDs based on the intersect partitions between the candidates that formed the rules Data superiority is recognized as one of the most significant difficulties for data organization. A central technical problem for data quality concerns inconsistency detection, to identify errors in the data. Further specifically, known a database DB and a set of dependencies serving as data quality rules is to discover all the violations of in DB i.e., all the tuples in DB that violate a number of rules in .When DB is a centralized database, the finding difficulty is not very tough. Consider, for example, conditional functional dependencies (CFDs) [1] that were newly proposed as data quality rules. We look into problems associated with CFDs We focus on consistency analysis. Data cleaning is the proficient detection of constraint violations in the data. Here a given relation DB is vertically partitioned into the different fragments residing at different sites. Then we have to checks all the CFDs. For moments it requires to ship the data from one site to another site. Our most important aim is to reduce the data shipment procedure by merging the alike CFDs for discovering a few and more accurate rules.

II. Related Work

In related work, Hammer and Sarm [HS78] gives a technique for efficiently detecting violations of integrity constraints, called integrity assertions, as a result of database updates. For each integrity assertions, there exists an error predicate which corresponds to the logical match of the assertion if the error-predicate is true for some instance of the database, and then the instance violates the assertion. There are a few advances for resolving data inconsistency in data combination based on the content of the conflicting data [2] .It identifies the continuation of data inconsistencies and give users with a few additional information on their character. It decides the data inconsistency through use of probabilistic data. Another one method is DQM (Data Quality Manager).DQM will include a Reference Model, a Measurement Model, and an Assessment Model to describe the quality criteria, the metrics and the evaluation techniques.DQM will also help in query planning by considering data quality inferences to find the best combination for the implementation plan. After query execution, and discovery of inconsistent data, data quality might also be utilized to do data inconsistency declaration.[3]. The work in Bohannon et al. [2005] is focuses on repairing inconsistencies based on typical FDs and addition dependencies, that is, to check over the instance via minimal value alteration such that the updated instance satisfies the constraints Since CFDs are further communicative than FDs, not all of the detected CFD violations can be repaired by the algorithm of Bohannon et al. [2005].A current addition of Bohannon et al. [2005] studies repairing inconsistent databases based on CFDs [Cong et al.2007]. Compared to the results of Bohannon et al. [2005], it has been demonstrated in Cong et al. [2007] that CFDs are further efficient than usual FDs in identifying and repairing real-life inconsistent data. Closer to CFDs is the tableau representation of dependencies [Wijsen 2005].This effort characterizes full dependencies by tableaux that also permit data values.Franconi et al. also argue restores based on updating personage values, in the framework of a data cleaning function. The aim is to calculate all probable repairs, in this case of an exacting kind of databases storing survey data, rather that steady query answering. The issues addressed consist of detecting and solving conflicts within the database and conflicts between replies to questionnaires and the planned declarative semantics of the latter, has opposed to conflicts between data and integrity constraints. This effort is an exact case of data cleaning [4].

MapReduce[10] is a programming mold and an associated implementation for processing and generating big data sets. Users state a map function that processes key/value pair to make a set of in-between key/value pairs and a decrease function that merges all middle key. Programs written in this functional method are automatically parallelized and executed on a huge cluster of product machines. The run-time system acquires care of the details of partitioning the input data, scheduling the programs execution transversely a set of machines, handling machine failures, and managing the required inter-machine communication. This permits programmers without any knowledge with similar and distributed systems to simply make use of the reserves of a huge distributed system. [5].

Conditional functional dependencies (CFDs) were proposed for data cleaning. It was exposed there that given a set of CFDs, a fixed number of SQL queries can be routinely generated, which are capable to become aware of violations of the CFDs in a centralized database in polynomial time. The SQL methods were generalized to detect violations of CFDs [6], an addition of CFDs by supporting disjunctions and contradictions. As mentioned previous, the SQL techniques do not be enough to identify CFD violations in fragmented and distributed relations, a practical setting. There has also been work on discovering CFDs data repairing with CFDs and CFD propagation via views. However, no previous work has studied how to detect CFD violations in distributed databases, a topic far more challenging than its centralized matching part.

Recently, there has been work on detecting distributed constraint violations for monitoring distributed systems [7].While planning to diminish communication cost, the effort varies substantially from our work in that the constraints in [7] are defined on system states and cannot express CFDs;in contrast, CFDs are to detect errors in data, which is usually much better than organization conditions. Thus, the algorithm in [7] is not applicable for CFD violation detection in distributed data.

There has been a mass of effort on query processing (see,e.g.,[8]) and distributed query processing. The number of algorithms has been extended for generating (distributed) query plans, mostly focusing on how to efficiently perform joins. inspection CFD violations in horizontally partitioned data do not involve join operations, and thus we do not have to give the price of full-fledged query arrangement generators in this context. Nevertheless, query processing methods can be applied to violation detection in vertically partitioned data, for which joins are frequently essential.

The main idea of multi-query optimization, in either centralized databases or distributed databases [8], is to take out and assembly general sub-queries to decrease evaluation cost, and to schedule data group to minimize the communication cost.[9] Likewise, when dealing with numerous CFDs, we merge CFDs with overlapping outlines into one. Additional, we distribute detection procedures to numerous sites to enlarge the parallelism. The fix techniques of multi-query optimization can be used when detecting violations of multiple CFDs in vertical fragments. Dependency preservation has been studied for lossless decompositions of relational schemas. In this work we revisit the issue for characterizing locally checkable CFDs in vertical fragments. A number of NP-complete results have been established for distributed query processing.

III. IMPLEMENTATION DETAILS

There having some limitations in the previous work. The batch algorithms used in the previous system are outperform and when the data updates are very large then batch algorithms do not better as expected. The data shipment problem is always occurred. The distributed CFD detection problem occurred with maximum communication cost. Our proposed system overcomes all these problems occurred in the previous work.

In this Section, we have described the aggregation technique in detail. We have discussed the design goals of our proposed system and the integration of the proposed technique with the Hadoop framework.

A. *System Analysis and Implementation Strategy*

- Dataset collection: Our System will consist of large datasets. So our first step is to collect the large datasets. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable.
- Preprocessing: Clustering the datasets has been used in this processing. Here we have load the data and then Preprocess the raw data.
- Conditional Functional Dependencies: CFDs allow data binding, a large number of individuals constraints may hold on a table, complicating detection of constraints violations.
- Error Detection: Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.
- Data Distribution: A database that consists of two or more data files located at different sites on a computer network. Because the database is

distributed, different users can access it without interfering with one another. However, the CFD must periodically synchronize the scattered databases to make sure that they all have consistent data.

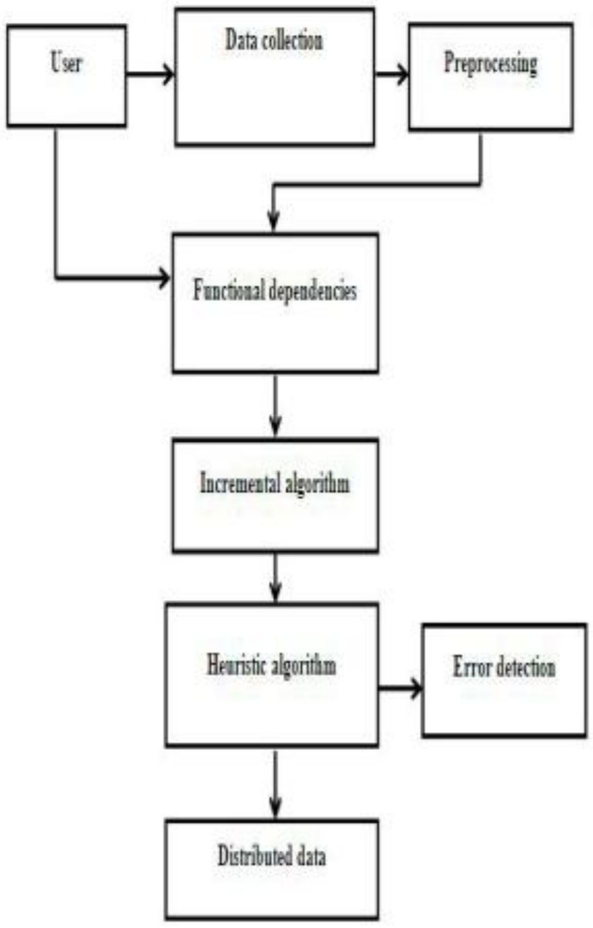


Fig. 1. System Flow

B. *Algorithmic Approach for finding Inconsistencies:*

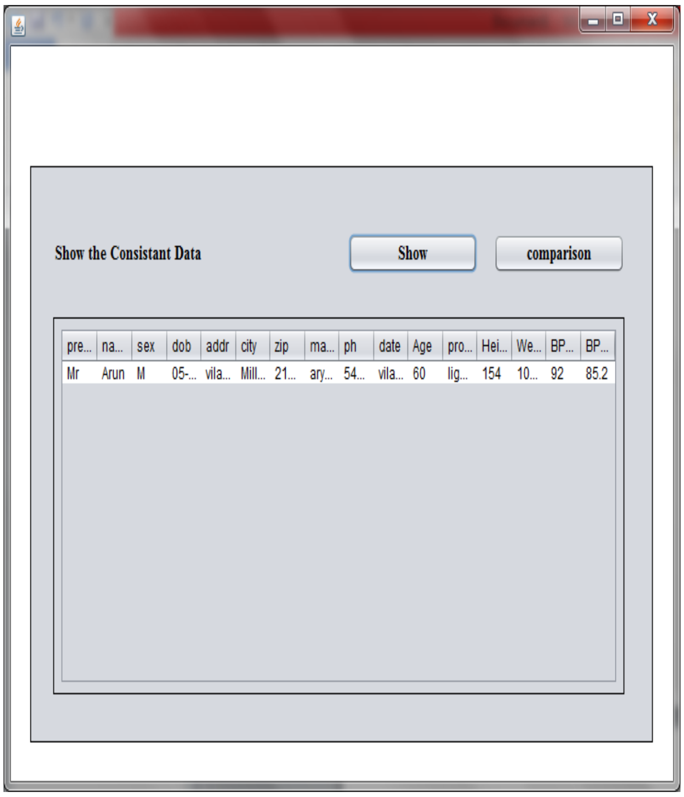
This algorithm is used as a heuristic algorithm. Its innovation is that it is one to find the way through which we can find out the inconsistent data and without assuming the end user to know the schema or the data to hold to a schema. The Algorithm is like as follows
Heuristic algorithm:
Input: Set of CFD’s and database D
Output: set of violations i.e. tuples that violate at least one CFD’s will be deleted.
1) Given first we input a value denoted by keyword K.
2) An equivalence class is a set of “cells/tuples” defined by a tuple say t and attribute say A’s value
3) Find all tuples/cells that matches the Keyword K.
4) Calculate Support values Si for each instance of Attribute A and assign each Attributes the support values.
5) Calculate Weight value Wi on basis of support values Si assigned for each instance of Attribute A and assign each Attribute the Weight values.
6) Then calculate the threshold value T for all instances of tuples in equivalence set E.
7) The tuple Ti that has Weight scores Wi less than threshold T calculated they are detected to be in consistent violating one or more CFD’s.
8) Inconsistent data tuples T={t1,t2,….ti} are removed from database D.
9) Only tuples T{t1,t2,….ti} will consistent data will be showed.

C. *Experimental Setup*

We give the details and specification of the hardware on which the system is expected to work. We use Processor Pentium IV 866 MHz, RAM 250 MB and hard disk 80 GB.We use Windows XP Professional operating system and Java as programming language.

IV. Result

Our first module is based on Dataset collection and preprocessing the datasets. Most commonly a data set corresponds to the contents of a single database table or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set.Here we have taken database of health information of various people and apply the incremental and heuristic approach for finding out the inconsistent data. After the detection of inconsistency in data we are removing it with showing the only consistent data. The results of both approaches are compared and shows on the graphical format.



pre...	na...	sex	dob	addr	city	zip	ma...	ph	date	Age	pro...	Hei...	We...	BP...	BP...
Mr	Arun	M	05-...	vila...	Mill...	21...	ary...	54...	vila...	60	lig...	154	10...	92	85.2

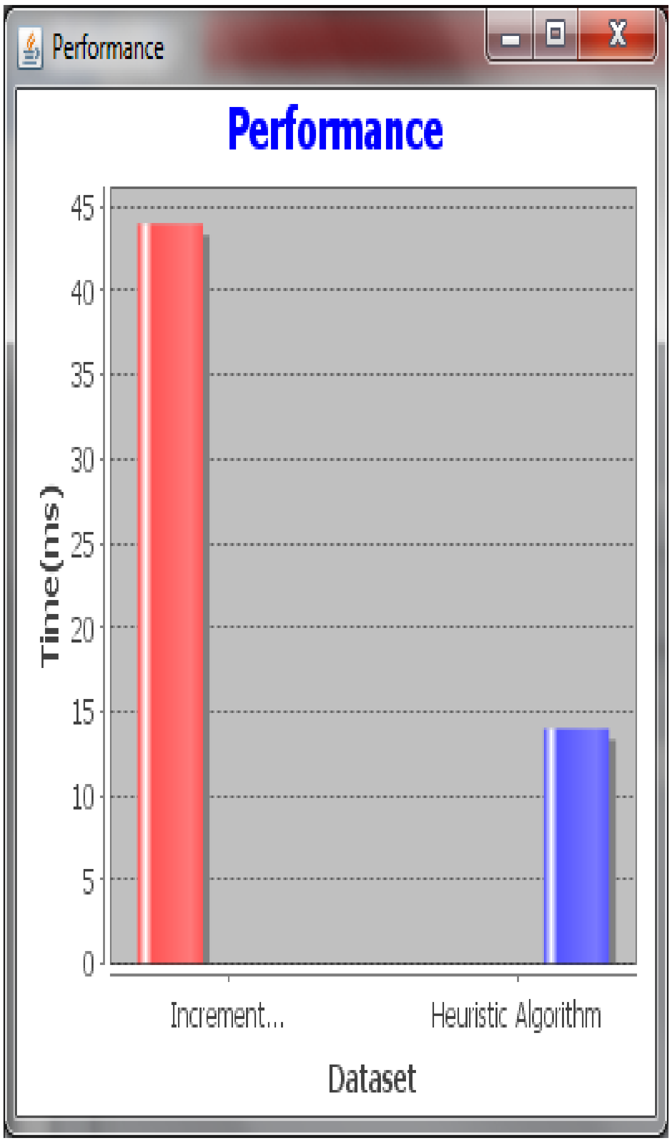


Fig2: Experimental Results

V. CONCLUSION

We have studied the problem of detecting CFD violations in distributed databases. The innovation of our effort contains
in a formulation of CFD violation detection as optimization problems to minimize data shipment or response time, to detect CFD violations in vertically partitioned data, aiming to
minimize either data shipment or response time. As verified by our experimental results, the algorithms scale well in the size of data, the number of fragments, and the complexity of CFDs,and hence provide active methods for catching inconsistencies in distributed data.

VI. ACKNOWLEDGMENT

We are giving a lot of thanks to all authors; those are mentioned in the references and all the respected people who helped us for designing and development of our work.

REFERENCES

[1] Gartner,Forecast: Data quality tools, worldwide, 2006-2011, 2007.

[2] Conditional Functional Dependencies for Capturing Data Inconsistencies by WENFEI FAN University of Edinburgh and Bell Laboratories FLORIS GEERTS and XIBEI JIA University of Edinburgh.

[3] P. Anokhin, Data inconsistency detection and resolution in the integration of het erogeneous information sources, Ph.D. Thesis, School of Information Technology and Engineering, George Mason University, 2001.

[4] Detection and Resolution of Data Inconsistencies, and Data Integration using Information Quality criteria. Maria del Pillar Angeles School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh,EH14 4AS.K. Elissa, Title of paper if known, unpublished.
.
[5] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C-A.Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In International Conference on Very Large Data Bases, pages 371, 380, 2001.

[6] MapReduce: Simplified Data Processing on Large lusters,Jeffrey Dean and Sanjay Ghemawat rejeff@google.com, sanjay@google.com.

[7] S. Agrawal, S. Deb, K. V. M. Naidu, and R. Rastogi, Efficient detection of distributed constraint violations, in ICDE, 2007.

[8] A. Kementsietsidis, F. Neven, D. V. de Craen, and S. Vansummeren,Scalable multi-query optimization for exploratory queries over federated scientific databases, in VLDB, 2008.

[9] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe, Efficient and extensible algorithms for multi query optimization, in SIGMOD, 2000

[10] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas,MRShare:Sharing across multiple queries in MapReduce, Proc.VLDB, vol. 3, no.12, pp. 494505, Sept. 2010.